6-29-2020

Demystifying Search Rank Fraud

Nestor G. Hernandez
Florida International University, nhern121@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DEMYSTIFYING SEARCH RANK FRAUD

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Nestor Hernandez

2020

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Nestor Hernandez, and entitled Demystifying Search Rank Fraud, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Mark Finlayson

Naphtali Rishe

Wensong Wu

Leonardo Bobadilla

Bogdan Carbunar, Major Professor

Date of Defense: June 29, 2020

The dissertation of Nestor Hernandez is approved.

Dean John L. Volakis
College of Engineering and Computing

Andres G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2020

DEDICATION

To my dad Hilario Hernandez and my mom Maria Hernandez.

ACKNOWLEDGMENTS

I want to start by thanking my advisor Dr. Bogdan Carbunar for all his invaluable support, encouragement, and patience throughout my PhD studies. He introduced me to Cyber Security and Privacy and gave me the freedom to explore different problems within the field. He provided key insights and direction on the research side, and taught me how to do academic research. It has been a great positive learning experience to witness how he conducts research.

I would also like to thank my PhD fellow and co-author Ruben Recabarren who has been an excellent collaborator and friend. I have been lucky to learn from him, not only about Cyber Security but about all aspects of computer science and life. He is one of the smartest individuals with whom I have worked.

I am grateful to my collaborators outside of FIU including Dr. Duen Horng Chau (Georgia Tech), and Dr. Ishtiaque Ahmed (University of Toronto). They provided me with the opportunity to collaborate in academia and learn from their approaches to solve and understand problems.

Some Faculty and Staff at FIU's School of Computing and Information Sciences have been very helpful and nice to work with during my studies. In particular, I want to thank Professors Mark Finlayson, Naphtali Rishe, and Leonardo Bobadilla for being part of my dissertation committee and for their great lectures and courses. Dr. Wensong Wu from the Statistics and Mathematics department has also been very helpful and has provided another perspective on my dissertation. I thank them for their insightful feedback on my dissertation. Additionally, I want to thank Catherine Hernandez, John Flynn, and Rebeca Arocha for their professionalism as part of the SCIS staff.

I want to also thank my labmates and classmates Mizanur Rahman, Mozhgan Azimpurkivi, Sajedul Talukder, Gregory Reis, and Deya Banishaker for making the PhD journey more enjoyable and for their insights and views on research ideas.

Finally, thanks to my dad Hilario Hernandez and my mom Maria Hernandez for all their support. They continue to teach me about life and support me in uncountable ways. I thank my friends Hector Aguilera, Jose Vollmann, Pablo Karg, Miguel Mossa, Alejandro Castillo, Farah Alarcon, Carlos Contreras, and Jose Mendoza for incredible friendship.

ABSTRACT OF THE DISSERTATION

DEMYSTIFYING SEARCH RANK FRAUD

by

Nestor Hernandez

Florida International University, 2020

Miami, Florida

Professor Bogdan Carbunar, Major Professor

Search rank fraud, i.e., the posting of large numbers of fake activities for products hosted in commercial peer-opinion services such as those provided by Google, Apple, Amazon, seeks to give the illusion of grassroots engagement, and boost financial gains, promote malware and even assist censorship efforts. Search rank fraud continues to be a significant problem, after years of investment from service providers and the academic community.

In this thesis we envision that knowledge of the authentic capabilities, behaviors and strategies employed by empirically validated workers, will enable us to develop solutions that efficiently manage and contain search rank fraud, by detecting, classifying and neutralizing its effects. We posit that to be effective, fraud detection and classification efforts need to involve the organizations and individuals who contribute to search rank fraud.

In this thesis we therefore engaged with professional workers to (1) collect ground truth knowledge and evaluate defenses, (2) develop fraud detection and classification solutions that adapt to rater strategy changes, and (3) attribute fraud to the organizations that posted it. More specifically, we first performed qualitative and quantitative investigations with professional workers, concerning activities they performed on Google Play. We reveal findings concerning various aspects of worker capabilities and behaviors, including novel insights into their working patterns. We confirm the existence of power workers who control many devices and user accounts, and also the emergence of organic workers, i.e., almost-regular users who occasionally promote products from the devices and accounts that they also use for personal purposes.

In a second contribution we develop RacketStore, a framework to capture detailed insights about how Google Play Store users use their devices and the apps installed therein. We use RacketStore to develop and evaluate the first solutions that disentangle organic from federated fraud, and from honest behaviors. Specifically, we use data collected from installations of RacketStore on 803 devices to show that features that model the user interaction with a device can be used to distinguish devices controlled by organic workers from those of power workers and regular users of the Google Play service.

In a third contribution we introduce a fraud de-anonymization approach to disincentivize fraud perpetrated by power workers: attribute user accounts used to promote apps to the human workers in crowdsourcing sites, who control them. We model fraud de-anonymization as a maximum likelihood estimation problem and develop a graph based deep learning approach to predict ownership of account pairs by the same fraudster. We introduce the first cheating-resistant fraud de-anonymization validation protocol, that transforms human fraud workers into ground truth, performance evaluation oracles.

The success of the approach proposed in this thesis suggests that the next generation of fraud detection and prevention solutions will benefit from the integration of validated professional workers into the problem modeling, solution design and evaluation processes.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

THESIS VISION

1.1 Introduction

The ubiquitous adoption of social networks and peer-opinion sites (e.g., Google services, app markets, Facebook, Twitter) is changing how we find partners, buy products online, access information from the news, and organize to demand political change. However, such sites are also susceptible to abuse in the form of private information collection and misuse, cyberbullying, political manipulation, fake news and opinion spamming. Such service providers rely on user feedback to rank products and content they host over the Internet, e.g., trending topics on Twitter, popular apps on Google Play, trending YouTube videos, etc. Unfortunately, many review-based platforms (e.g., Google Play [Rei17a], TripAdvisor [Ros17], Amazon [Woo17], Twitter [CDHH18]) are the targets of undisclosed and deceptive marketing practices whereby product developers engage in artificial promotion and demotion of products. We call this attack search rank fraud.

On the other hand, black hat crowdsourcing or *crowdturfing* offers a viable opportunity for developers to hire specialized workers who spam for profit [Wam17, YVC$^+$17, LWG14, WWZZ14, TZX$^+$15]. Thus, crowdsourcing websites (e.g., Fiverr, Freelancer, Upwork) [Fiv, Upw, Fre] play an important role in the fake endorsement ecosystem since they facilitate the communication between product owners who try to engineer their products' rankings and workers who control hundreds of accounts to artificially alter the online sentiment of such products. Similarly, Facebook groups have become a source of fraud workers where they advertise services for several online platforms including Google Play, Google Maps, Yelp, Amazon, etc. Case in point, Figure 1.1(a) shows the per-group distribution of number of

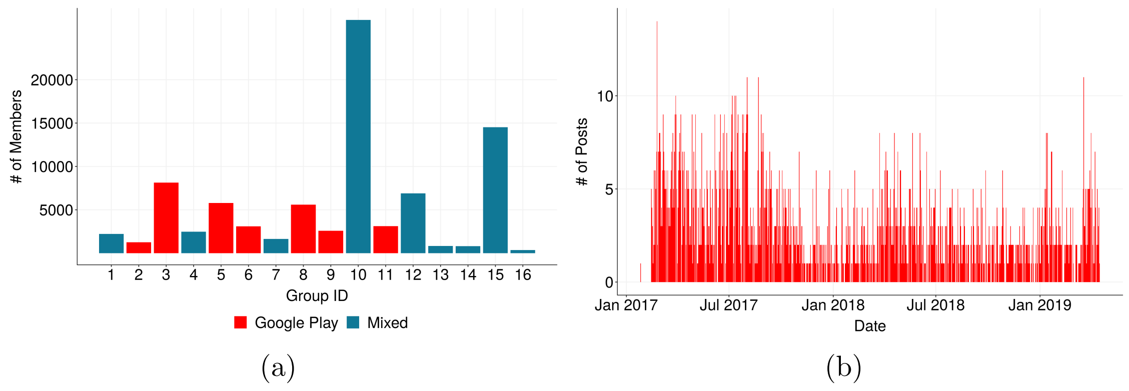


Figure 1.1: (a) Per-group distribution of number of members for 16 fraudulent Facebook groups we infiltrated. Red bars represent groups that exclusively promote Google Play fraud while the rest advocate for fake reviews in an array of services including Yelp, Amazon, and Google Maps. (b) Number of daily posts that advertise fraudulent campaigns for a group that focus exclusively on Google Play fraud.

members for 16 fraudulent Facebook groups that we identified and infiltrated. Figure 1.1(b) displays the number of daily posts for a group that focus only on Google Play promotion. This suggests that fraud detection is not being enough to prevent the fake endorsement problem in online services.

This type of propaganda has a detrimental effect on the trustworthiness and quality of online services, and users can suffer from such bait-and-switch schemes. For this reason, most major online, peer-opinion services seek to detect and remove fake reviews that result from hidden endorsements [NA16, SCM11, MVLG13], which are unlawful in accordance with FTC regulations [1].

Search rank fraud continues to be a significant problem [Par18, Mah19, Fer20], after years of investment from service providers [Cip16, Per16, Woo20] and the academic community (see Chapter 3 for related work).

We posit that one reason for this failure stems from our misunderstanding and underestimation of the capabilities, behaviors, and strategies of the professional

[1]If the endorser has been paid or given something of value to promote the product, the connection between the marketer and endorser should be disclosed [FTC]

raters recruited to perform search rank fraud: existing work is built on assumptions about professional raters, that are either extracted from small datasets of fraud, made based on intuition, or revealed by commercial site insiders. We have recently challenged these assumptions, in qualitative studies that we performed with professional raters that target Google services [RHR+19, HRRC18, RHCC18]. We found raters who evolved fraud-posting strategies that circumvent and even exploit key assumptions made by fraud detection work (§ 3). This makes some raters particularly successful. For instance, 90% of 1,164 Google Play accounts that 39 professional raters revealed to provably control, were still active one year later.

1.2 The Vision

In this thesis we envision that knowledge of the authentic capabilities, behaviors and strategies employed by empirically validated workers, will enable us to develop solutions that efficiently manage and contain search rank fraud, by detecting, classifying and neutralizing its effects.

Our thesis is that to be effective, fraud detection and classification efforts need to involve the organizations and individuals who contribute to search rank fraud. Therefore, in this thesis we engage with professional workers to (1) collect ground truth knowledge and evaluate defenses, (2) develop fraud detection and classification solutions that adapt to rater strategy changes, and (3) attribute fraud to the organizations that posted it.

1.3 Challenges

To realize the above vision we need to address several challenges:

Fraud diversity. Fraud detection and classification solutions need to flexibly target diverse types of fraud organizations, behaviors and strategies, such as the ones that we found in preliminary studies [RHR$^+$19, HRRC18]. Examples include (1) *federated fraud*, carried out by raters who organize in mostly static teams (see Figure 4.3 for a photo of a team's brick-and-mortar offices) and post fraud from hundreds of mobile devices and tens of thousands of user accounts that they pool, and (2) *organic fraud*, generated by individual operators with personal accounts and devices, who mix fraud among genuine activities, and form ad-hoc teams.

Binary classification is not enough. The remarkable success of fraud suggests that the current, binary classification of activities, e.g., fake vs. honest reviews, fraudulent vs. genuine accounts, followed by the removal of detected fraud, fails to stop prolific federated raters, who can easily create new accounts and post new fraud. Further, the decentralized nature of organic fraud enables it to elegantly evade status quo assumptions, e.g., that fraud produces synchronized, lockstep behaviors or suspicious activity spikes.

Training and evaluation of developed solutions. Commercial platforms are close-sourced, and their Terms of Service (ToS) prohibit posting fraudulent activities. However, fraud detection and classification solutions need to be trained using large sets of ground truth data, and, importantly, need to be evaluated in production-like environments, under real-time fraud posting conditions.

1.4 Contributions

In this thesis, we (1) present empirical data from actual ASO workers through interviews and quantitative analyses, understand their behaviors, capabilities, and avoidance strategies; (2) develop a framework to collect data from, and compare the app and device use of ASO workers and regular users; and (3) propose to discourage search rank fraud instead of merely discovering it, by de-anonymizing the organizations behind it. Note that (2) allows us to collect fraud information directly from the perpetrators devices eliminating the possibility of false positive artifacts and subject claims that are impossible to corroborate otherwise. In the following, we introduce each contribution.

1.4.1 A Study of Worker Capabilities and Behaviors

We first present the design and results of a qualitative study with 18 ASO workers we recruited from 5 freelancing sites, concerning activities they performed on Google Play, along with a quantitative investigation with fraud-related data collected from other 39 ASO workers. We reveal findings concerning various aspects of ASO worker capabilities and behaviors, including novel insights into their working patterns, and supporting evidence for several existing assumptions. Further, we found and report participant-revealed techniques to bypass Google-imposed verifications, concrete strategies to avoid detection, and even strategies that leverage fraud detection to enhance fraud efficacy. We report a Google site vulnerability that enabled us to infer the mobile device models used to post more than 198 million reviews in Google Play, including 9,942 fake reviews. We discuss the deeper implications of our findings, including their potential use to develop the next generation fraud detection and prevention systems.

1.4.2 RacketStore: Demystifying Device Use

In this thesis we conjecture that worker interactions with promoted apps differ from the regular, personal use of installed apps; further, that organic workers use their devices in a manner distinguishable from power workers dedicated to promotion activities, and regular users. To validate these hypotheses, we develop RacketStore, a framework to capture detailed insights about how Google Play Store users use their devices and the apps installed therein.

We report results from a study over 943 installs of RacketStore on 803 unique devices controlled by 580 workers and 223 regular users that we recruited online, including the 12,341 apps installed on their devices and their 110,511,637 reviews from Google Play. We provide novel insights into the device usage and installed-app interaction of workers and regular users, that confirm our hypotheses. We introduce features that model the use of a device and of the apps on the device. We show that supervised algorithms distinguish between apps suspected of promotion and personal-use apps, with an F1-measure that exceeds 99%. We show that most of the evaluated devices are controlled by organic workers, but also report devices controlled by power workers, and introduce a new type of exchange workers who magnify their capabilities by exchanging fraud with peers without payment. Our approach brings to light 217,041 reviews posted from 10,310 Gmail accounts registered on 580 ground-truth worker devices, suggesting potential benefit of integration with existing app store defenses.

1.4.3 De-Anonymization of Power Fraud

We propose to discourage fraud instead of merely discovering it. To this end, as illustrated in Figure 1.2, we seek to bridge the anonymity gap between existing

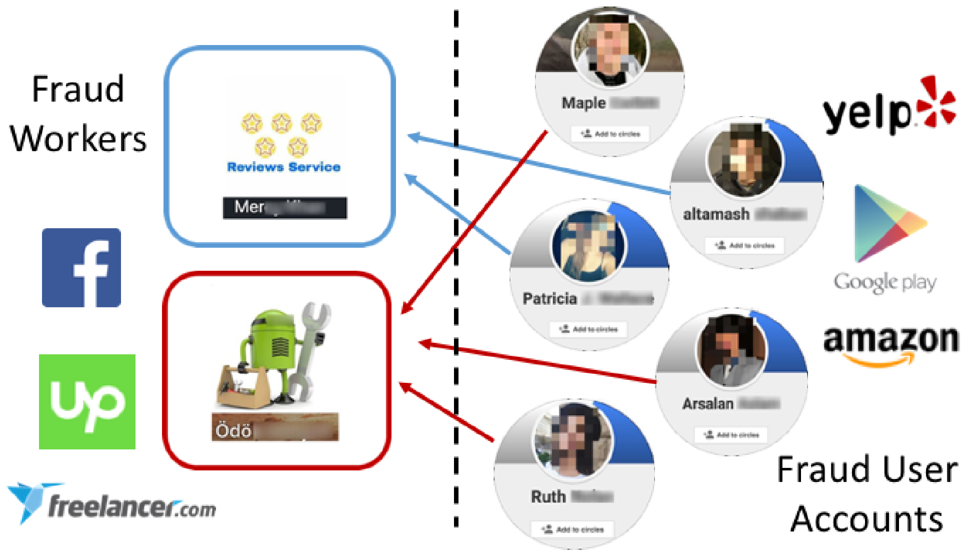


Figure 1.2: DETEGO de-anonymizes fraud. Fraud detection only identifies suspicious user accounts on the right. Fraud de-anonymization also finds the crowdsourcing account (left side) that controls them. Arrows signify control.

fraud detection techniques, that only uncover pseudonymous user accounts that post fraud, and the real identities of crowdsourcing site accounts who control them. Specifically, we leverage the observation that crowdsourcing site accounts contain uniquely identifying payment information, e.g., bank, Paypal accounts, to take steps toward de-anonymizing fraud, by attributing accounts uncovered by fraud detection algorithms in online peer-opinion systems, to their human owners in crowdsourcing sites.

We propose a general theoretical framework for the fraud de-anonymization problem via Maximum Likelihood Estimation (MLE) and assume a generative review-posting model wherein fraudster-controlled accounts are more likely to endorse products in a predefined partition of the product space. We introduce UODA, an unconstrained optimization de-anonymization approach that attributes a fraudulent user account to the fraud worker with the highest likelihood of having generated its review history.

We develop DeepCluster, a semi-supervised approach to cluster user accounts based on deep learning features extracted from the common activities of the accounts. We leverage DeepCluster to build a *co-ownership predictor* that determines if two input user accounts are controlled by the same worker. We use the co-ownership predictor to introduce (1) DDA, a discriminative de-anonymization solution that trains a classifier to attribute a fraudulent user account to the worker who controls it, and (2) PFD, a pseudonymous fraudster discovery algorithm that clusters fraudulent accounts that cannot be attributed to known workers, such that each cluster is likely controlled by a different, not yet discovered worker.

We introduce DETEGO [2], a system that combines fraud de-anonymization with fraudster discovery to iteratively expand both knowledge of identifiable fraud workers and the accounts that they control. We believe that DETEGO can help peer-review sites identify the experts from among hundreds of advertised fraud workers, who control large numbers of user accounts, and are responsible for posting substantial numbers of fake reviews. Peer-review sites can use this information to provide counter-incentives for expert fraudsters, e.g., by pursuing them through their bank accounts (retrieved from their crowdsourcing site accounts). Peer-review sites can also disincentivize developers from hiring such identifiable fraudsters, e.g., by "shaming" promoted products with posts displaying information about the fraudsters found to promote them [Yel12].

To validate developed solutions, we introduce the first cheating-resistant, *fraud de-anonymization validation* protocol, to obtain ground truth confirmation on the performance of developed solutions. The protocol asks human fraud workers to reveal a seed set of user accounts that they control, and subsequently confirm and prove control of accounts that we predict that they control. We introduce multiple

[2] Latin for uncover, reveal.

verifications of participant attention and honesty, including asking confirmations for accounts for which we already know the answer, as well as e-mail and token based verifications.

1.5 Thesis Outline

The main topics of this this thesis are organized into 6 Chapters. Chapter 2 introduces the system and adversary model, and some terminology used throughout the rest of the chapters. Chapter 3 provides an overview of the existing literature in the fields of author identification, cross-site identity linking, fraud and opinion spamming detection, and data collection from underground markets that offer fraudulent services. We describe previous approaches used to deal with the fake review problem across multiple platforms and contrast them with our proposed solutions.

In Chapter 4, we present a fraud workflow map based on findings from a qualitative and a quantitative studies that we conducted with 18 and 39 ASO workers respectively. We analyze empirical data from actual workers, to advance our understanding of their work. Specifically, we report findings on the capabilities and behaviors exhibited by workers including concrete strategies to avoid detection. Finally, we disclose vulnerability points in the fraud workflow and discuss their potential use to advance current fraud detection and prevention solutions.

Chapter 5 describes RacketStore, a framework to study the interaction of users with their Android devices and the apps that they install, that proved compatible with 298 device models from 28 Android manufacturers. We present empirical data from RacketStore deployment on 803 unique devices controlled by ASO workers and regular users, and insights from interviews with 13 participants. We develop a supervised classifier to detect apps installed only to be promoted, and an unsupervised classifier to differentiate between ASO-dedicated devices, organic worker devices, and devices used solely for personal activities

Chapter 6 introduces a fraud de-anonymization approach in online peer-opinion systems and presents two approaches to solve it. We first present an unconstrained optimization approach (UODA) built on top of a maximum likelihood formulation of the problem. Second, we introduce a graph based deep learning approach to predict ownership of account pairs by the same worker. We use such predictor to present disciminative fraud de-anonymization approach (DDA) and pseudonymous fraudster discovery (PFD) algorithm. Further and importantly, this chapter also introduces the first cheating-resistant protocol to conduct a live validation of fraud detection and de-anonymization techniques. Finally, in Chapter 7, we present the conclusions of the thesis.

CHAPTER 2

CONCEPTS AND BACKGROUND

In this chapter, we first describe the system and adversary model and then define the basic terminology used throughout the thesis. We discuss how peer-opinion systems including app markets are susceptible to abuse in the form of fake reputation.

2.1 System and Adversary Model

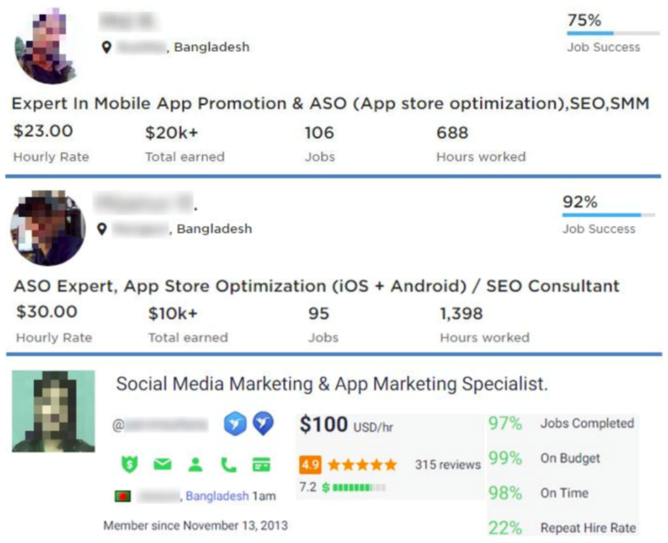
We consider online peer-opinion systems, e.g., Google Play, Google Maps, Yelp, Amazon, that host accounts for developers or product owners, users and products. Developers use their accounts to launch and publish products while users are expected to post reviews only for products they have used or bought. The survival of products in peer-opinion services is contingent upon their review influenced search rank. Higher ranked products are acquired more often and thus generate higher profits, via direct payments or ads. For example, a one star boost in rating was shown to help restaurants increase revenue by a 5-9% margin [LZ16]. While online systems keep their ranking algorithms secret for security reasons [SCM11], popular belief claims that the search rank of a product is at least linear on the number of its positive reviews and installs [Ank13].

App Search Optimization (ASO). Developers use their accounts to upload apps and users use their accounts to search for and install apps on their Android devices. Users can review apps after installing them. Google Play displays the account name and profile photo of the user along with the review, but not the link to the user's account. Users can register multiple accounts on their Android device, including multiple Google Mail accounts and accounts from other services, e.g., Facebook, Twitter. Users are then able to post reviews for an app, from all the accounts registered on the device where the app was installed: The Google Play Store application allows users to switch between accounts.

In this thesis we consider black hat app search optimization (ASO) efforts, where developers hire specialized, online *workers* to perform search optimization. ASO efforts include (1) installing the app on many devices, (2) performing *retention installs*, i.e., keeping the app installed for prolonged intervals, and (3) writing reviews with high rating values.

Fraud Origin. The pressure to succeed has created an underground economy for *search rank fraud*. Specialized ASO workers (also referred to as fraud workers, or fraudsters) control multiple user accounts and seek employment by product developers to post fake reviews or activities for their products. The accounts controlled by a fraud worker are also known as Sybils or sockpuppets [ZXL+18, KCLS17, AQAA+17, LGWM15, YWW+14, YGKX10, DM09, YKGF08]. Fraud workers advertise their services through crowdsourcing sites [Fiv, Upw, Fre], social networks (e.g., Facebook groups), and specialized fraud platforms [TSM16, RL16, AV16, AS16, AR16]. Moreover, fraudulent activities are profitable as evidenced by their price ranges. For instance, we identified 44 fraud workers in Facebook groups, Zeerk, Peopleperhour, Freelancer and Upwork that advertised prices ranging from a few cents ($0.56 on average from Zeerk.com) to several dollars per review (up to $10 in Freelancer.com) [RHCC18].

Facilitating Fraud. Crowdsourcing sites like *Fiverr*, *Upwork* and *Freelancer* [Fiv, Upw, Fre] host accounts for *workers* and *employers*. These crowdsourcing accounts have a unique identifier and require a linked bank account for depositing employer's escrow money or withdrawing worker's earnings. Workers on these sites bid on employer posted *jobs* while employers assign jobs to workers after successful negotiation. Thus, these crowdsourcing sites provide a comprehensive platform for


75%
Job Success

, Bangladesh

Expert In Mobile App Promotion & ASO (App store optimization),SEO,SMM

$23.00	$20k+	106	688
Hourly Rate	Total earned	Jobs	Hours worked

92%
Job Success

, Bangladesh

ASO Expert, App Store Optimization (iOS + Android) / SEO Consultant

$30.00	$10k+	95	1,398
Hourly Rate	Total earned	Jobs	Hours worked

Social Media Marketing & App Marketing Specialist.

@

$100 USD/hr

4.9 ★★★★★ 315 reviews

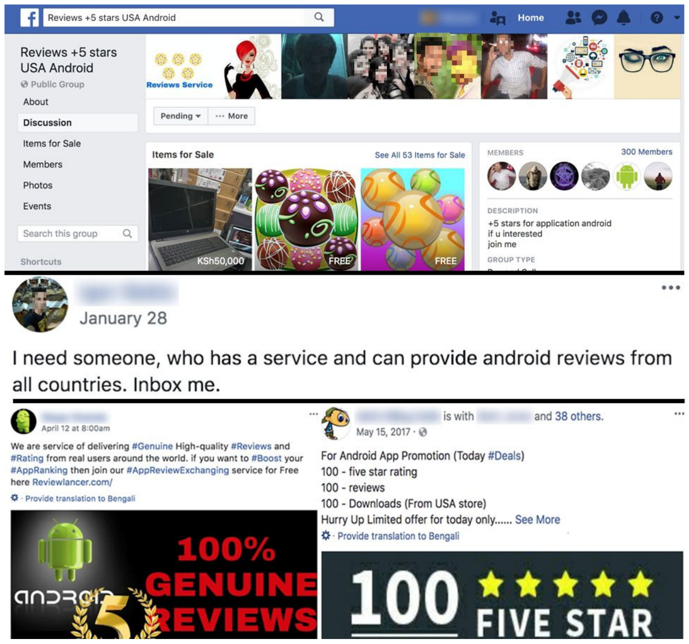
7.2 $

, Bangladesh 1am

Member since November 13, 2013

97%	Jobs Completed
99%	On Budget
98%	On Time
22%	Repeat Hire Rate

Figure 2.1: Anonymized snapshots of profiles of prolific ASO workers from Upwork and Freelancer.com. Workers control thousands of user accounts and earn thousands of dollars through hundreds of work hours.



+5 stars for application android
if u interested
join me

January 28

I need someone, who has a service and can provide android reviews from all countries. Inbox me.

April 12 at 8:00am

We are service of delivering #Genuine High-quality #Reviews and #Rating from real users around the world. if you want to #Boost your #AppRanking then join our #AppReviewExchanging service for Free here Reviewlancer.com/

Provide translation to Bengali

is with and 38 others.

May 15, 2017 ·

For Android App Promotion (Today #Deals)
100 - five star rating
100 - reviews
100 - Downloads (From USA store)
Hurry Up Limited offer for today only...... See More

Provide translation to Bengali

Figure 2.2: Anonymized screenshots of search rank fraud from Facebook. (Top) Page of Facebook group dedicated to search rank fraud. (Middle) Recruitment post from developer. (Bottom) Posts of fraud workers.

performing peer-opinion system fraud. Figure 2.1 shows how fraud workers earn thousand of dollars thru hundred of work hours in such sites.

For instance, we used Facebook's search tools to identify groups dedicated to the promotion of products through artificial endorsements. We concentrated on finding groups and pages that offer their service to post fake reviews on online marketplaces including Google Play and Amazon. We identified 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. Figure 2.1 shows the number of members for each group. While 7 groups were focusing particularly on reviews over Google Play, the rest 9 groups also promoted paid activities for other peer-opinion sites such as Google Maps, Yelp, Amazon, Facebook, TripAdvisor, and TrustPilot. As a consequence, social networks like Facebook provide high visibility to these services due to their large user base (see Figure 2.2 for sample snapshots). Moreover, fraud workers can also create their own service advertising pages hoping that developers discover them using keyword search on Internet search engines [Reva, MoP, Revb].

Effective fraud. In a separate Upwork data set experiment, we collected 161 search rank fraud jobs and their 533 bidding workers. We found that jobs assigned to a single worker occurred less frequently than jobs awarded to 2 workers. Furthermore, some developers assigned a single job to as many as 12 workers. We conjecture that this assignment distribution occurs due to the limited ability of a single worker to effect a significant impact over a subject's search rank. This observation reveals that subjects targeted by search rank fraud will usually receive fake reviews from multiple fraud workers.

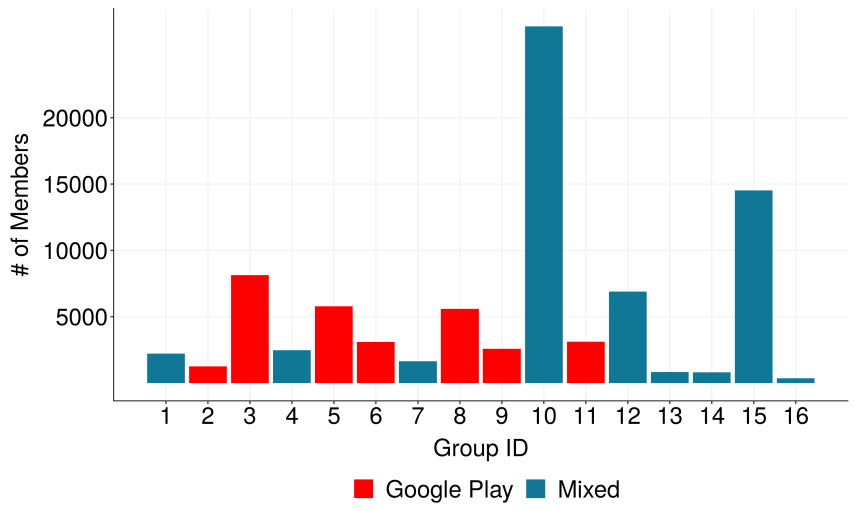


Figure 2.3: Number of group members for 16 fraud-centered Facebook groups we infiltrated. Red bars represent groups that exclusively promote Google Play apps. The blue bars are for groups that target an array of services, e.g., Yelp, Amazon, Google Maps, TrustPilot, and Justdial.

2.2 Basic Terminology

- **User**. A person or entity who posts reviews about a *subject* on an online peer-opinion system. Users make use of *user accounts* to establish their identity online.
- **Subject**. A *developer* created object or product that receives *user* created reviews on the peer-opinion system.
- **Developer**. A person or entity that hosts *subjects* on the peer-opinion system. Developers usually have incentives to maximize their subject's visibility via review manipulation for which they hire *workers*. Thus, we also refer to developers as *employers*.
- **Fraud or ASO worker**. A person or entity that performs review manipulation about a *subject* on behalf of a *developer*. Workers often use *Sybil accounts* to post fraudulent reviews on the peer-opinion system.

In our studies we have encountered and have recruited different types of ASO workers, which we briefly introduce in the following:

- **Regular workers**, which we call *workers* in the following, are the actual people who perform the ASO services, e.g., install the apps on devices that they can access, and write reviews or post ratings for the apps.
- **ASO Administrators** or *admins*, organize and coordinate communities of regular workers. Admins act as intermediaries between clients and regular workers. Admins can also act as regular workers, but their main duties include price and job detail negotiation with clients, identifying regular workers, forwarding them job details, verifying their work and paying them.
- **Organic workers** are a hybrid of regular users and workers, blending product promotion with personal activities.
- **Exchange workers** seek to magnify their ASO capabilities, by identifying peers to swap work: each peer helps the other with their job, e.g., by installing and writing reviews for the app promoted by the other peer.

Fraud detection and defenses. Online systems implement a suite of fraud detection and defense mechanisms [SCM11, YN18, NA16]. For Google Play, such observable mechanisms include:

- **Account validation**. Request users to prove control of a mobile phone, e.g., by providing its calling number, then retrieving a code sent to it through SMS.
- **Install-then-review**. Users can review an app only if they install it first [gplb].
- **Filter fake reviews**. Detect and remove reviews suspected of being fake.
- **Close fraudulent accounts**. Identify and close user and developer accounts suspected of behaviors that violate the site's terms of service.

CHAPTER 3

RELATED WORK

In this chapter, we present related work on de-anonymization and crowdturfing in online systems. First, we survey techniques used to de-anonymize users and to link identities across multiple sites. Second, we present previous work on the Sybil community detection problem in which an attacker creates multiple identities and pretends to be distinct users in the system. We then discuss previous approaches to deal with fraud and opinion spamming detection in online peer-opinion systems. We emphasize the differences between previous work and our novel approach to disincentivize fraud online. Finally, we study related work on underground markets and ethical data collection from fraud workers.

3.1 Author identification and cross-site identity linking

The *author identification* problem seeks to identify the original author of a document [NPG+12]. Narayanan et al. [NPG+12] used linguistic stylometry to perform large scale identification of blog post authors and argue damaging implications to anonymous bloggers and whistleblowers. Another closely related problem is that of *cross-site identity linking* attacks [AGL17, BBG+16, SYBT15, ZL13, JKJ13]. Adversaries were shown to be able to exploit linguistic [AT12] and location [GLP+13] patterns to link pseudonymous identities of the same user across different sites. Backes et al. [BBG+16] introduced relative and absolute linkability measures that rank identities by their anonymity, and used information about matching identities to estimate linkability risks. Andreou et al. [AGL17] further studied relationships between anonymity and risks of linkability of Facebook and Twitter accounts.

Venkatadri et al. [VGZ+16] leveraged this attack to develop a framework to transfer trust between sites and identify trustworthy accounts. Their key insight is that although users may be new on a particular site, most honest users would have long histories and established reputations on other sites they have been using before. Jain et al. [JKJ13] studied different identity search methods to link Twitter accounts to their respective Facebook accounts. These methods works mainly on the assumption that both online services share common attributes for the profiles that are created. Our work is different since crowdsourcing websites and peer-opinion systems (e.g. Google Play) are very dissimilar in nature. Also, instead of finding a one-to-one mapping, our research focuses on a many-to-one de-anonymization strategy that seeks to attribute many fake identities to a real identity (i.e. underlying fraud worker).

Zafarani et al. [ZL13] links accounts across different social networks by exploiting redundancy in username generation: individuals tend to select usernames that are generally not long, not random, and are repeated. They propose MOBIUS, a supervised learning approach that employs minimal information available on all social media sites to derive a large number of features that can be used to connect users across sites.

In the context of our work, de-anonymization is not an attack but a desirable feature. Our goal is to map fraudulent accounts in one site to the underlying worker (real identity) in another site that has substantial different functionalities. Our solutions go beyond stylometry to extract features that model the similarity of a wealth of activities (commonly reviewed products, times and ratings) between Sybil accounts, and introduce and leverage a new protocol to collect ground truth validation data.

3.2 Sybil Community Detection

Peer-opinion systems such as Google Play, Amazon, and Yelp are known to be particularly vulnerable to Sybil attacks. In a Sybil attack, a malicious user creates multiple fake identities and pretends to be multiple, distinct users in the system. The pseudonymous fraudster discovery problem that we present in this work is equivalent to uncovering Sybil (or sockpuppet) communities. Sybil accounts disconnect physical from online identities, thus have a suite of malicious uses, that include gaining control over systems [Dou02], vandalism [SHM13], or creating the illusion of widespread support of ideas, people and products [SR07]. Early Sybil detection work in online systems has focused on social networks [YGKX10, YKGF08, TMLS09, DM09], and made the assumption that attackers can easily form social relationships between Sybil accounts they control, but find it hard to establish links to honest accounts. For instance, Yu et al. [YKGF08] present SybilGuard, a protocol that leverages the existing human-established trust relationships among users to bound the number and size of sybil groups. Their assumption is that malicious users can create many identities but few trust relationships. Therefore, there are weak links between honest and sybil users. Danezis et al. [DM09] propose SybilInfer, a Bayesian inference approach to Sybil detection that relies on a probabilistic model of honest social networks. Their inference mechanism outputs regions of dishonest nodes with an assigned probability that indicates its degree of certainty.

Tran et al. [TMLS09] present SumUp, a vote aggregation system that leverages the trust network among users to defend against Sybil attacks. They use adaptive vote flow aggregation to limit the number of synthetic votes by adversaries to no more than the number of atack edges in the trust network. However, Yang et al. [YWW^{+}14] showed that in Renren, Sybil accounts do not form dense communities, and are well connected with honest users. Using link creation timestamps, they observe that most links between Sybil accounts are created accidentally. This shows that Sybil defenses such as the ones described above are unlikely to succeed. In peer-opinion systems that lack strong social links between user accounts, social graphs can be replaced by *co-activity graphs*, such as our co-review graphs. Then, in discussion communities, Kumar et al. [KCLS17] showed that Sybil accounts still differ from honest accounts through social network structure, posting behavior and linguistic traits. They leveraged the discovery that pairs of accounts controlled by the same individual are more likely to interact on the same discussion, to build a co-ownership predictor. They found that Sybil accounts tend to star fewer discussions, write shorter posts, use more personal pronouns such as "I", and have more clustered ego-networks.

Zheng et al. [ZXL$^+$18] predict Sybil links between user accounts based on the similarity of their reviews, in terms of the products targeted, times and ratings. They employ the Louvain method to detect communities on the generated Sybil graphs, then classify each community as benign or Sybil using supervised learning.

3.3 Fraud and Opinion Spamming Detection

There is a large body of research on defending against online system fraud. State of the art approaches use inference on the social graph [WGF17, RA15, ACF13, KCS18, PCWF07] and classical machine learning based on several assumptions. These assumptions include: (i) bursty activity [LFW$^+$17, YKA16, FML$^+$13, LCM$^+$15], (ii) review plagiarism [KCS18, KCA17, HTS16, MKL$^+$13] and distinguishability of machine vs. human generated reviews [YVC$^+$17], (iii) extreme reviews and deviation [RA15, XZ14, MKL$^+$13, WXLY11], (iv) lockstep behavior [BXG$^+$13, SMJ$^+$15, TZX$^+$15], and (v) ratio of singleton accounts [YKA16, RA15, SE15].

The seminal work by Jindal and Liu [JL08] introduced the opinion spamming problem and showed that it is different from web and email spam. Based on the analysis of 5.8 million reviews in Amazon, they found that opinion spam is widespread and proposed a logistic regression algorithm that runs on reviewer and review centric features. Wang et al. [WXLY11] model the opinion spam problem using a tripartite graph of reviewers, reviews, and stores; and proposed an iterative method utilizing influences among objects in these three sets.

Akoglu et al. [ACF13] propose FraudEagle to detect both fake reviews and reviewers in online peer-opinion sites. This approach models the system as a bipartite graph of products and users, and infer node labels via inference with the Loopy Belief Propagation (LBP) algorithm. It operates in an unsupervised and semi-supervised fashion requiring no previous labeled data, or incorporating such information if available. To score users, they assume that (1) honest users mostly post positive reviews to good products and negative ones to bad products, (2) honest users could also rate negatively good products and might like bad products based on their preferences, (3) fraudsters tend to write positive reviews to bad products and negative reviews to good products, and (4) fraudster could also mimic honest users to camouflage their activities. Similarly, Wang et al. [WGF17] use Loopy Belief Propagation on the directed social graph and estimate the posterior probability for each user which is then used to predict label. Since LBP is not guaranteed to converge, they propose GANG and derive conditions under which this algorithm converges.

Kaghazgaran et al. [KCS18] propose TwoFace, a system to uncover crowdsourced review manipulators who target Amazon products. TwoFace first sample actual evidence of fraud by exploiting crowdsourcing platforms. Based on this initial source of ground truth information, they then propagate suspiciousness of these seed users to identify similar users via random walks. Finally, they map users into a low-dimensional embedding space that captures the structure of the community and were able to uncover distant users who serve structurally similar roles.

Li et al. [LFW$^+$17] noticed that reviewers' posting rates in Dianping follow a bimodal distribution and exploit this discovery to introduce a two-mode labeled Hidden Markov Model to model spamming using only individual reviewers' posting times. They then extend it to the Coupled Hidden Markov Model to capture reviewer posting behaviors and co-bursting signals. Mukherjee et al. [MKL$^+$13] model spamcity as a latent variable. Their approach works in a Bayesian fashion and is built on the intuition that spammers have different behavioral distributions than non-spammers.

Xie and Zhu [XZ14] study opinion spamming on Apple's China App Store and present GroupTie, a graph-based approach to detect collusion of reviewers. To this end, they build a weighted undirected graph and detect collusion by applying graph clustering. Their key insight is that since members of a hidden collusion group have to work together and their ratings deviate from the app's average rating, collusive actions will be more evident. Xie and Zhu [XZ15a] also study the underground market of mobile app review promotion. They build AppWatcher, an automatic data collection system to obtain ground truth data on app promotion and monitor 52 paid review service providers. They also propose an app tracer to narrow down the list of promoted apps in this underground market.

Yang et al. [YHZ$^+$12] have shown that "criminal" Twitter accounts tend to be socially connected, forming a small-world network. This is confirmed by the work of Mukherjee et al. [MLW$^+$11, MLG12, MKL$^+$13], who introduced features that identify reviewer groups, who review many products in common, post their reviews in bursts, and are among the first to review the product. Yang et al. [YHZ$^+$12] further proposed a criminal account inference algorithm that detects new criminal accounts by propagating malicious scores from a seed set of known criminal accounts to their followers according to the closeness of social relationships and the strength of semantic coordination.

Fake endorsement has also been investigated on Facebook. Beutel et al. [BXG$^+$13] propose CopyCatch to detect lockstep Page Like patterns on Facebook by analyzing the graph between users and pages and the times at which the Likes were created. CopyCatch is based on one-class and subspace clustering and is built on the intuition that spammers control a few accounts that they use to like many pages. Satya et al. [BSLL$^+$16] investigate the problem of fake likers on Facebook by first collecting information from honeypot pages and then contrasting fraudulent with legit users via supervised learning algorithms on extracted features. Cao et al. [CYYP14] also exploit lockstep behaviors and use insider information, to cluster user accounts according to the similarity of their actions, and uncover groups of suspicious accounts in Facebook.

There is also significant work on different types of fraud and abuse in online systems. For example, Stringhini et al. [SMJ$^+$15] present EvilCohort, a service-agnostic approach to detect accounts on online systems that are controlled by cybercriminals. EvilCohort works by identifying communities of online accounts that are all accessed from a number of shared connection points. Specifically, it builds a bipartite graph between online accounts and IP addresses from which they build a projected graph representation where vertices are accounts and the edge weights represent the number of shared IP addresses. Graph clustering is then used to detect communities of online accounts that are accessed by the same IP addresses. Tian et al. [TZX+15] analyze crowd fraud spamming for internet advertising (fraud clicks), define a synchronization similarity between click histories and transform the problem into a non-parametric clustering problem. Li et al. [LMC+16] extend lockstep behavior identification with semi-supervised learning based on local spectral graph diffusion, to detect YouTube fraud. Nilizadeh et al. [NAG+19] proposed OneReview, a method for locating fraudulent reviews, correlating data from multiple review sites, and assuming that the reputation of a product should be similar in several crowd-sourced websites.

Unlike most of this work, that has focused on providing binary classification of reviews as fake or honest, and accounts as fraudulent or benign, we seek to identify the prolific workers responsible for significant fraud. We implement a maximum likelihood estimation and deep learning based process to expand seed, worker-controlled accounts, and assign them to the crowdsourcing account of the fraudster who controls them.

3.4 Fraud Data Collection and Underground Markets

Collecting fraud ground truth data is a notoriously hard task. Previous work has gathered such information by hiring spammers to target synthetic honeypot products on different online services. For instance, De Cristofaro et al. [DCFJ+14] deployed 13 Facebook honeypot pages and promoted them using legitimate Facebook Ads and 4 popular *like farms.* They monitored the "liking" activity on these pages every two hours and then analyzed the differences between the two distributions based on demographic, temporal and social dimensions. Some farms seemed to be operated by bots while others mimic regular users' behaviors.

In a similar way, Stringhini et al. [SWE$^+$13] studied Twitter follower markets by purchasing followers from different merchants and used such ground truth to discover patterns and detect "market" accounts in the wild. To locate these markets, the researchers use three different methods: identifying suspicious clusters of Twitter accounts, searching for advertisement tweets, and querying search engines for terms such as "twitter followers", "buy services", etc.

Thomas et al. [TGSP11] identified over 1.1 million accounts suspended by Twitter for violating the terms of service. Accounts are suspended for (1) frequent request to befriend users in short periods of time, (2) reposting duplicate content across many accounts, (3) posting only URLs, and (4) posting misleading content to trending topics. In the process, they collected 1.8 billion tweets, 80 million of which belong to spammers. They characterized the behavior and lifetime of spam accounts, their campaigns, and the wide-spread abuse of legitimate web servers. Thomas et al. [TMG$^+$13] also investigated the market for fraudulent Twitter account to monitor prices, availability, and fraud by 27 merchants over 10 months. After placing 144 orders, they could buy a total of 120,019 accounts from merchants that operate their own websites, are active on blackhat forums, and work on freelancing websites such as Fiverr and Freelancer. They have further developed a classifier based on registration signals that detects millions of fraudulent accounts that merchants sold.

Springborn et al. [SB13] studied *pay-per-view* networks in the context of online fraud advertising where invalid traffic generation aims to inflate ads impressions on websites. To this end, they purchased traffic for a set of honeypot websites and analyze the mechanisms used for impression fraud by *pay-per-view* networks. Their results showed that these networks deliver hundreds of millions of fraudulent impressions per day.

Similarly, Park et al. [PJM$^+$14] built an automated data collection system to study fake payment scams targeting users on Craiglist. To do this, they created 1,376 *magnetic honeypot* advertisements that would selectively attract scammers but not legitimate users. Additionally, they developed an automatic conversation engine that performs linguistic analysis of emails from scammers, and engages in communication with them. From their analysis, they found that around 10 groups of scammers were responsible for nearly half of the over 13,000 total scams attempt received. These groups used shipping addresses and phone numbers from Nigeria and the U.S.

Portnoff et al. [PAD$^+$17] proposed an automated approach based on machine learning and natural language processing for analyzing underground forums and better understand cybercrime. Such tools allow an analysis to determine the nature of the post, the product being offered, and its price. Their approach achieved over 80% accuracy after being tested on posts from 8 distinct underground forums.

We also found studies on account theft or *account hijacking.* For example, Bursztein et al. [BBM$^+$14] explored manual hijacking of Google accounts from incidents that occurred between 2011 and 2014. Using 14 proprietary data sets, they linked manual hijacking with phishing and found that phishing requests target victims' email and banking institutions accounts, as well as their app stores and social network credentials.

Mirian et al. [MDS$^+$19] studied *hack-for-hire* services to understand the playbook that attackers use to gain access to victim email accounts. Posing as buyers, they interacted with 27 underground market services, only five of which succeeded in attacking synthetic identities that the authors controlled. They found that despite the ability to successfully deliver account access, the market exhibited low volume, bad customer service, and had multiple scammers. Perhaps unsurprisingly, attackers primarily relied on tailored phishing messages that bypass SMS two-factor authentication. McCoy et al . [MPG$^+$12] studied the business model of *online pharma*, using ground truth data sets of transaction logs that were leaked and found on underground forums and file-sharing sites.

Critical operational details of the fraud market have remained however mostly unstudied. This thesis seeks to address this, by both documenting and validating operational procedures of ASO workers who target Google Play. Unlike previous work, we conduct an interview study to directly engage and seek insights from fraud perpetrators, then support them through an analysis of empirical fraud data. In our fraud study, we seek to also identify (1) Google Play vulnerabilities that fraud workers found and exploit, (2) evolution in fraudulent behaviors to avoid detection, and (3) their intrinsic weaknesses, to be exploited by the next generation fraud detection solutions. In chapter 4, we provide self-reported insights from studied ASO workers, that confirm the existence of organic workers in the wild. Organic workers attempt to mimic the behavior of real users and use their personal devices to post fraud.

In chapter 5, we confirm the rise of organic workers and show that they are in fact regular Android users who use their personal devices and accounts to occasionally post paid reviews. We collect data for the device and app usage from such workers and conduct interviews with 5 of them. We further report a new type of *exchange* workers who seek to magnify their fraud posting capacity without payment. 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 devise device classification solutions.

Identifying organic and exchange workers is difficult, since their behaviors evade the above detection solutions. For instance, organic workers use their personal devices and accounts to mix paid reviews among their everyday activities, and operate independently thus are less likely to have synchronized behaviors. Further, exchange workers establish ad hoc p2p-like relations to exchange reviews for the products they promote. The spontaneous nature of exchange relations also suggests the absence of detectable lockstep behaviors. Instead, in this thesis we collect ground truth device usage information, and reveal that features extracted from app and device usage can detect and differentiate between devices of organic workers and those used exclusively to promote products.

Relevant work also includes efforts to automatically generate reviews and detect such reviews [YVC+17, RJS17]. For instance, Yao et al. [YVC+17] identify a new class of attacks that leverage deep learning language models to automate the generation of fake reviews, and develop novel automated defenses against these attacks. Radford et al. [RJS17] used representation learning to generate reviews given only a sentiment (positive vs. negative). In this thesis we focus on the orthogonal investigation of device and app usage by workers and regular users. It is conceivable that our worker detection and classification solutions would be used in conjunction with techniques to detect DNN-generated reviews [YVC+17].

CHAPTER 4

THE ART AND CRAFT OF FRAUDULENT APP PROMOTION IN GOOGLE PLAY

4.1 Introduction

Popular online services that host products, news, social relationships and peer-opinions, are the targets of fraudulent behaviors, that skew public opinion and bias product reputation and popularity [Ako18, Rei17b, WF19, Kna19, CDHH18, Ste19, HGT+17]. To reduce the effects of such behaviors, commercial peer-opinion sites employ proprietary solutions to detect and filter fraud, e.g., [Cip16, YN18, Jan18, Bra18, air18, Per16, MVLG13, SCM11, KS18, NLS+17]. Similarly, a substantial body of academic research has focused on the detection aspect of the fraud problem, and has proposed and used assumptions about the behaviors and capabilities of fraudsters, that are based on intuition, extracted from small datasets of fraud, or revealed by collaborators within commercial sites. While such previous efforts have revealed important insights into the operations of fraudsters, most have not been validated with empirical feedback from the actual perpetrators.

In an effort to address this limitation, we first performed a structured interview study comprised of 118 questions, with 18 Black Hat App Search Optimization (ASO) workers that we recruited from 5 freelancing sites, concerning fraud that they post on Google Play [Rah18, RHR+19]. Second, we performed a quantitative investigation with data that we collected from 39 other ASO workers recruited from the same sites. The data includes 1,164 Google Play accounts that the 39 ASO workers revealed to control, and 21,767 fake reviews posted from these accounts for 6,362 unique apps. Further, we identified, and report a Google site bug that enabled us to infer the device models used to post 198,466,139 reviews for the 6,362 apps.

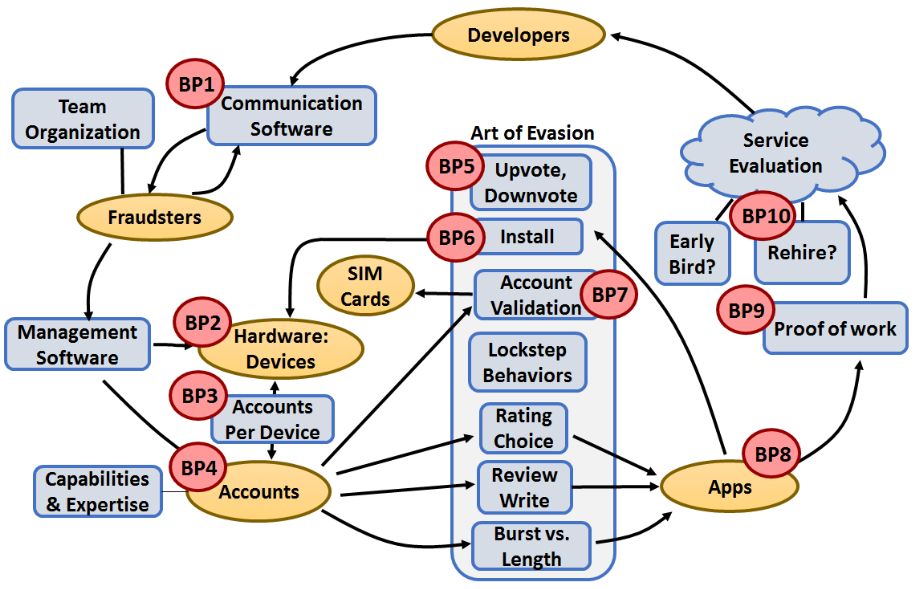


Figure 4.1: Map of discovered fraud workflow in Google Play. Orange ovals denote tangible participants and assets, blue rectangles denote several investigated capabilities, behaviors or strategies. Small red ovals represent fraud vulnerability points that we identified and discuss in § 4.4.

Based on the findings of our studies, we present the fraud workflow map of Figure 4.1, showing newly identified and previously explored fraud capabilities, behaviors and detection avoidance strategies. Specifically, we report multiple, novel insights into the working patterns of ASO workers, including that they (1) pool in physical, brick-and-mortar offices, friends-and-family organizations, and online teams, (2) have either a well-articulated role and are salaried on a regular basis, or are part of unstructured teams and share earnings, (3) have access to many user accounts, of both sockpuppet (fake) and organic (controlled by real users) types, (4) have access to large and diverse stocks of low to high-end, and new to old mobile device models, (5) flexibly outsource work when their number of accounts or device models are insufficient, and (6) implement interactive work verifications.

Further, our studies provide evidence that supports several observations and assumptions made by previous fraud detection work, about, e.g., the emergence of organic fraud [KCS18, KAC19, ZXL$^+$18], the timing of fraud [FML$^+$13, YKA16, LNJ+10, MKL+13, MLG12, Xu13, KM16], the fake review writing process [MKL+13, FML+13, Xu13, HTS16, LNJ+10, SE15, MVLG13, XZ15b, KCS18, LCNK17, RA15, YA15, MLG12, KCA17] and the choice of ratings [MKL+13, ACF13, KCA17, MVLG13, KCS18, RA15, MLG12, XZ14, XZ15a].

However, we also report and validate concrete, participant-revealed behaviors that do not fit the mold of assumptions made in previous work, including lock-step behaviors [SMJ+15, TZX+15, YKA16, XZ15b, JCB+14, CYYP14, SLK15, XZ14, XZ15a, LFW+17] or posting reviews in bursts [FML+13, HTS16, LFW+17, HSB+16a, BSLL+16, XZLW16, Xu13, GGF14, BXG+13, KCA17, LNJ+10, MVLG13, MKL+13, KCS18, LCNK17, YKA16, FLCS15, DCFJ+14, XWLY12, CYYP14].

We also found and report participant-claimed techniques to bypass Google-imposed verifications, e.g., user account validations and review-posting sanity checks, and even strategies to leverage Google's fraud detection mechanisms to improve fraud efficacy, e.g., downvoting negative reviews to trigger their removal, or using singleton accounts that exploit detection cold-start problems.

Finally, and importantly, we identify several vulnerability points in the fraud workflow, and propose defenses that exploit them. In summary, we introduce the following contributions:

- **ASO worker studies**. Present empirical data from actual ASO workers, to advance our understanding of their work, through interviews and a quantitative analysis of gold standard fraud data [§ 5.3].
- **ASO worker capabilities, behaviors and strategies**. Report new findings on the capabilities and behaviors exhibited by ASO workers [§ 4.3]. Provide evidence that supports several observations and assumptions made by previous detection work. Report and validate concrete strategies to avoid detection, in-

cluding departures from existing assumptions. Build a first map of the Google Play fraud workflow.

- **Google Play vulnerabilities**. Identify and report a bug that can be exploited to collect device model information from reviews [§ 4.2.2]. Report Google Play verifications claimed to be ineffective by participants. [§ 4.3].
- **Impacts**. Identify vulnerability points in the fraud workflow and discuss their potential to advance fraud detection and prevention work [§ 4.4].

4.2 Methods

Our study involves both a qualitative exploration of and a quantitative investigation into various aspects of fraud production. In this section we describe both studies.

4.2.1 Qualitative Study

The qualitative study of our work is comprised of in-depth interviews with 18 ASO workers. We recruited participants from several Facebook ASO groups, and also Upwork [Upw], Fiverr [Fiv], Zeerk [Zee], and Peopleperhour [Peo], all popular among ASO workers. We identified 560 such workers, and invited them to participate in our study through the 1-on-1 communication services of the corresponding sites. We include the recruitment message in the auxiliary document.

72 of them responded to our invitation. To select participants who are actively involved in ASO jobs, we asked the responders, 3 questions, all for Google Play: (1) "how many accounts do you control?", (2) "for how long have you been actively doing ASO?", and (3) "on how many ASO jobs did you work, approximately?".

We identified 25 participants who control at least 100 accounts on Google Play, have been active for at least 1 year, and have completed at least 100 ASO tasks. Following recruitment, and before starting the interview, we read to these participants the introductory script included in the auxiliary document. 18 of them (all male, 19-29 years old, located in Bangladesh(13), India(4) and New Zealand(1)) agreed to participate.

In the following, we refer to the interview participants as P1, .., P18. With these participants, we conducted a structured interview study that had 46 questions, with additional 72 questions for clarifications, see section 7. The questions range from demographic information to workflow, and from the devices used to the operational methods employed. We conducted the interviews over Skype, between August and October, 2018. Interviews lasted from 33 to 66 minutes (M = 46.38, SD = 12.34). We paid a rate of 5 USD for every 15 minutes a participant spent in our interview. We audio recorded the interviews with the participant permission, then transcribed and anonymized the data.

We analyzed the anonymized data using the Grounded Theory method [CB07]. We used open coding to identify 169 unique codes, including both abstract and concrete labels. Two members of our team independently coded the data. The inter-coder agreement was 84.61%. In the cases where codes of the two coders did not match, a discussion was held with a third member of our team, to decide the final code. We used axial coding to relate the generated codes, and ended up with 22 categories grounded in the collected data. Some of the categories are: account blending, account creation, devices, early-bird fraud, extreme reviews, strategy, etc. We have then further refined our categories into the codes that form subsection titles in § 4.3.

4.2.2 Quantitative Investigation

We performed a quantitative investigation with user accounts collected from 39 ASO workers, different from the qualitative study participants, but recruited using the same methods described in $ 4.2.1. In the following, we refer to the quantitative study participants as F1, .., F39. Each of the selected workers claimed to control up to 500 Google Play accounts ($M = 211, SD = 166$), and each shared the IDs of at least 15 Google Play accounts that they control. This yielded a total of 1,164 account IDs for analysis.

We then crawled the 6,362 unique apps that the ASO workers reviewed using those IDs, and that were available in Google Play. These apps had received 21,767 reviews from the 1,164 worker-controlled accounts, and a total of 218,167,727 reviews. We used the AppBrain API [App] to collect the category and release date of each app.

Device model data collection. We have collected information provided by Google Play about the devices used to post fraudulent reviews. Google Play's client-side enforced functionality, allows an authenticated user to filter reviews according to the model of her registered devices. We used this functionality to query the reviews posted for an app, for all possible device models, and thus identify the device model used to post any individual review. We used the list of 21,597 Google supported devices [GP], that contains the parameters that we needed to identify the device models used to post the above 21,767 reviews, posted from the 1,164 ASO worker-controlled accounts, as perceived by Google's systems. In addition, we collected the device release date and price (in EUR) from GSM Arena [gsm] and Gadgets360 [gad].

4.2.3 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 [BRBMR17]. Our study had a very clear declaration of the researchers' identity, research objective, and potential impact on the participants' work without following any sort of deception. The whole study procedure was scrutinized and approved by the institutional review board of a major North American university (IRB-18-0077@FIU). We include our recruitment message and introductory script in Appendix 7. We include a discussion of the process of our recruitment, the possible reasons for our participants to respond, and other relevant issues, in the auxiliary document.

We used GDPR [Par16] recommended pseudonymisation for data processing and statistics, and other generally accepted good practices for privacy preservation. We were very careful to hide participant identity and to avoid risks of being misunderstood by them. We also explained to them very clearly any risks that their job may have through our research. After data collection, we have deleted all device-to-identity links and only generated statistics that allowed us to validate our assumptions. We have avoided obtaining additional information about the devices used or the accounts involved. We have contacted Google about our discovered device model identification issue, through Google's vulnerability reward program (VRP) [GVR] (issue: 119676181). Google has accepted our finding and has invited us to join their hall of fame.

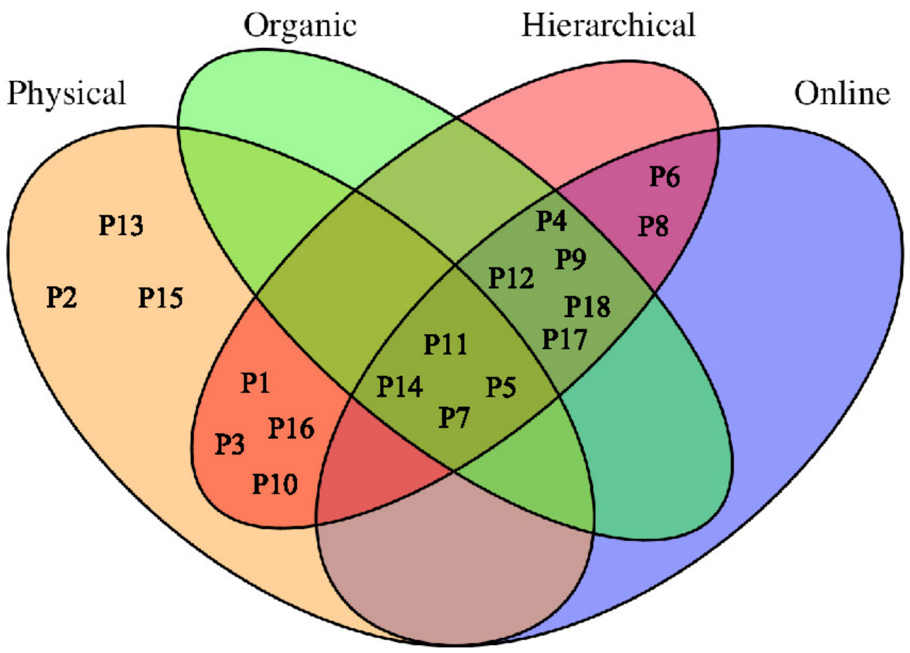


Figure 4.2: Venn diagram of participant categories, reveals diversity and complexity of fraud organizations. Participants are part of teams that are either (1) physically co-located or online, (2) hierarchical or flat, and (3) sockpuppet account based or organic.

4.3 Findings

We organize, analyze and report findings from the interview and quantitative studies. Figure 4.1 provides a map of the topics that we investigated.

4.3.1 Team, Location, and Organization

All the 18 interview participants claimed to be part of organizations dedicated to posting fraud in Google Play. Our data shows that ASO workers assemble in various organizational structures. While some of them work in a team where each person has a well-articulated role and they are salaried on a regular basis, many of them work in a more unstructured team and the whole team share their earnings. We classify ASO teams into several categories, based on their location, organization type, the type of fraud, and profit sharing structure. Figure 4.2 shows the Venn diagram of the 18 participants grouped according to 4 of these categories, for readability.

Team size. The first column of Table 4.1 lists the team sizes claimed by each participant for their organization, including both physically co-located and online team members. 5 participants claimed to work alone. The other 13 participants claimed to have a team with at least 10 members. Notably, P4 claimed to be part of a big company with around 150 people in their team, who organize 15,000 organic ASO workers through virtual (WhatsApp, Facebook) groups.

Physical co-located vs. online teams. Seven participants (Figure 4.2) claimed to work with a physically co-located team. 5 of them claimed to have brick and mortar offices. Figure 4.3 shows a photo taken by P10, with the premises and employees of his fraud team. 7 others claimed to have strictly online teams. The remaining 4 claimed to be a part of hybrid organizations that (1) are a physical team, including working alone with their own devices and accounts, and (2) have access to online ASO workers. Notably, P18 said (1) *"I run a mobile repair shop. I use the devices that I get to repair."* and (2) *"I share the link in my group and they review it."* P11 said *" I use two types of accounts, my friends and family, and my own 100 accounts."*

P14 claimed to be part of a flat team of 4 physically co-located members, and manage 30 online team members. P5 claimed to have a team of 12 and a Facebook group with organic users, while P7 has a team of 50 people and also runs campaigns to recruit more reviewers. P18 has access to physical devices and also a Facebook group.

Organization structure: hierarchical vs. flat. 15 participants claimed a hierarchical structure of their organizations (Figure 4.2). 11 of them described specific roles in their organizations, that include *job managers*, who interface with the de-

Figure 4.3: Photo taken and volunteered by participant P10, with the premises and (anonymized) employees of his business. Photo reproduced with permission from the participant.

velopers and manage work from the marketplace, *team admins*, who organize, distribute tasks, and verify the work of *review posters*, and *new account creators*. For instance, P3 said *"I am one of the admins in our team and we have 10–12 admins. Under each admin, we have 15–20 members. All admins work as subcontractors, and some of our other team members work with the developers and manage work from the marketplace."* However, 2 participants claimed to work in teams with a flat organization. For instance, P15 said *"We all work together. There is no hierarchy."*

Organic fraud. 9 participants claimed to organize or be part of online teams of "organic" users, workers who use their personal accounts to post fake reviews (Table 4.1). P5 said *"I also have my own Facebook group where I have combined 60 real users to write reviews."* P7 did not specify the number of organic accounts that they can access, but stated *"we have 3,000 accounts. If we need more we run CPI/CPA campaign where people get an incentive to install apps."*

Profit sharing. One participant claimed to pay team members a monthly salary, while another one claimed an even split among members. Three of them mentioned preferential cuts for the job manager (10–25%) and team lead (10–50%) and equal split of the rest among the actual review posters. Two participants claimed a flat rate for the review posters ($0.40 per review). The rest of the participants did not respond to this question.

Summary. Our study thus confirms observations made by existing work, that fraud is perpetrated by experts who control either (1) many sockpuppet user accounts, e.g., [YA15, BXG$^+$13, MKL$^+$13, SLK15, XZLW16, LCNK17, MLG12, XZ14, LFW$^+$17, FLCS15, KCLS17] or (2) *organic fraudsters*, i.e., real account owners recruited online [mic12, rap]. Our study also provides concrete numbers and extends the existing literature by adding that (1) ASO workers can be hybrid (e.g., both organic and sockpuppet masters) and (2) product developers can hire multiple types of expert ASO workers to promote their products.

Participants claimed to charge between $0.5 and up to $6 per posted review (M = 2.16, SD = 1.86), and to have between 1 and 6 years of experience in ASO jobs (M = 3.03, SD = 1.53). During this time, they claimed to have worked on between 150 and 4,000 apps in total, and between 6 and 50–60 apps in the past month (M = 34.11, SD = 18.37). They also declared a diverse educational background, including 2 masters degrees, 11 completed bachelor degrees, 2 ongoing bachelors, and 4 high school graduates.

4.3.2 Fraud Capabilities and Expertise

The middle columns of Table 4.1 list the number of user accounts claimed to be controlled by or accessible to each of the 18 participants. Most participants control a few hundred accounts, however, 8 participants can access several thousands: P13 claimed to be part of a team of 13 workers who control 80,000 accounts.

		Accounts		Devices		
P	**Members**	**Organic**	**Inorganic**	**Mobile**	**Laptop**	**Online**
P1	40	0	15,000	300	0	0
P2	12	0	300	40	0	0
P3	195	0	1,500	200	0	0
P4	150	15,000	0	0	0	15,000
P5	12	100	0	0	0	60
P6	1	0	1,500	0	0	500
P7	50	N/A	3,000	1,000	0	0
P8	35	0	150	0	0	100
P9	15	400	0	0	0	450
P10	30	0	450	30	35	0
P11	1	200	100	45	0	200
P12	1	500	0	0	0	500
P13	13	0	80,000	13	13	0
P14	34	5,000	0	0	0	5,000
P15	10	0	300	50	0	0
P16	50	0	500	70	0	0
P17	1	1,000	0	0	0	1,000
P18	1	500	30	30	0	500

Table 4.1: Number of team members, and of accounts and devices claimed by the 18 interview participants.

7 participants, each claiming to control thousands of accounts, also claimed to be able to write an "unlimited" number of reviews for an app, i.e., more reviews than the developer can ask or afford (as inferred from the participant's past experience). The other 11 participants, with up to 3,000 accounts, claimed to be able to write a number of reviews that was consistent (i.e., smaller or equal) to the number of accounts they claimed to control.

To provide perspective on several of these claims, Figure 4.4 shows the number of accounts revealed, and the number of unique apps reviewed from those accounts, by each of the participants in our quantitative study (§ 4.2.2). In total, we have crawled information from 1,164 accounts and the 6,362 unique apps that were reviewed from these accounts. Even in this limited gold standard dataset, one participant (F18)

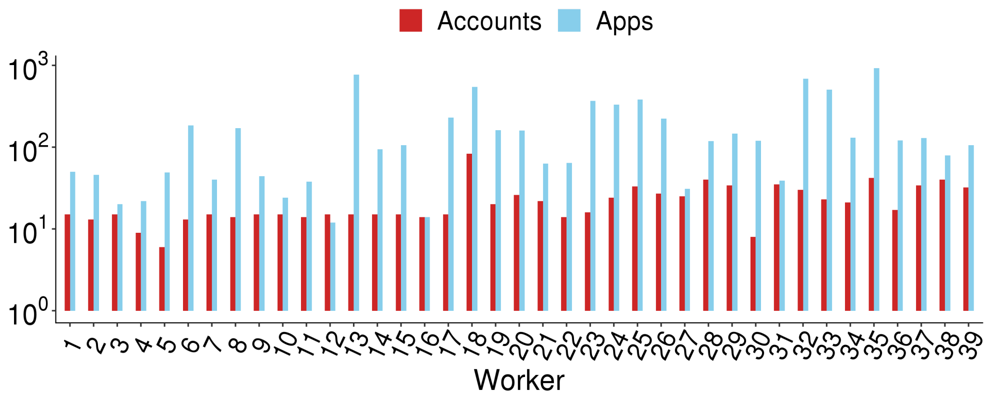


Figure 4.4: Number of accounts revealed by F1,..,F39 and number of apps reviewed from them. F18 revealed 83 accounts. 14 workers have reviewed at least 150 apps from the revealed accounts. F35 has reviewed 927 apps!

was able to reveal 83 accounts that he controls, and F35 has reviewed 927 unique apps from his 42 accounts.

4.3.3 Hardware: Devices

All the interview participants claimed to own or have access to multiple mobile devices. The last columns of Table 4.1 list the number of devices, organized by types, claimed to be controlled or accessible by each participant. 9 participants claimed to post fraud from mobile devices; 11 participants claimed this also happens from the mobile devices of organic ASO workers that they control. 2 participants said that they also post from emulators running in laptops, e.g., P13 claims to have 13 laptops and use the BlueStacks emulator [Blu] to install and review apps, and also 13 smartphones.

P8 and P18 have an almost 1-to-1 account-to-device mapping. Participants such as P2, P3, P7, P10 and P15, have a small but many-to-one mapping, e.g., up to 7 accounts per device. Others, such as P1 and P13, claim to have significantly more accounts than devices (e.g., 15,000:300 and 80,000:30 respectively).

Mobile device models. Several participants claim access to communities of organic users (see Figure 4.2), thus to a diverse set of devices. 4 participants (P1, P10, P11, P13) claimed to own only low-end, cheap devices. Others (P7, P15, P16) claimed to own a mix of low, medium and high-end devices, dominated by low-end devices. For instance, P7, who claimed to own more than 1,000 devices said that (1) *"we try to choose cheap devices with more features and memory,"* however (2) *"we also have high-end phones like Nokia, Samsung, which we need to review virtual/augmented reality apps"*.

Device source. Most participants claimed to purchase their devices on the regular market. However, P11 said, about his claimed 45 devices, that *"I have bought them from the black market with a very low price."* Further, as mentioned in § 4.3.1, P18 claimed to run a mobile device repair shop, and use the devices he is supposed to repair, to write reviews.

Device storage. 6 participants claimed to store the devices on a table, readily accessible. P1 claimed to store the devices in a separate room. P7 said that *"the department who handle reviews and installs is on a different floor, and high-end phones are kept in the locker after use for safety."* We also asked P7 about how they manage to charge 1,000 devices. He mentioned that they have a dedicated team to manage all the devices, and charge a device every 2–3 days. Further, he claimed that they keep the devices on during office time, and turn them off after 11pm–midnight.

App-device compatibility issues. 9 participants (P5, P6, P8, P10, P11, P13, P14, P16, P17) said that they have not had compatibility problems. However, P7 said that he runs campaigns to recruit ASO workers who have compatible devices, or even purchase such devices. P9 and P15 said that they provide as many reviews as they can from their compatible devices, and contact the developer to explain the

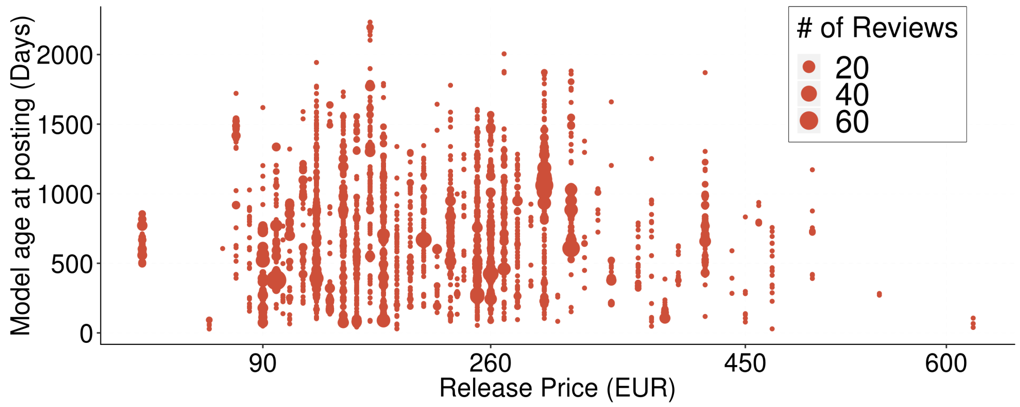


Figure 4.5: Scatter plot of device release price (EUR) vs. model age (Days) at posting time, for each of 9,942 reviews posted from 344 unique device types. Most devices are old and low-end (45.98%) or mid-end (31.41%), or fresh and low-end (15.31%). High-end and even free devices have been used!

problem.

Quantitative Investigation. We used the technique described in § 4.2.2 to find 344 unique device models, used to post 9,942 of the 21,767 reviews written from the accounts controlled by the 39 participants. We found that 12 participants posted reviews from at least 20 different device models; F35 used at least 84 distinct device models. However, participants F9 (215 reviews), F10 (166), F14 (162), F16 (67), F17 (459), and F27 (197) have posted reviews only from devices of unknown models. We confirmed that the "unknown" device category includes reviews posted from Google Play's website interface and certain types of emulators.

Figure 4.5 shows the relationship between the device release price (in Euros) and the device model age at posting time, for each of 9,942 presumed fake reviews posted from 344 unique device models. We consider that a device is low, mid, or high-end, if its release price is in the range $[0, 260)$, $[260, 450)$, and $[450, \infty)$ respectively [Tri17]. We classify a device model age into Fresh ($<$ 1 year), Middle-aged (12-18 months), and Old ($>$ 18 months). We found that 61.3% of reviews were posted from low-end, 38.2% from mid-end, and 0.5% from high-end devices, while 77.39% are from old

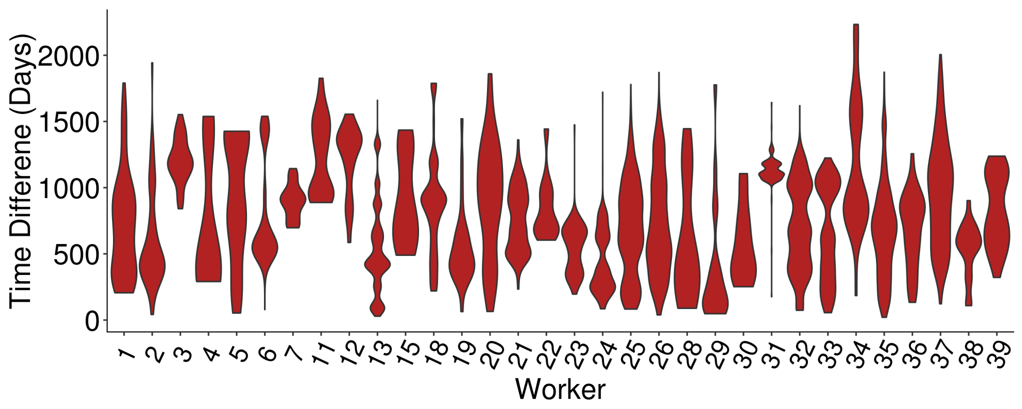


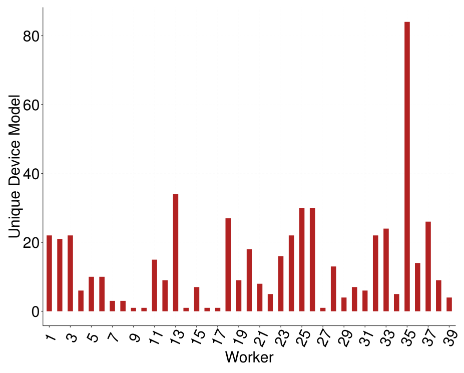
Figure 4.6: Per-worker distribution (violins) of the "age" of devices used to post reviews, i.e., the time difference in days between the review date and the release date of its posting device. Workers not shown had insufficient known device models. F3, F7, F11, and F31 use old devices. Most others (F1, F2, F13, F20, etc), use both newly released and old devices.

and 19.66% from new models. Further, most of these reviews were written from old low-end devices (45.98%), old mid-end (31.41%) and fresh low-end (15.31%) devices.

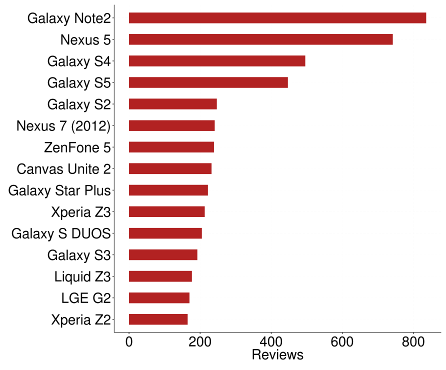
A notable case is that of tablets given away (price 0EUR, leftmost points in Figure 4.5) by the Uruguayan government to students as part of an inclusion plan named Plan Ceibal [cei]. Participants F25 and F32 used this device model to write 159 reviews for 137 apps. In addition, 3 reviews were posted from Galaxy S9+ devices whose price exceeds 600EUR (rightmost points in Figure 4.5).

Figure 4.6 shows the per-worker distribution of the "age" of their devices: the time difference between the review date and the device release date for all the fake reviews posted from known devices. 13 ASO workers have each posted at least 100 reviews from devices that are over 6 months old. Additionally, F13, F24, F25, F32, F33, and F35 have each posted at least 30 reviews from devices that are less than 6 months old. We conclude that different workers rely on stocks of either old devices, new devices or a mix of old and new, to post fraud.

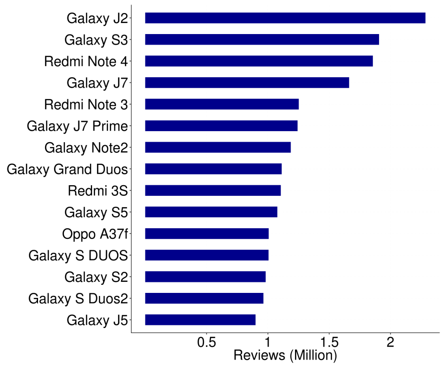
We found that 93.8% of the 9,942 fake reviews were posted from smartphones and 6.2% from tablets. Figure 4.7(a) displays the number of unique device models



(a)



(b)



(c)

Figure 4.7: (a) Number of distinct devices per ASO worker (F1 .. F39) including unknown category. F9, F10, F14, F16, F17, and F27 have only unknown devices; F35 used at least 84 distinct device models. (b) Device model popularity for top 15 devices used by ASO workers to post reviews. The 39 participants have used 344 distinct device models. (c) Device model popularity for top 15 devices in the wild. 11,934 unique device models were used to post over 198 million reviews in Google Play.

	Low-end	**Mid-end**	**High-end**	**Total**
Fresh	15.31%	4.35%	0.16%	19.66%
Middle-aged	0.00%	2.45%	0.34%	2.45%
Old	45.98%	31.41%	0.00%	77.39%
Total	61.29%	38.21%	0.50%	100%

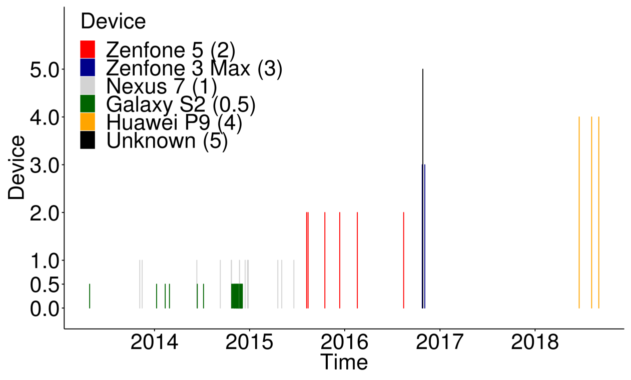
Table 4.2: Tabulated summary of Figure 4.5: reviews by device release price and device model.

used by ASO workers, including the "unknown" category (i.e., not among the 21,597 officially supported device models provided by Google [GP]). While F35 has used 85 unique device models, participants F9, F10, F14, F16, F17, and F27 have posted all their reviews from unknown devices. Figure 4.7(b) shows the popularity of device models used by the 39 participants, over all their 9,942 reviews posted from devices of "known" models. The top 6 most used devices by ASO workers to post these reviews are Galaxy Note 2 (836 reviews), Nexus 5 (742), Galaxy S4 (496), S5 (447), S2 (247) and Nexus 7 (241). Further, Figure 4.7(c) shows the popularity of the top 15 most popular devices, out of 11,934, that were used to post 198,466,139 reviews in Google Play.

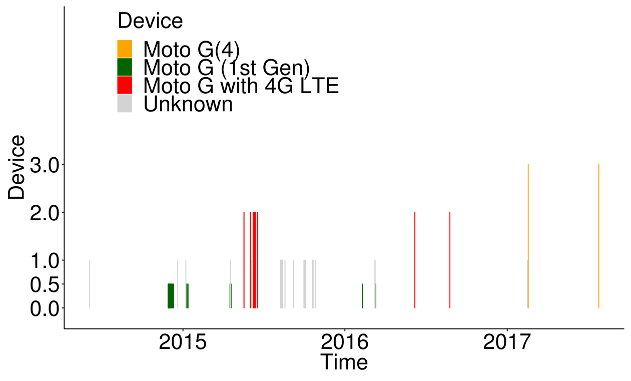
Table 4.2 shows the distribution of these reviews across categories: 45.98% of reviews are from old low-end, 31.41 % from old mid-end, and 15.31% from new low-end devices.

Finally, Figure 4.8 shows the review timelines of two accounts controlled by participant F13, in terms of the device models used to post those reviews. We observe that used device models change every few years, and that accounts can use multiple device models to post fake reviews at the same time.

Summary. We found ASO workers who claim to have access to large number of devices, either owned, or accessed through their communities of organic fraud. This



(a)



(b)

Figure 4.8: Device timeline for two accounts controlled by F13. (a) account has used 6 distinct device models to post 183 reviews in 5 years. It occasionally uses two devices at the same time, and changes devices every few years. (b) account has used at least 4 distinct device types in 4 years to write 163 reviews including 45 reviews from unknown devices.

claim is partially confirmed through our gold standard fraud data. Both in our interviews and in the quantitative study, we found that ASO workers have a diverse stock of low to high-end and new to old devices. Participants with many devices reported streamlined solutions to manage them, while those with fewer devices reported ways around cost limitations and compatibility issues, e.g., further outsourcing jobs.

4.3.4 Software

Team formation. 10 interview participants (P3, P5, P6, P8, P9, P11, P12, P14, P17, P18) said that they used Facebook and/or Whatsapp to create online teams. For instance, P6 said that *I have a Facebook group of more than 500 people, from different locations in Bangladesh, collected from various freelance groups in Facebook."*

P8 said *"I have posted invitations to Facebook groups, and I created a WhatsApp group for those who answered"*. P9 hints at eligibility criteria: *"To build a team, we first post message in Facebook groups. Then we contact those who respond, personally, and talk to them. We then decide if each is eligible, then we include him in our Facebook group."* P17 claimed access to multiple groups, *"We have 20 groups of real users in WhatsApp."*

Team communications. For communications, the above 10 participants claimed to use the corresponding Facebook and Whatsapp messenger app. P6 said *"I post the app link in my Facebook group, and ask them to download and post reviews."* P8 said *"When I get a job I post the link to my WhatsApp group, and they start writing the reviews."*. P11 said *"When I get a job, I send them messages in WhatsApp or I reach them personally."*

P7 however claimed to use specialized software: *"We have our own system where we push the apps. Users who use our system get the notifications about the new task and once they complete the task they get paid. Due to the privacy policy, I can't disclose the system name."*

Account maintenance. 5 interview participants (P1, P11, P13, P15, P16) said they access their accounts frequently. 12 participants mentioned that they access them manually. However, 3 participants (P13, P15, P16) said they have scripts and automatic login systems to access their accounts, keep them alive, and report if any are inaccessible. For instance, P13 said *"We have built a system in Linux where if we input 100 accounts, the system automatically logs into those accounts, and keeps them alive."* The participants who organize organic users said that organic users access their accounts regularly.

Job automation. P6 said that *"We can post reviews, ratings and installs using bots if the client has no problem. The bot names are like QZ362, YNX32, or something like these."* The rest of participants mentioned that they write their reviews manually.

4.3.5 Techniques: The Art of Evasion

Awareness of fraud detection. All interview participants are aware of their fake reviews being detected and deleted. All of them have reported that Google deleted some of their reviews. Although most of them have reported deletion as a small or negligible percentage (under 5%) of all the reviews they posted, four of our interview participants have said that 10–20% of their reviews were deleted. P6 said that the review deletion percentage depends on the app and ranges from 2% to 30%. Most participants said however that it is very infrequent for their accounts to be deleted. P2 said that *"Sometimes the email might be disabled; in that case the review will still be shown as written by a Google User."*

Perceived reasons for deletion. Participants reported diverse reasons for deletion:

• *Device re-use.* P5 and P10 blame it on using the same device to write multiple reviews for an app: *"I always track the screenshot that my workers provide as work proof. If I see two or more reviews from one worker have been deleted, I am pretty* *sure that they have used the same device for those reviews."* Proof of work details in § 4.3.12.

• *Improper VPN use.* P10 also blamed VPN: *"One safe way is, login from normal IP, then write review from VPN. If you login using VPN, Google will detect this as fraud."*

• *Improper app use.* P12 said that Google deletes reviews if the users *"do not care to use the app and keep it installed for more days."* More details in the app retention part of § 4.3.5.

• *Extended account use.* P3, P9, P18 report that using the same account to write many reviews in a short time, may trigger redflags.

• *Misfires of Google fraud detection.* P6 blames it on Google: *"Sometimes genuine reviews get deleted and sometimes multiple reviews from same devices don't get deleted."*

User account validation. P2 and P3 said that they prefer to use e-mail to validate user accounts. P3 also said that Google may force them to use phone numbers. Only P16 claimed that *"we use virtual phone numbers and Google accepts them."* All others said that they use real phone numbers to validate accounts.

Real phone numbers require access to SIM cards, which can be expensive. However, workers mentioned some solutions to overcome these constraints. For instance, P3, P10, P11 and P17 use friends and family: P3 said that *"We use our friends and family phone numbers. For example, I meet a friend on the road, I ask him to check the message and I use his phone number to verify an account."* P10 said that *"In Bangladesh one person can buy as many as 20 SIM cards using his credentials. [..] For example, for my 450 Gmail accounts I have used at least 200 phone numbers."* P5 mentioned that he borrowed SIM cards from friends. P7 and P15 use phone number verification services. Concretely, P7 said *"we pay other people to get a one-*

time code from their mobile SMS to verify those accounts." , while P15 said *"we use a person who has lots of phone numbers and provides a service to verify Gmail accounts."* P13 said that they purchase user accounts that are already validated.

Several participants reported limitations on phone number reuse. For instance, P3 and P8 said that one number could be used for 3–5 accounts but not immediately, while P1 said that *"between two verification using the same number, we have to wait at least 3 months."*

Review without install. When asked, P5, P10, P13 and P18 said that one can review an app without its prior installation from a device on which the account is logged in [gplb]: *"Click on install then stop installing immediately. The app would not be installed but it will allow us to write reviews."* We have tested this claim and verified that it works as suggested. This vulnerability breaks Google's intended security design [gplb] and facilitates the creation of fake reviews by reducing the amount of resources needed from the ASO worker.

App installation and use. 14 participants claimed to wait, open, or even use the app before posting a review for it. P5 and P9 wait a few hours before reviewing the app upon installation. P9 claimed to also use it for 5–10 minutes. P6 and P8 claimed to open the app 1–2 times before reviewing. P7 claimed to use the app as a normal user. P10, P13 and P16 claimed to keep the app open for 3–15 minutes before writing the review. P12, P14, P17 and P18 claimed to recommend to their online and organic teams to open the app for a few minutes and even use it before reviewing. P4 said *"We try to navigate all the pages of the app before writing the reviews."*

All the participants admitted to perform retention installs. P10 said that this is required to prevent filtering: *"Google takes 72 hours to verify the review. If you delete the app in this period, Google will drop the review."* Most participants said that they keep the app for a few days after reviewing it: 1 day (P1, P5 and P15), 2–3 days (P4, P5, P8, P10, P13, P14), 1–2 weeks (P17), and 7 days – 2 months (P2). P4 said that his workers keep the app until they need the space.

Upvote, Downvote. 6 of the 18 participants claimed that they upvote reviews written from other accounts controlled by members of the same team. P7 said *"We upvote the reviews put by our team and also other reviews which are positive."*

P10 said that his team downvote negative reviews of the apps they promote, in order to trigger Google's algorithms, thus remove such reviews. P7 said *"We provide upvote and downvote services to move positive reviews to the top and negative to bottom."*

Singleton accounts. 6 participants said they worked on apps where they had to create accounts just to post one review and then to abandon them. P1 and P2 said that such reviews are more expensive, \$8 and \$10 respectively. The reason for this is due to the effort to create an account (phone verification), which will not be amortized over multiple fake posting activities. Participants mentioned that the reason to do this is that Google does not filter reviews posted by singleton accounts, since its algorithms need more information to build a reputation for the account.

Account blending. 12 participants claimed to have seen jobs that required only the use of old accounts. However, P1, P2, P7, P10, P11, P13, P15 said that they have worked on jobs where they only used fresh accounts. P10 said that *"We do it because Google always keeps the reviews received from new accounts."* P1, P2, P7, P16, P18 said that they regularly use a mix of old and new accounts. In § 4.3.13 we report account creation and purchase strategies.

Noisy reviews. 8 participants claimed that they do not review other apps to camouflage their fraudulent behaviors. Of the on-site co-located teams, only P1 said that they review products for which they have not been hired, which they pick at random. 7 participants with online and organic team members said that their online team members review the apps they use regularly as normal users. P4 said *"That's why we use real users. We don't need to follow any strategy. The real users' behaviors serve the purpose of authenticity. We always instruct them to use other popular apps from their accounts."*

Device reset. P10 said that before logging in to an account, they flush the virtual device and change its MAC address. After using the account and virtual device pair for a few days to install and review apps, they log out and repeat the process with another account. They then leave the previous account unused for 1–1.5 months: *"after that interval, Google does not check that the new login is from the same MAC address as the previous one."* P13 claims to stay logged in to the account for 3 days, then they reset the device (via cccleaner) before logging in to the next account.

VPN use. P1, P3, P5, P13, P15 admitted to use VPNs, while the other 10 explicitly claimed to not use them. P3 said *"We use VPN or proxy only when it is required in the job specification. For example, if I need to install from USA, we have to use USA proxy server. (sic)"*

Emulator use. 2 participants (P10,P13) said that their teams use virtual devices running in laptops. The others claimed to use mobile devices or have access to real users who have physical devices.

Summary. Several of our interview participants confirmed several observations proposed in previous work: (1) ASO workers adjust their behaviors to avoid detection [ACF13, HSB$^+$16b, RA15, PCWF07, RA15, DCFJ$^+$14], including using VPNs [LCM$^+$15, TMG$^+$13], and mobile device emulators running on PCs [Xu13, LCM$^+$15, SMJ$^+$15]. However, P10 noted that improper use of VPNs can also trigger fraud filtering. (2) ASO workers also write genuine reviews, for products for which they have not been hired [ACF13, FML$^+$13, WXLY11, KCS18, RA15, DCFJ^{+}14]. However, this is only supported by participants who claimed to recruit and use organic ASO workers. (3) Some participants claimed to upvote their own reviews [PCWF07]. (4) Some participants also report using singleton accounts [MKL^{+}13, YKA16, RA15, SE15, XWLY12]. We however report a surprising motivation for this, which is not convenience, but rather a fraud detection strategy that exploits cold-start problems of Google's fraud detector.

Further, we identified new black hat ASO behaviors, that include downvoting negative reviews to promote their filtering by Google, and the unexpected benefits of using singleton accounts. Participants revealed ingenious solutions to bypass Google-imposed verifications, and validate the user accounts that they control, with real phone numbers. They provide circumstantial support for previous work studying the underlying technical and financial capabilities of social network fraudsters [TIB^{+}14].

Several participants mentioned to be able to bypass Google's check of preventing reviews without prior app installation. However, to avoid filtering, all participants said they use a combination of app interaction, delaying of review posting, and retention installs.

Further, we conjecture that the claimed use of a blend of older with newly created accounts, enables ASO workers to increase their base of accounts controlled, build the reputation of older accounts, and reduce detection of lockstep behaviors (§ 4.3.8), and the use of singleton accounts.

4.3.6 Review Burst vs. Campaign Length

We now present findings on the timing of the review process. 16 interview participants claimed to have seen jobs (1–45 in the past month) that specify how many reviews per day the workers should post. For instance, P5 said that *"Most buyers don't want to get all the reviews in a single day. They want a slow rate, like 2–3 reviews each day. To maintain this rate, they provide the review text on a daily basis."* However, P6 also said that *"some developers with money don't care whether reviews stay or not. They just need the number of reviews, quality doesn't matter. They just want short-time business."*

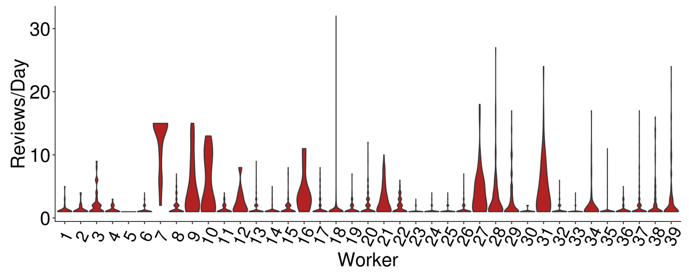
P1, P3 and P5 reported that they suggest to the hiring developers, the rate of posting reviews. P5 said *"If the developer asks for 30 reviews each day, I have to warn him that it's harmful to his app as Google may detect this as fake. Then I'll suggest to him that I will take 10 days to provide 30 reviews."* Most participants suggest 2–3 reviews per day, but some (e.g., P11, P14, P17, P18) recommend higher numbers, up to 30–40 reviews per day (P14).

Several participants suggested that the number of recommended daily reviews is a function of the app's existing review count. Concretely, P6 said, *"for new apps with less installs, it is better not to provide many reviews each day. But for popular apps, 20–50 reviews each day would be acceptable."*

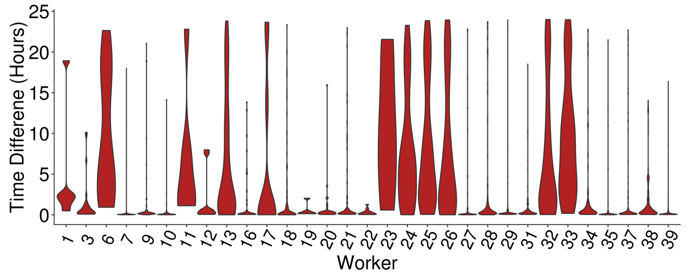
P10 revealed a different strategy: *"We provide a slow rate at the beginning. Like 1 review per day, or 5 reviews in 6–7 days. After 10 reviews we start posting 2 reviews each day. After 150 reviews we can provide 3–4 reviews each day."*

All the participants except P4 mentioned that they have seen ASO jobs that require a duration for the promotion campaign. P6 and P15 said that this is rare, and that developers are more interested in reaching a high number of reviews. However, P2 said almost all the jobs he has seen in the past month, mention the campaign length. In the past month, 5 participants have seen 3–5 such jobs, 4 have seen 6–10 jobs, and 5 have seen 11–35 such jobs. 12 participants reported longest seen required campaigns of 1–6 months, and 6 participants reported campaigns of 7–18 months.

Quantitative Investigation. Figure 4.9(a) shows the per-worker, violin-shaped distribution of the number of reviews per day, posted from accounts controlled by the 39 ASO workers, for each targeted app. Figure 4.9(b) shows the violin plots for the distributions of the inter-review times (only those posted within the same day). We observe several participants, e.g., F7, F9, F10, F16, F27, F28, F31, who tend to post more reviews per day, an do so in bursts. We also see participants, who even though write fewer reviews per day, still tend to post them in bursts (F1, F3, F19, F20, F23, F29, F35, F37-39). However, as also reported by the interview participants, we also found ASO workers who post only a small number of reviews per day and space them well through the day. Notably, F6, F11, F13, F23-F26, F32, F33 have a mean inter-review time of 8-9 hours.



(a)



(b)

Figure 4.9: (a) Per-worker distribution of the number of reviews per day for each targeted app. (b) Per-worker distribution of time difference in hours between consecutive reviews posted within one day for targeted apps. F7, F9, F10, F16, F27, F28, F31, tend to post more reviews per day, in bursts. F1, F3, F19, F20, F23, F29, F35, F37-39 post few daily reviews, but in bursts. Others like F6, F11, F13, F23-F26, F32, F33 post few daily reviews, but space them through the day (post one every 8-9 hours).

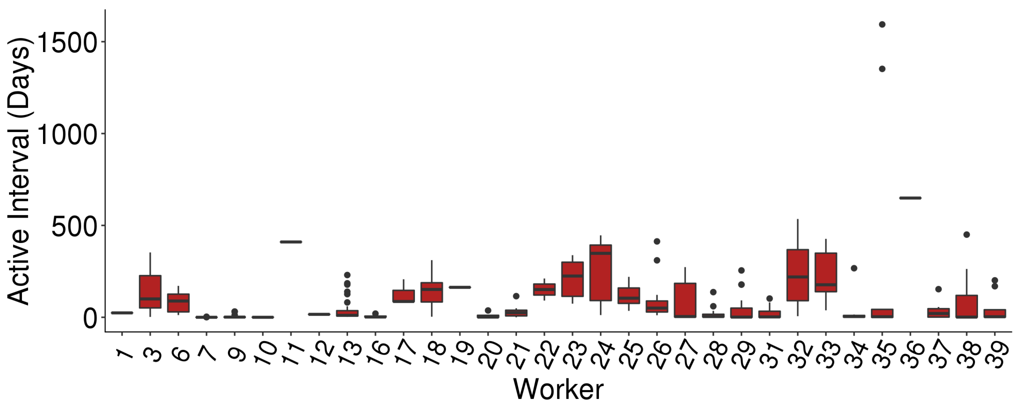


Figure 4.10: Per-worker distribution of active intervals (in days) over apps targeted. Each point represents the active interval of an ASO worker for an app. We observe workers who have posted reviews for certain apps, for more than 1 year, and up to more than 4 years.

Further, we call a worker's *active interval* for an app, the time span (in days) between the worker's last and first review for the app from accounts that we know he controls. Figure 4.10 shows the per-worker active interval distribution over the 316 apps that received at least 10 reviews from the 39 participants. Some ASO workers were often active for more than 1 year for an app.

Summary. We found ASO workers who post fake reviews in rapid bursts in both our qualitative and quantitative investigations. This is consistent with assumptions made in previous fraud detection work, e.g., [FML$^+$13, HTS16, LFW$^+$17, HSB$^+$16a, BSLL$^+$16, XZLW16, Xu13, GGF14, BXG$^+$13, KCA17, LNJ$^+$10, MVLG13, MKL$^+$13, KCS18, LCNK17, YKA16, FLCS15, DCFJ$^+$14, XWLY12, CYYP14]. However, multiple interview participants have revealed both developer and ASO worker assumptions that Google flags review bursts. Some participants also claimed to push back on developers who asks for many daily reviews. Our quantitative analysis reveals ASO workers whose behavior is consistent with these statements. Interview participants further revealed avoidance techniques that include (adaptive) rate control.

Rate control implies longer campaigns, as workers need more time to post their review quota. This is further supported by statements made by several interview participants and by evidence we extract from the quantitative investigation.

4.3.7 Accounts Per Device Strategies

Participants revealed mixed strategies for the number of accounts used on a device, and the number of reviews that they publish from a single device. P10, P11, P13, P18 said that they only log in to one account at a time, on any device that they control. P18 has 30 devices and 30 accounts, and a 1-to-1 mapping between accounts and devices. P11 said that *"If we provide multiple reviews from one device, Google will keep only one review for that device."* P5, P6, P7, P8, P9, P15 and P16 claimed to log in to multiple accounts (2–5) from a single device and also instruct their remote workers to do the same. However, P5 and P9 claimed to only provide one review from one device for an app. P15 and P16 keep track of which accounts they use to log in to any device, and once they log out from one account, they wait 7–10 days before they use it again.

P5 admitted using a fixed set of 2–3 accounts to sign in to one device simultaneously, then uses those accounts to review several apps. However, he also claimed to review apps using only one account from such devices. P6 claimed that he instructs his workers to log in to at most 2 accounts from any device at a time, from which they can review the app using both accounts. P8 claims to login to 5 accounts on his device, and his Whatsapp group members log in to at most 5 accounts per device.

Summary. ASO workers generally claim that it is possible to review an app from different accounts using the same device. We have tested this claim and verified that it works as suggested. This vulnerability facilitates the creation of fake reviews

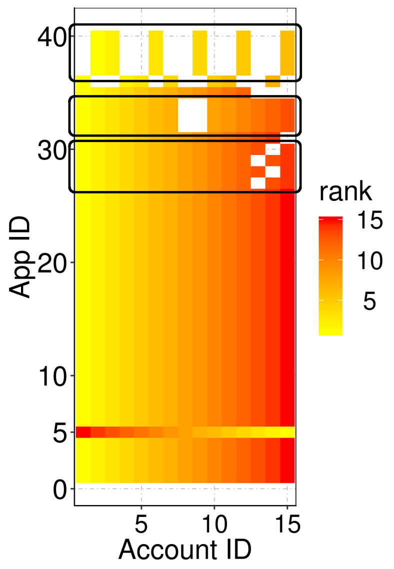
4.3.8 Lockstep Behaviors

Interview participants revealed different strategies to choose which of their accounts and devices to use for a job. Several participants revealed lockstep-indicative behaviors, based on a spreadsheet of accounts and devices that they maintain across all their jobs. P5, P7, P10, P13, P18 select the devices in a sequential, round-robin manner, while P5, P7, P13, P15, P16, P18, use their accounts in a sequential fashion. For instance, P15 claimed that *"We have statistics on how many times an account was used previously. From there we try to find accounts that have been used fewer times. We also track which device was used for which account, so next time we use the same device for that account."*

Others however claimed non-lockstep indicative behaviors. 7 of the 18 participants (P6, P8, P9, P11, P12, P14, P17) claimed a random choice of accounts and devices, including made by their remote online employees. P16 claimed to monitor the reviews filtered, and choose accounts based on their filter avoidance success rate.

To investigate lockstep behaviors in the gold standard fraud data (§ 4.2.2), we used frequent itemset mining [AS94, MLG12] to discover sets of apps that are co-reviewed by many accounts in the same or similar order. Intuitively, a set of apps reviewed by the same, many user accounts, is said to be "frequent". More formally, let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a set of apps, and let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ be a set of users in Google Play. We say that a set $A \subseteq \mathcal{A}$ is $s-$frequent if $\frac{|\{u \in \mathcal{U}; A \subseteq T_u\}|}{|\mathcal{U}|} \geq s$ where $T_u = \{a \in \mathcal{A}; a \text{ is reviewed by } u\}$.

We used the A-priori algorithm [LRU14, AS94], to find per-worker *maximal* frequent itemsets: frequent itemsets for which none of their immediate supersets are frequent. 25 of the 39 participants had maximal frequent itemsets with $s = 0.5$. That is, they used at least half of their accounts to review common subsets of apps.



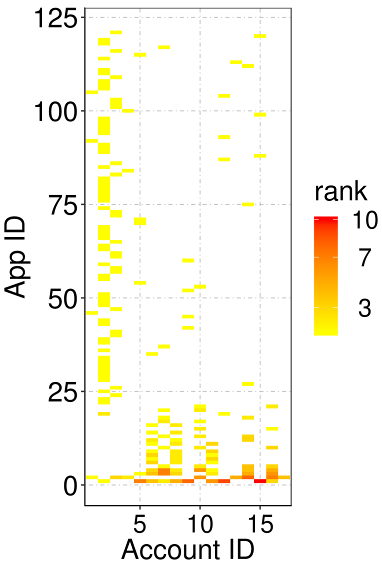
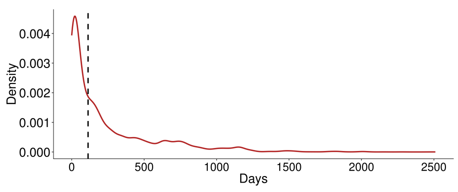


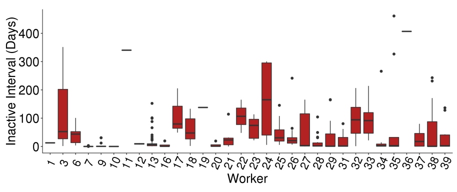
Figure 4.11: Lockstep matrices for F7 (left) and F32 (right). Rank (color) indicates the order in which an account was used to review an app. F7 exhibits strong lockstep behaviors, having used almost all his revealed 15 accounts to review all the 40 apps (exceptions shown within black rectangles). F32 however exhibits less obvious reviewing patterns.

Figure 4.11 shows *lockstep matrices* for two of the ASO workers. In the lockstep matrix M_{ij} of a worker, columns are user accounts controlled by the worker and rows are apps reviewed from those accounts. $M_{ij} \in [n_w]$ denotes the chronological order of the review posted by account j on app i. n_w is the total number of reviews posted by the worker to app i. ASO worker F7 (left) shows a nearly perfect lockstep behavior with the same set of 15 accounts used for almost all the 40 apps, and in the *same order*. We also see attempts at "variation": F7 uses his accounts in exact reverse order to promote app 5. Further, for several sets of apps (black rectangles in Figure 4.11), F7 does not use the same set of accounts, and uses all his other accounts in the same order.

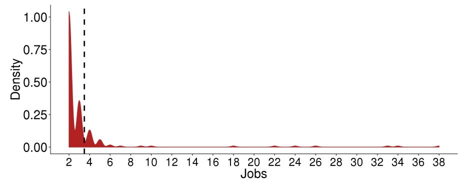
However, 14 participants exhibit less pronounced lockstep behaviors, e.g., F36 (Figure 4.11 right). Out of 121 apps reviewed, in only two apps, F36 used more than 50% of the 17 accounts he revealed.



(a)



(b)



(c)

Figure 4.12: (a) Relative likelihood for the time difference between launch time and reviews by ASO workers, for 585 apps that received at least 10 fraudulent reviews. Vertical dashed line is the median. (b) Per-worker distribution of the maximum inactive interval measured in days for each targeted app. 8 participants, e.g., F7 and F9 are intensely active, however, F3, F24, F32 and F33 exhibit more evidence of later rehiring. (c) Density function of number of jobs received by ASO workers from the same developer. One worker worked on 38 apps of the same developer. The vertical dashed line corresponds to the median value.

Summary. 6 out of 18 interview participants claimed lockstep-indicative behaviors; 25 of the 39 quantitative study participants exhibit lockstep behaviors, some even using their accounts in the *same order* to review multiple apps. This is consistent with and provides evidence for assumptions made in previous work, e.g., [SMJ+15, TZX+15, YKA16, XZ15b, JCB+14, CYYP14, SLK15, XZ14, XZ15a, LFW+17].

However, we also report claims (8 of 18 participants) and evidence (14 of 39 participants) of random account and device choice. We conjecture that ASO workers may adopt evasion strategies, e.g., by using different sets of accounts for different jobs, and use organic workers, less likely to be frequently active at the same time.

4.3.9 Timing: Fraud Event Points

First Reviewers. 14 participants claimed that they have promoted recently released apps, and either the hiring developer mentions that the app was recently launched, or that they infer this information based on the number of installs and reviews when starting the job. Declared numbers range from 1–2 jobs in the past month (P1, P11) to 20–40 (P9, P10, P13). P7 said that *"We even work on apps which are going to be launched soon. A few of our clients rely on our agency from pre-launch to launch and then post-launch."*

Re-hires. All 18 participants claimed to have been re-hired for apps that they previously promoted (total times M = 186.1, SD = 190.7, Min = 15, Max = 600). P1 said that *"If the app is getting bad reviews, the developer will hire us again to get good reviews. We have seen this case for minimum 30 to 40 apps per year."* P12 said *"I have around 20 regular clients. They hired me for the same app, around 40–50 times."* Further, all of the 18 participants claimed to have regular customers, who hire them to promote multiple apps.

Quantitative Investigation. Figure 4.12(a) plots the time difference in days, between the app launch time and the posting time of each review from a fraudulent account controlled by any of the 39 participants in the quantitative study (§ 4.2.2), over the 585 apps that received at least 10 fraudulent reviews in total. The distribution is left-skewed, with 50% of the reviews being posted after less than 3 months after app launch. However, we observe cases where the first reviews from any of the accounts of our 39 participants, are posted long after the app was released: the median and 3rd quartile are 113 and 344 days respectively. Thus, about 25% of the fake reviews were written after one year.

We call the *inactive intervals* of an ASO worker for an app, to be the time differences between consecutive reviews that he posted to that app, from accounts that he controls. Figure 4.12(b) shows the per-worker distribution of the *maximum inactive interval* computed over each app that the worker reviewed from accounts that he controls. We show only the workers with enough points to compute statistics. 8 workers have very short inactive intervals, thus are more intensively active for the apps that they target. However, ASO workers such as F3, F24, F32 and F33, have longer inactive intervals, suggesting rehiring. For instance, we found 16 cases where the worker was inactive for more than 8 months for an app.

Figure 4.12(c) plots the density function of the number of apps uploaded by the same developer, and reviewed by the same worker, over the 39 workers of the quantitative investigation. We observe that the mean number of jobs assigned is 3.48, and 7 workers have been hired by the same developer more than 10 times. We found one developer that hired 6 workers to each promote at least 10 apps.

Summary. Our qualitative and quantitative studies provide evidence confirming observations and assumptions made in previous work, that (1) ASO workers tend to be hired early after app launch, or even before launch, to control review sentiment, see e.g., [FML+13, YKA16, LNJ+10, MKL+13, MLG12, Xu13, KM16] and (2) developers rehire some of these workers at later times, when honest feedback reduces the product rating [KM16].

4.3.10 Review Writing

We asked interview participants about, and report findings on the source of review text, plagiarism, and review length:

Review text source. 2 participants (P3, P4) claimed their reviews are original. The other participants said that they may receive or request the review text from the developer, and they may also write their own reviews. Several participants (P1,

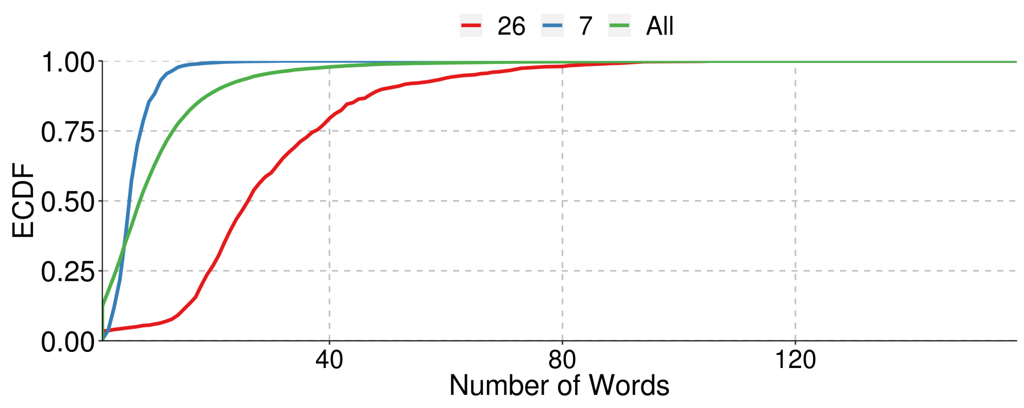


Figure 4.13: Empirical CDF for two extreme behaviors shown by two participants. All other workers have their corresponding CDF between these two curves and are not displayed for better visualization. We note that $\mathbb{P}(Length \leq 25|F3) = 0.99 \gg \mathbb{P}(Length \leq 25|F26) = 0.46$, and the all-worker ECDF is closer to worker 7 who writes shorter reviews.

P5, P6) said that the reviews are given by the developers, but if the developer needs many reviews, they also need to complement by writing new reviews. P11 reported that some developers provide review samples, from which they are supposed to create similar reviews. 3 participants (P7, P8, P15) said that they either prefer or even ask the developer to provide the review text. P3 and P13 said that they check and study the app before posting a review. P13 asks developers to provide the app's main features to create a review.

Review posting process. The participants revealed a mixed strategy of typing the reviews directly on the device, and cut-and-pasting them from another source. 11 of the 18 participants said that they type the reviews directly from their devices. For instance, P5 said that they cut-and-paste reviews if given by the developer, otherwise they type their own reviews. Several participants organize teams of remote ASO workers, thus stated that they are not aware of their review-typing actions.

Review plagiarism: 8 participants (P1, P3, P5, P12, P13, P14, P15, P18) denied plagiarism and self-plagiarism. 7 participants (P2, P4, P6, P9, P11, P17) however admitted that they plagiarize reviews. P2 blamed it on developers: *"Yes, sometimes*

Review Text	**Geometric Mean**
good	25
Good	20.83
nice	20.19
app	13.56
Love it	11.83
Awesome	10.95
Ok	10.24
Not bad. Keep it up! :-)	10.09
Great app	9.94
Excellent	9.79
Nice app	9.79

Table 4.3: Top 11 repeated reviews sorted by the geometric mean between the number of distinct workers that wrote it and the frequency of the review.

we copy, but only if buyers mention the source, for example, apps hosted in other sites." P4 said that *"we don't copy-paste. But our reviews are short and sometimes similar."* P16 said *"We have a review data set, and we use those reviews for all apps. Sometimes we change a the reviews bit for different apps."* P9 said that *"Not exact copy-paste. But sometime we copy and modify reviews from other apps that are similar."* P12 also complained about some organic users, who are careless and write random comments, e.g., "nice game" for a non-game app.

Review length: 11 participant claimed that their reviews exceed 10 words (10–40). P3 and P4 admitted that their reviews are short (3–5 words). P4 motivated this choice: *"We don't use many words or big sentences because Google may match the pattern. We always use short messages like "Good app", "Awesome", "Fantastic". These are very common but easy to write and Google may not complain."* P6 argued that *"if you write too long reviews, they will certainly look like paid reviews, because real users don't have time to post a paragraph."*

Quantitative Investigation. Figure 4.13 shows the empirical CDF of the review word count over all the reviews posted by the 39 participants, and also only for F7 and F26, who wrote 542 and 771 reviews respectively, and are the ASO workers with the most distant CDFs from one another: $\mathbb{P}(Length \leq 10|F7) = 0.88 \gg \mathbb{P}(Length \leq 10|F26) = 0.06$. The overall fake review word count CDF is closer to F7, with the overall $\mathbb{P}(Length \leq 10) = 0.63$. This reveals that some workers write longer reviews than others. We performed a Welch two sample t-test to compare the differences between the average length of reviews by workers 7 and 26: ($H_0 : \mu_7 = \mu_{26}$, $H_a : \mu_7 \neq \mu_{26}$), and obtained $p - value < 2.2e - 16$ suggesting that there is a statistical difference between the two workers with respect to the length of their reviews ($\bar{x}_7 = 36.4$, $\bar{x}_{26} = 170.33$).

Further, we identified exact review duplicates among the 21,767 reviews posted by the 39 participants (§ 4.2.2), and sorted them by the geometric mean between the number of ASO workers who have written the review and its overall frequency. An advantage of the geometric mean is that it gives a balance between two quantities that are in different ranges. 993 reviews were empty (154.37). Table 4.3 shows the next 10 most repeated reviews ordered by geometric mean. We note that these reviews are short, generic, and app-agnostic. This analysis validates the survey answers by some ASO workers, that short reviews may be preferable since long reviews may trigger Google's defenses and block their content.

Summary. Most interviewed participants said that the text of the reviews is provided by the developers, but also they can write their own reviews. Consistent with previous observations, e.g. [MKL$^+$13, FML$^+$13, Xu13, HTS16, LNJ$^+$10, SE15, MVLG13, XZ15b, KCS18, LCNK17, RA15, YA15, MLG12, KCA17], several participants admitted to reuse common linguistic patterns and copy reviews across similar products. We also confirmed this finding in our quantitative study.

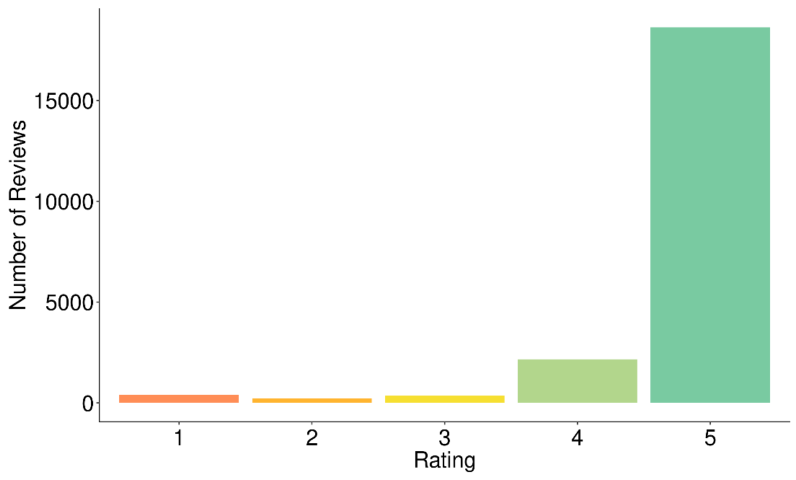


Figure 4.14: Rating distribution: workers get mostly jobs that consist in promoting apps. 92% of reviews were either 4 or 5 stars, while 4.7% were 1 or 2 star reviews.

Further, most participants claimed to write short reviews, which is also reflected in our gold standard fraud data. Previous work, e.g., [KCA17, MVLG13, JL08, KCS18, FML$^+$13, LCNK17], also made this observation, and attributed it to the fraudster lack of experience with the product. However, we also present evidence of ASO workers who post much longer reviews. We conjecture that fraud evasion can also be a factor.

4.3.11 Ratings

Rating choice strategies. All 18 interview participants admitted writing mostly 4 or 5-star reviews unless they receive special instructions from the developers. 8 participants (P3, P7, P8, P9, P10, P11, P14, P18) claimed that they receive guidelines from developers on how to distribute the ratings. For instance, P12 said that *"developers request us to write a few 4, 3 and even few 1 star reviews."*

Several participants claimed to maintain their own ratio. For example, P5, P16, P17 claimed to post a 10% vs. 90% ratio of 4 to 5 star reviews, P2, P10, P18 have a 20%-80% ratio, P1, P9 have a 30%-70% ratio and P13 has a 40%-60% ratio. 3 participants (P6, P11, P14) claimed that they do not maintain any ratio, while P4 and P12 post only 5-star reviews.

3 participants claimed to post low ratings, in order to avoid detection and camouflage their behaviors. For instance, P6 said that: *"if the average rating goes up to 4.3 or 4.4, I also write a few 3-star reviews."* P7 said that *"when posting more than 200 reviews, we suggest to the client to have at least 5 to 6 reviews with 3 star ratings."* P15 claimed to post a 10%-30%-70% ratio of 3, 4, 5-star reviews.

Negative campaigns. When asked if they were ever hired to post negative (1-2 star) reviews, and how many such jobs they worked on, only two participants said that they participated in such negative review campaigns. P3 had participated in only one such job, but later morally objected to it, while P4 also admitted to have worked on only a few such jobs (5-7). The other participants said that they never participated in negative campaigns. We did not ask participants how many such campaign jobs they have seen.

The gold standard fraud data we collected from 39 participants confirms that 95.52% of the 21,767 reviews posted from the accounts they control, were either 4 or 5 stars. Only 1.67% were 3-star and 2.81% were 1 or 2 star reviews.

Figure 4.14 shows the rating distribution over the data collected on a set of 16 other fraudsters. We can confirm what fraudster claimed during the interviews, that is, that they are mostly hired to promote apps and in consequence give positive reviews. We conjecture that the negative (1 and 2-star) and neutral (3-star) reviews can not only be the result of negative campaigns, but may also be due to camouflage strategies of fraudsters as four of the fraud workers claimed during the interview that this may help the arriving of reviews look legit.

Summary. Both interview participants and gold standard fraud data reveal the prevalence of positive ratings. This confirms observations and assumptions made in previous fraud detection work, e.g., [MKL$^+$13, ACF13, KCA17, MVLG13, KCS18, RA15, MLG12, XZ14, XZ15a]. However, we found that negative review campaigns (or negative ratings) are unpopular. Further, several interviewed participants reported rating-level detection evasion strategies, e.g., the sprinkling of neutral and negative ratings, among positive reviews.

4.3.12 Proof of Work

After ASO workers finish their jobs, it is expected that they show proof of their work. 12 participants claimed that they take screenshots of their reviews and send them to the developers. 5 participants said that they send the usernames of accounts used to write the reviews. P6 claimed *"I check my reviews for 2–3 days and then send the permalinks that are direct links of the review I post, or names I used to post the reviews."*

Team-level verification. Team members get their job verified by ASO admins. For example, P3 mentioned that *"[..] we ask everyone to post reviews in the team. Then I track how many reviews we provide and they also send me the screenshot. If the buyer requires the screenshots I send him those too."* P6 claimed that *"If we get a report that any review is being deleted then we check that user's mobile and ask him to provide a screenshot of the app installed immediately. If he fails to provide that, I flag him as a bad user and we consider him less for the next tasks."* P9 also checks that their team does not post several reviews from the same device, by looking at the screenshots sent.

Follow-up. P3 said that *"Sometimes, the developer keeps track of the reviews we post, and gives us 24 hours to show that the reviews are alive. If any review is* *deleted during this time, we have to re-post the reviews."* P3 stated that *"I have to ensure the buyer that after 24 hours they will have the required amount of reviews, after deletion. For that I have to check and provide some reviews again if some are deleted."* P7 claim to provide guarantees of reviews sticking for 5–7 days and refill deleted ones for free.

In summary, developers verify that fraudsters satisfy the terms of the job, including that posted reviews are not filtered for multiple days. Fraudsters need to replenish deleted reviews for free. We observed claims of interactive and even team-level, hierarchical work verifications: teams verify the work claims of their members, and punish cheaters.

4.3.13 Account Creation

13 of the 18 interview participants, mentioned use of fake name generators, e.g., [FNG], to name their user accounts. Some of them create account names to correspond to specific countries and continents, since it is sometimes required by developers. P2 even claimed to send the chosen names to the employer for feedback. P11 claimed to get random names by using Google search and P7 said that they have their own database of names. P2 claimed they also modify the name pattern from this fake name generator site. He said that "Sometimes we change the name format. For example, if the fake name generator site produce 'Stephen G. Lord', we remove the middleinitial 'G.' from the name to make it more realistic (e.g. 'Stephen Lord')." P4, P7 and P14 said that organic ASO workers use their own personal accounts which have real user names.

7 participants said that they add pictures to the account profiles, which they retrieve from different sources, e.g., Google search, Google Plus, pixabay.com, to

create the illusion of authenticity. P9 said *"After we use fake name generator to create the account name, we search the name in Google Plus and choose a profile, then we choose a random person from the list of followers and use his image for the account profile."* P10 however said that *"We use no picture as picture defines your demographics. Buyers do not want this now."*

Creating and Purchasing Accounts. 6 participants (P1, P3, P7, P10, P13, P16) mentioned that they create new accounts periodically, ranging from once a day (P10) to once a month (P16). P2 and P9 said that they create new accounts when they don't have enough accounts for a job, e.g., when the employer asks for accounts from a specific city or country. P5 and P18 create new accounts when Google deletes some of their accounts. 5 participants (P1, P5, P13, P17) admitted to buying new accounts. P1 claimed to have purchased more than 10,000 accounts, while P13 claimed to have purchased 47,000 accounts. Two participants (P1 and P3) age their new accounts (1–2 months) before using them for promotion jobs.

Participants had a diverse set of strategies to create new accounts, and use them in ASO jobs. Two participants claimed proactive strategies. One participants claimed to create 30-50 new accounts each month, and age them for at least 2 months before using them. The participant refered to accounts that are 2 months old to be "new accounts" and "old accounts" to be accounts that are older than 2 months old. This participant claimed to use a mix of 50% old vs. 50% new accounts for ASO jobs. The second participant claimed to proactively create 5-10 new accounts at a time, at random times, but twice a week on average, as back up. This participant also claimed to age the accounts for 1 month before using them. One participant claimed to create new accounts only on demand (on average 30 to 40 at a time) when the employer asks for more more reviews than the accounts controlled can provide, or if we don't have accounts located in the region requested by the employer.

Only one participant admitted to purchase accounts from other parties, at random times. When asked what they do when the number of reviews requested is higher than the number of accounts that they have, the participant said that he purchases new account. One participant said that they collaborate with other fraud teams. Two participants said that this has never happened. One justified this statement by saying that his company controls 1,500 user accounts. The other, the "leader" worker, organizes an organic community of 15,000 members.

4.3.14 Credential Reuse

P1, P13, and P16 said that all their passwords are random. P8, P10 said that they use the same passwords for all their accounts. P8 said however that he does not know the password of his team members. P3 said that *"I use the same password for all the accounts, but I use two step verification for all my accounts to prevent hacking."* P5 claimed that most of his passwords are the same, and for the rest he uses a pattern based on the account, to make the password easy to remember. P2 claimed that *"we use some common strategies to make the passwords easier to remember for us."* P11 and P14 use common passwords that are easy to remember. P6, P12, P14 claimed that their team members (including organic ASO workers) use their own passwords. P1, P2, P7, P13, P15, P16 said that they write down the passwords of the user accounts that they control. P2 and P13 store their passwords in an excel sheet. We observe that memorability seems to affect ASO worker choice of user account names and passwords, leading to poor authentication practices. This makes workers vulnerable to account hijack attacks.

Further, despite the importance of stolen accounts for the underground economy argued by Onaolapo et al. in [OMS16], we found in this study that none of the participants interviewed, acknowledged using stolen account credentials. We conjecture that this could be the result of their manual operational preferences as described above or the evolution of an evasion tactic.

4.3.15 External collaborations

5 participants mentioned that they collaborate with external fraud teams. For instance, P2 said that his company seeks collaboration with 4-5 other companies when they can not provide all reviews (e.g., number of reviews required is greater), and share the profit. 2 participants said that they only tried to collaborate once, but without success.

Two participants said that they did not see jobs that asked them to collaborate with other workers, while 2 participants said that they never collaborate with other workers. Only 1 participant said that he had seen such jobs. All participants had either collaborated with other teams in fraud jobs, or had heard of other teams collaborating. One participant said that his company collaborates with 4-5 other companies when developers request more reviews than they can post, and share the profit. Another participant reported having a bad first experience with collaboration which led his company to stop collaborations. Two other participants said that they have seen other collaborations but they do not do it.

4.3.16 Account Abandonment

9 participants claimed that they have never abandoned an account. 6 participants mentioned that they abandoned their accounts only if Google blocked such accounts. P17 and P18 said that they give up on an account only if its reviews are continuously deleted by Google. P2 said that *"I remember I used some accounts for many*

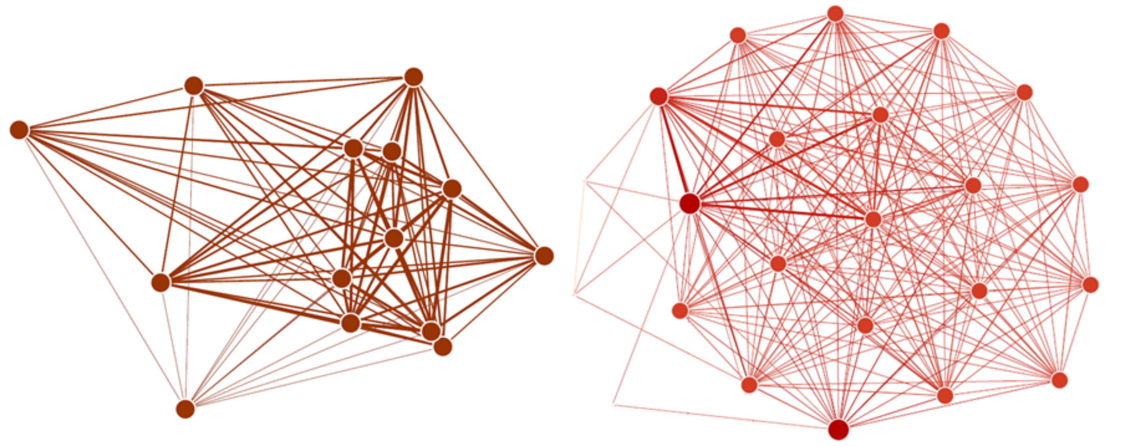


Figure 4.15: Co-review graphs built over the accounts claimed to be controlled by (left) F13 and (right) F32. Edge width is proportional to the number of apps reviewed in common by the endpoint accounts. 14 accounts revealed by F13 form a clique, and on average, any two accounts reviewed 78 apps in common.

different jobs, like YouTube review, Google Place review, app review and other services. Google disabled those account for misuse. After that, we dedicated accounts for app reviews.". P3 and P9 claimed that they never abandon accounts, but use them intermittently. P7 claimed that only 2-3% of their accounts were deleted by Google in the past 2 years. P14 said that this never occurs to them, since they use organic fraudsters.

4.3.17 Validation and Efficacy of ASO

Validation of quantitative study. Collecting ground truth fraud data attributed to the workers who created it, is a difficult task. We believe that any process to obtain such ground truth data needs to involve the workers. In addition, to gain confidence in the correctness of the accounts claimed to be controlled by the 39 workers, we used *co-review graphs* built over the accounts claimed to be controlled by each worker: nodes are user accounts, and edges have weights that denote the

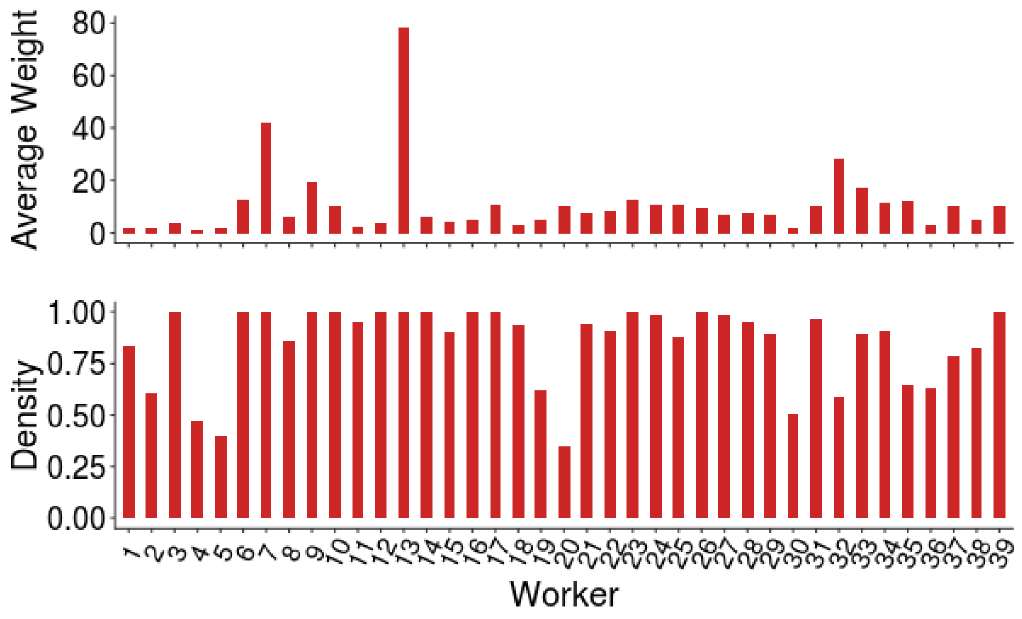


Figure 4.16: Density and average weight for co-review graphs of 39 ASO workers. 12 workers have complete graphs (density=1). 30 workers have graphs with density at least 0.75.

number of apps reviewed in common by the end-point accounts. Figure 4.15 shows example co-review graphs built over the accounts revealed by F13 and F32.

Figure 4.16(top) shows the average co-review weight of the accounts claimed to be controlled by each of the 39 ASO workers, i.e., the ratio of the sum of all edge weights to the number of edges. Figure 4.16(bottom) shows the edge density of the worker co-review graphs, i.e., the ratio of the number of co-review edges to the maximum number of edges possible in that graph. The co-review graphs of 12 of the workers are cliques, i.e., any two accounts have reviewed at least one app in common. Further, the co-review graphs of 16 workers have an average weight of at least 10, up to 78.61 for F13. This is in contrast to the probability of co-rating two apps in Apple's China App Store, of 0.163% (computed over 0.5 million random accounts) [XZ14].

In addition, we manually investigated the accounts revealed by the 39 workers, and found multiple instances of repeated profile photos, mostly of glamorous people,

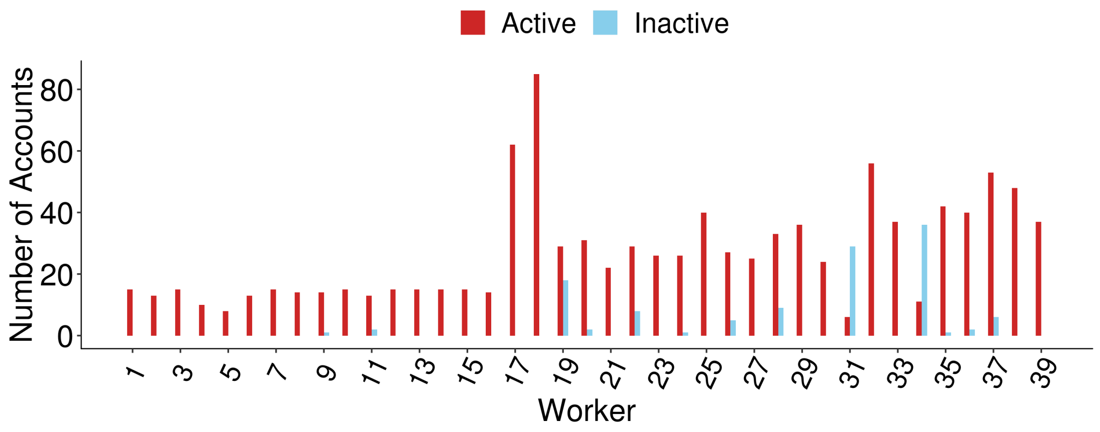


Figure 4.17: Active vs. inactive accounts controlled by the 39 quantitative study participants. We observe diverse success in keeping accounts active on the long term.

and simple patterns in the account names.

Efficacy of ASO. To investigate the efficacy of the ASO strategies employed by the 39 workers who participated in our quantitative study, we look at (1) the number of accounts that they control that are still active, and (2) the impact of their ASO campaigns.

Figure 4.17 shows the number of accounts controlled by each of the 39 workers, that are active and inactive (i.e., Google returns 404 not found error). Of the 1,164 accounts known to be controlled by the 39 workers, 120 were inactive (10.30%) in May 2019. Qualitative study participants stated that they never abandon accounts unless they are closed by Google or Google filters all their reviews. Thus, Figure 4.17 reveals diverse success among the 39 ASO workers, in terms of being able to keep their accounts active long term: while a majority of the workers have all their accounts still active, including the workers with more than 40 accounts, several workers had a majority of their accounts closed. Notably, 36 out of the 47 accounts controlled by F34 are closed, as are 29 out of 35 accounts of F31.

In addition, we studied the *impact* of a worker on each app on which he has performed an ASO campaign. We denote the impact I_A of an ASO worker W for

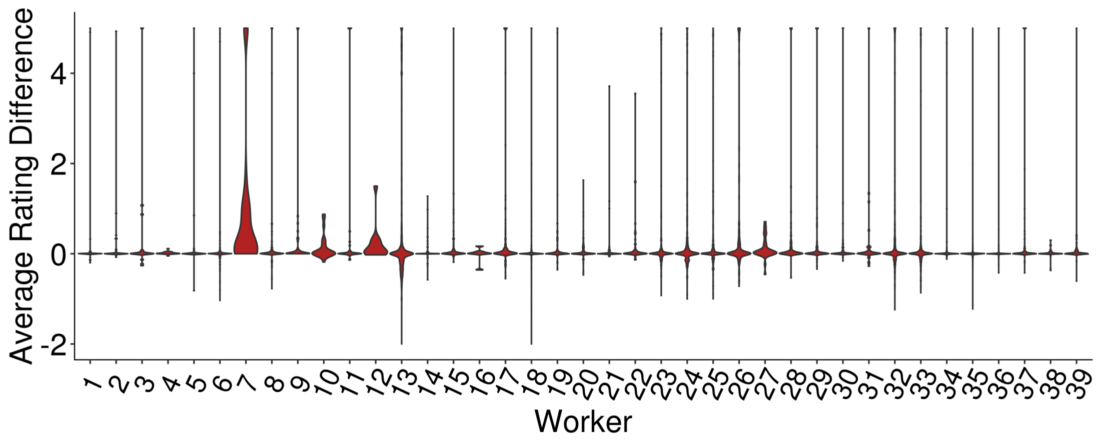


Figure 4.18: Impact of campaigns conducted by the 39 quantitative study participants, on the average rating of apps for which they campaigned. We observe diverse success in increasing the average rating of targeted apps.

an app A to be the change in A's rating during W's active interval. Specifically, $I_A = R_f - R_i$, where R_i is A's "initial" average rating, i.e., before the first review posted by the worker for A, from any of his accounts, and R_f is A's "final" average rating, after W's last review posted for A. Figure 4.18 shows the violin plots of the distribution of impact values, over all the apps campaigned by each of the 39 ASO workers, from all the accounts that each controls. We observe diverse abilities of these workers. For workers like F7, F12, and F21, we observe only positive impact on the average ratings of all the apps that they target. Most workers however have mixed impact, with many of their targeted apps seeing up to 5 star increase in average rating during their active interval, and a few others seeing up to a 2 star drop. We observe however that overall, apps seem to benefit from the campaigns in which these workers have contributed.

We conclude that different strategies have different impact on the ability of ASO workers to avoid detection and impact the ratings of apps that they target.

Our study has several limitations. First, we do not know all the accounts controlled by the 39 ASO workers. Second, we cannot pinpoint the exact strategies that are responsible for the success to maintain accounts active or ensure that reviews are not filtered. Such an analysis would require detailed experiments that explore the impacts of altering a single feature of a fraud detection algorithms that is kept a close secret. Third, the impact that we computed, is oblivious to simultaneous campaigns being conducted by other workers on the same apps. Finally, our computed average rating of an app is imperfect, since (1) we do not have access to ratings posted without reviews, and (2) may not correctly model Google's algorithm, that e.g., may assign weights to ratings based on perceived usefulness, fraudulence or recency [Per19]. We describe more limitations of our studies, in § 6.12.

4.4 Discussion and Recommendations

The varied capabilities, behaviors and evasion strategies claimed and exhibited by the studied participants, suggest that fraud detection solutions should cast a wider net. While some of our participants seem to fit the mold of assumptions made in previous work, we present claims and evidence of evolution, perhaps fueled by the competitive nature of the market. In this section, we propose disruption strategies for each vulnerability point identified in the fraud workflow of Figure 4.1, and discuss potential implications of our study's findings, on future fraud detection and prevention solutions. The opaque nature of commercial fraud detection systems prevents us from establishing the costs and scalability of implementing the proposed recommendations, or from determining if they are already implemented. However, manual verification of statements made by ASO workers revealed several weaknesses in Google's defense. Some of the following defenses propose to address them.

VP1: Proactive Fraud Monitoring. Recruiting WhatsApp/Facebook groups need to aggressively accept new collaborators. We verified that these communication channels are easy to infiltrate. Thus, we recommend to proactively detect campaigns at this point, and flag apps likely to receive fraudulent reviews, and suspicious accounts engaged in posting fraud.

VP2: Device Fingerprinting. We observe that device models and their per-country popularity can be used to detect reviews written from accounts claiming to be from a country where the posting device is not popular. However, this vulnerability could also be used by ASO workers to blend in with normal users, by mimicking the distribution of devices observed in Google Play.

Further, this device-model leaking bug can also be used by computer criminals to perform reconnaissance on potential victims. Figure 4.7(c) in shows the top 15 most popular devices, out of 11,934, that were used to post 198,466,139 reviews in Google Play. An adversary could use this bug to for instance, identify owners of device models known to be vulnerable, e.g., [Bar18, War17]. We notified Google about the dangers of this bug, see § 5.3.3.

VP3: 1-to-1 Review-To-Device. Our interviews and experiments revealed that a user can download an application once, and review from all the accounts registered on a device. We suggest enforcing that a device can be used to post only 1 review per downloaded app.

VP4: Organic Fraud Detection. We suggest the use of account activity levels to differentiate organic from inorganic (sockpuppet) accounts. Organic ASO workers are likely to use their devices continuously, like the normal users that they almost are. Sockpuppet accounts are more likely to experience inactive interludes given the dynamic of their workflow (§ 4.3.5). Account activity includes but is not limited to the number of apps with which the account interacts per time unit, the duration of such interactions, and the number of other Google services (maps, gmail, drive, music, etc) to which it is subscribed. Additionally, our data and experiments reveal that some workers may even be posting only laptop-based reviews as all their reviews were written from devices of *unknown* models. Our study suggests that these workers are more likely to control sockpuppet accounts. This requires however future validation.

VP5: Monitor Review Feedback. An account should be able to upvote or downvote a review only if it has installed the respective app on at least one device. We verified that this is not currently enforced by Google Play. Fraud attribution (see below) can also be used to discount upvotes from accounts known to be controlled by the same ASO worker as the one that posted the review.

VP6: Verify App Install and Retention. We recommend developing protocols to verify that an app has been or is still installed on the device, e.g., before accepting a user review from that device. While remote attestation inspired solutions (e.g., [Jak18]) will not be secure without device TPMs, defeating such solutions will require significant investment from ASO workers.

VP7: Account Validation and Re-validation. The cellular provider used during account validation can also be used to detect inconsistencies with the claimed profile (e.g., location) of the user account. Further, several ASO workers mentioned using SIM cards of others to validate their accounts. Peer-opinion sites could ask users to re-verify their accounts at random sign-in times (e.g., veiled as "improved authentication security"), especially if their validating SIM cards have also been used for other accounts.

VP8: App Usage. Most ASO workers suggest that they use apps before reviewing them, and keep them installed after review for a while, to mimic genuine behaviors. However, we believe (but have not investigated) that features extracted from per-app waiting times, app interaction modes and times, and post-review behaviors, are different for honest vs. fraudulent accounts, and could be used to pinpoint sockpuppet and organic fraud accounts. For instance, it is suspicious if an app receives a good review soon after it was downloaded, has received little interaction, and is quickly uninstalled. Coupled with VP6, mandating wait times to post reviews will impact the number of apps that an ASO worker device can store, thus the number of apps that a worker can target at a time.

VP9: Mislead ASO Workers Through Fraud Attribution. SIM cards can also help attribute sockpuppet accounts to the ASO workers who control them, see e.g., [HRRC18]. Account-to-ASO worker attribution can be used to reduce worker ability to adjust to detection [SCM11]: to mislead ASO workers into believing that their actions are effective, peer-opinion sites could show removed fake positive reviews only to the accounts used to post them, the other accounts suspected of being controlled by the same worker, and the account of the app developer. This would force ASO workers to partition their account set into monitoring-only sets that cannot be used to post fraudulent reviews, and regular fraud-posting accounts.

VP10: Once a Cheater, Always a Cheater. Our qualitative and quantitative studies (§ 4.3.9) provide evidence that developers rehire ASO workers not only for the same app, but also for other apps that they develop. We recommend to monitor overlapping accounts that review sets of apps by the same developer, and redflag fraud developers early on.

4.5 Limitations

Recruitment Bias. We have not performed a complete exploration of the ASO worker universe, and cannot claim that our participants are a representative sample. Our recruitment process is biased, since we selected only candidates who (1) we could reach out to, (2) responded, (3) were English speakers, (4) were willing to participate after approving the consent form, and (5) claimed qualifying capabilities (i.e., control at least 100 accounts, have at least 1 year of ASO expertise and participated in at least 100 ASO jobs, § 4.2.1).

For instance, out of the 560 contacted workers, 72 replied to our invitation, 25 qualified, and 18 agreed to finally participate. Thus, other workers will likely have both fewer and more capabilities than the participants in our studies. However, from the answers and data that we collected, we reveal previously unknown ASO strategies, provide insights into previously proposed defenses that may be effective against them, and report Google defense vulnerabilities.

We leave for future work an investigation into the ability of deception and more substantial financial incentives, to increase the recruitment success rate and identify novel ASO strategies. We believe that our approach is a best effort in recruiting workers, without the use of deception.

Generalization of Results. We have used crowdsourcing sites such as Upwork, Fiverr, Zeerk, and Peopleperhour for years, and have found them to be reliable sources of ASO activities. In addition, we have also found and used, after being pointed out by multiple ASO worker contacts, large groups in Facebook, that specialize in ASO. However, we do not claim that we were able to contact most of the active ASO workers.

The participants in our studies also claimed expertise in fake reviews and ratings in Google Maps, Apple Store, Amazon, Facebook and Twitter, fake installs in Apple App Store, fake likes and followers in Facebook and Instagram, and influential tweets in Twitter.

However, we did not ask participants, questions about their strategies in other platforms. Thus, we do not claim that our findings apply to other sites or other types of ASO work.

Validation of Findings. Due to the sensitivity of the topic surveyed and data collected, we did not perform the quantitative and qualitative studies on the same participants. Our quantitative study is also performed only on a subset of the accounts controlled by 39 participants. We have corroborated multiple survey answers with quantitative measurements, and also manual verification by the authors. In § 4.3.17 we describe the process we used to validate the data collected in the quantitative study. However, several participant claims are difficult to validate (e.g., team organization, size and location, capabilities, interactions with employers, number of devices controlled, etc). The particular nature of our participants, makes any suspicion on these topics, legitimate.

4.6 Discussion of Reasons to Participate

We now discuss possible reasons for the recruited participants to be willing to talk about their ASO activities, given the potential for their participation to affect their livelihood and put them at legal risk.

Participant Openness. From our interaction with many ASO workers, not only the ones involved in these studies, we found that many report the need to frequently discuss their work and capabilities, e.g., to convince prospective employers that they have the expertise to do the work. While such disclosure of their strategies may only be required to convince the potential buyers, they also discuss them more generally with other people (often to find ways to improve them). Hence, the information that we obtained from our participants is not a piece of hidden information that we managed to get by any tricks; rather it is information that the participants share with others willingly. While this information is not written anywhere in formal text (and hence probably has not been discussed much in academia), it was clear from our interaction with our participants that this information was not secret in their social and professional sphere.

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, where most of our participants are from, 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 [Ban18a]. However, this law has mostly been applied to control the dissemination of politically motivated, unfounded information over social media (see [Fre18, Ban18b, AHG$^+$17], 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, Pakistan, and Vietnam. Hence, the job of our participants was not illegal or unethical according to their own law of the land.

Furthermore, we asked the qualitative study participants questions on the perceived legality of their work, "*Are there any legal challenges for you in this job? How fair do you think are those legal challenges?*". 11 participants answered. All participants said that they have never faced any kind of legal issues. 3 participants (P1, P3, P5) said that the only issue they have faced is that they can not openly advertise their black hat ASO expertise in crowdsourcing sites, since the sites would block their accounts. Instead, they advertise white hat ASO expertise. Further, 5 participants (P1, P2, P4, P7, P14) said they are posting reviews like real users, thus their activities are legal. For instance, P4 said *"This is the art of my approach and also what ensures my success. Since I am doing this in a proper way, there has been no legal issue so far"*. These responses confirm that our participants did not face any legal complexities imposed by their own government.

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 [IVD$^+$10]. 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 [PLT09].

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.

Affinity and Financial Incentives. The author who recruited and interviewed the participants, had cultural affinity with the participants, which allowed for rapport establishment. Further, the rate of \$5 for each 15 minutes of participation (i.e., \$20 per hour) is consistent with hourly rates advertised by ASO workers in crowdsourcing sites, exceeds the minimum US salary (\$7.25 per hour), and significantly exceeds the *average* hourly salary in e.g., Bangladesh (332 BDT $\approx$ \$3.93 [Sal19]), India (96 INR $\approx$ \$1.38 [Sal19]), at the time of writing.

Disclosure Despite Risks. It might be interesting to think why our participants shared their strategies with us when they knew that our study might result in developing counter-strategies. We did not ask them that question. However, from the conversation that we had with our participants and from the living experiences of two of our authors in Bangladesh, we can make the following educated guesses:

- Although we told participants that others could try to develop solutions to counter their strategies, from the tone of their responses it was often evident that they were not entirely convinced that this was a real threat for them. Participants also often declared their teams to have substantial capabilities including access to many devices, accounts, and hard-to-detect organic account holders. They also shared with us how their strategies changed when the companies came up with any countermeasures. These responses made us believe that our participants were not scared of the threat our study might impose upon their work.

- Our participants had expertise in many other types of jobs, including fake reviews and ratings in Google Maps, Apple Store, Amazon, Facebook and Twitter, fake installs in Google Play and Apple App Store, fake likes and followers in Facebook and Instagram, and influential tweets in Twitter. Furthermore, these ASO workers often also offer other kinds of digital services (including: designing a logo, responding to surveys, etc.), and their job is not entirely dependent on ASO work. Hence, we conjecture that the participants did not find the sharing of their ASO strategies for Google Play, to be a significant risk.

- Several of the revealed strategies may be open-secret in Bangladesh. In fact, while conducting this study, some of our contacts in Bangladesh who were not involved in this business confirmed several of these strategies, which they had heard from their social contacts. This shows that at least some of the information that we found from our participants was not secret among the ASO workers only, but was knowledge shared in the society. Our qualitative study allowed us to be the first to find and share this knowledge with the research community.

Understanding of Risks. While we did not explicitly test whether the participants clearly understood the risks involved in participating in our study (which would go beyond our study protocol), our experience of talking to our participants clearly reflected that they understood the risks quite well. In fact, several of our interview participants provided detailed explanations on their strategies to address such risks, i.e., to avoid detection (§ 4.3.5). They also explained to us how they come up with new strategies to handle new "challenges" that occur when sites change their rules.
Intentional Misleading. We have used data from a quantitative study, and also manual validation, to confirm several statements and strategies reported by our participants. While by association, we feel inclined to believe other participant statements (e.g., that are hard to verify remotely), given the particular nature of our participants, we agree that any suspicion is legitimate. Hence, we present the hard-to-verify findings, to be taken with all possible limitations.

CHAPTER 5

DEMYSTIFYING DEVICE USE IN APP SEARCH OPTIMIZATION

5.1 Introduction

Understanding the behaviors and strategies employed by app search optimization (ASO) workers to promote products in online app stores such as Google Play, is key to develop effective and appropriate detection and response solutions. This is because ASO workers have been shown to develop effective product promotion and detection-avoidance strategies [ZXL+18, RHR+19]. For instance, while some workers pool into teams and share resources (e.g., devices, accounts), others, e.g., *organic workers*, were shown to work alone using their personal devices and accounts [ZXL+18, RHR+19].

While existing fraud detection solutions can be effective against dedicated ASO workers, promotion campaigns that involve organic workers are harder to detect, as they are difficult to differentiate from grassroots movements. More specifically, the higher degree of independence of organic workers may help them evade standard fraud-detection signals, e.g., lockstep behaviors [SMJ+15, TZX+15, YKA16, XZ15b, JCB+14, CYYP14, SLK15, XZ14, XZ15a, LFW+17] and posting reviews in bursts [FML+13, HTS16, LFW+17, HSB+16a, BSLL+16, XZLW16, Xu13, GGF14, BXG+13, KCA17, LNJ+10, MVLG13, MKL+13, KCS18, LCNK17, YKA16, FLCS15, **?**, XWLY12, CYYP14].

In this thesis, we posit that, in the Google Play app market, the interaction of ASO workers with promoted apps is distinguishable from the regular, personal use of installed apps. For instance, we expect that promoted apps tend to be reviewed soon after being installed, and receive little use before being uninstalled. Figure 5.1 illustrates the interaction timelines for an app promoted by two distinct workers

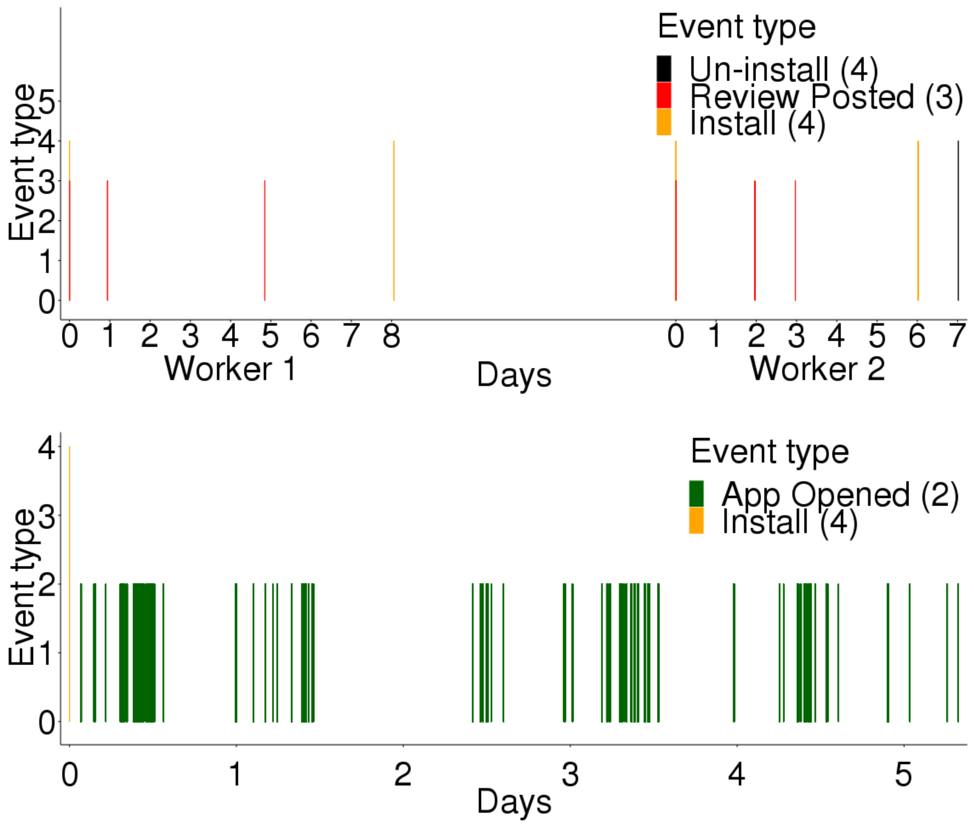


Figure 5.1: App-interaction timelines for two ASO workers (top) and one regular user (bottom). Worker timelines start with the app installation event (type 4 on y axis), followed by several review posting events across several days (type 3), with no interaction with the app. In contrast, the regular user timeline shows frequent interaction with the app in the form of placing the app in the foreground (type 2 event), but no review even after 5 days of monitoring.

(top) and a normal timeline from a regular user (bottom). Further, we posit that organic workers differ in their use of devices, when compared to workers dedicated to app promotion, and regular mobile device users.

To investigate these hypotheses, we need, however, the data that is only available with the app market developers who have a default app installed on the devices of their users, e.g., the Play Store app. To address this limitation and open access to the research community, in this thesis, we introduce RacketStore, a framework to collect data from, and compare the app and device use of ASO workers and regular users. The RacketStore app periodically collects detailed snapshots of device and app usage from the devices where it is installed, and reports them to a backend server, where they are aggregated with corresponging data collected from the Google Play app store and the VirusTotal [Vir20] site.

We have recruited participants from Facebook groups that specialize in ASO work, and through ads shown in Instagram. RacketStore has received 943 installs from 803 unique devices: 580 devices controlled by ASO workers and 223 devices of regular Android users. We found that 31 of the worker-controlled devices belong to a new category of *exchange workers*, who magnify their capabilities by swapping ASO jobs with peers, without payment. We have captured 58,362,249 snapshots from the participating devices, including the 12,341 apps installed and in-use on the participant devices, and their 110,511,637 reviews from Google Play. We identified a new vulnerability in the Google ecosystem that allowed us to map Gmail addresses to Google IDs, and thus retrieve the reviews written by accounts registered on participant devices.

Together, the 580 worker-controlled devices 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.

Further, we have conducted interviews with 13 of the participants who installed RacketStore, to advance our understanding of their observed behaviors, including their perception of privacy, malware and legal challenges associated with their work.

We use extracted insights to introduce features that model the usage of an app on a device, and train learning algorithms to distinguish suspicious from personal patterns of app interaction. We show that supervised models distinguish between apps suspected of promotion and apps used for personal purposes, with an F1-measure of 99%. Such a classifier can be used to disentangle personal from suspicious activities of organic workers and provide an appropriate response to detected fraud.

We further introduce features that model the general use of a device by its owner and use semi-supervised learning to cluster devices based on the similarity of their behaviors. Out of 166 worker devices that reported sufficient data for evaluation, we found (1) 107 to be controlled by organic workers, (2) 19 to be controlled by *power workers*, i.e., had more than 1,100 Gmail accounts registered, had written more than 44,400 reviews, and had each used in a suspicious manner at least 89% of their installed apps, and (3) 40 to have such low levels of promotion activities to be mislabeled as regular devices.

In summary, we introduce the following contributions:

- **RacketStore**. Introduce a framework to study the interaction of users with their Android devices and the apps that they install, that proved compatible with 298 device models from 28 Android manufacturers [§ 5.2].
- **Quantitative and Qualitative Studies**. Present empirical data from RacketStore deployment on 803 unique devices controlled by ASO workers and regular users [§ 5.3.1], and insights from interviews with 13 participants [§ 5.3.2].
- **Fraud Detector and Classifier**. Develop a supervised classifier to detect apps installed only to be promoted, and an unsupervised classifier to differentiate between ASO-dedicated devices, organic worker devices, and devices used solely for personal activities [§ 5.4].
- **App and Device Use Dataset**. Build datasets of app and device usage, integrated with Google Play reviews and VirusTotal analysis, for devices controlled by regular Android device users and a variety of ASO workers, including a newly identified *exchange* worker type [§ 5.5].

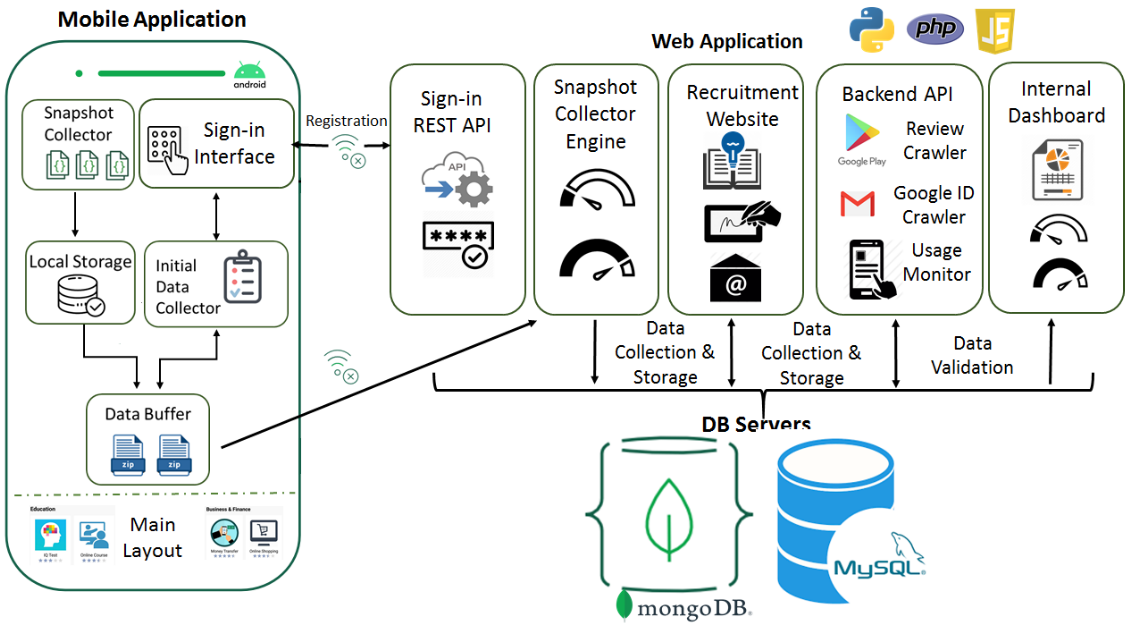


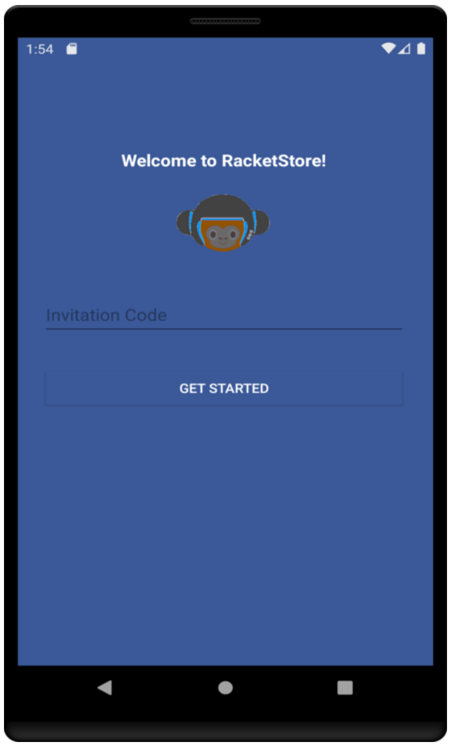
Figure 5.2: RacketStore architecture consists of a mobile app installed by participants and a back-end server that collects and aggregates snapshots reported by deployed apps.

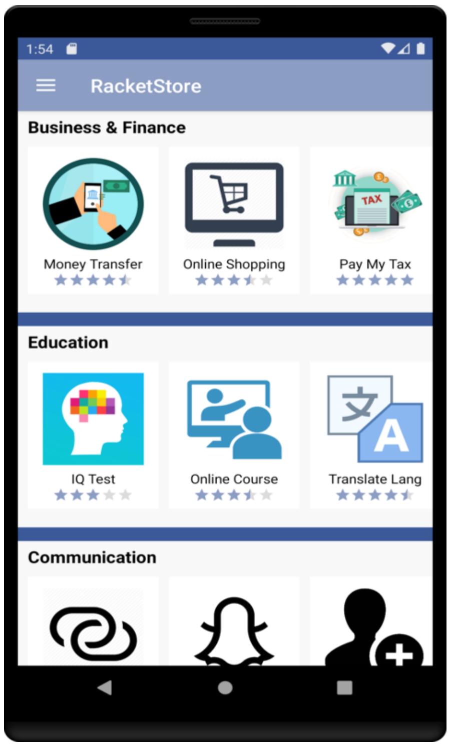
5.2 Data Collection Infrastructure

We have developed RacketStore, a framework to collect information from the Android devices of ASO workers and regular users. Figure 5.2 illustrates the RacketStore architecture, that consists of the RacketStore mobile application, a web application that handles different aspects of the data collection and validation process, and the database servers where data is stored. In the following we detail each component.

5.2.1 RacketStore App

We have developed the RacketStore app in Android to help us investigate fraudulent and honest behaviors of Google services users. RacketStore acts as a portal to a honeypot app market. Upon first start-up, to comply with Google anti-abuse policy





(a) (b)

Figure 5.3: RacketStore mobile app's screenshots. (a) Registration screen to enter invitation code sent to participants. (b) Main layout of the app, showing different app categories.

[ant20], RacketStore first asks for explicit consent of our privacy policy, then shows an in-app disclosure of the data being collected.

Sign-in Interface. RacketStore's sign-in interface (see Figure 5.3(a)) asks the participant to enter a unique *participant ID*, a 6-digit code, that we send upon recruitment (§ 5.3.1) through an offline channel, e.g., e-mail, 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 workers and honest users. Only after the user has accepted to participate in our study and has agreed to the data collection process, are they given the passcode. The app does not collect any information if the user has not entered the 6-digit passcode. Upon sign-in, RacketStore generates the *install ID*, a 10-digit random identifier.

Initial Data Collector. The initial data collector module (Figure 5.2), prompts the user with several questions on their review posting experience: (1) Did you ever write a review for an app in Google Play?, (2) Have you ever been paid to write a review in Google Play?, (3) Have you ever written paid reviews from this device?, (4) How many user accounts do you control in Google Play?, (5) How many mobile devices do you own or can access?, and (6) For how many days can you keep our app installed?.

The initial data collector module retrieves the user answers, and also (1) the list of apps installed on the device, (2) device information including Android API version, device model, manufacturer, and a unique device identifier (Android ID) [and17], and (3) whether the device is an emulator, using Flutter built-in methods and legacy detection systems [flu20].

Snapshot Collector. The snapshot collector module periodically collects information with two levels of granularity, slow and fast, described next. The slow snapshot collector is triggered by an alarm every 2 minutes, to extract the following information:

- **Identifiers**. Install ID, participant ID, and Android ID.
- **Registered accounts**. The accounts registered on the device across different services.
- **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 the application.
- **Device status**. Internal and external device memory, save mode status (on/off), sim card status (on/off).

The “fast” snapshot collector module further captures the following data, with a 1s granularity:

- **Identifiers**. Install ID and participant ID.
- **Active app**. The app currently running on the foreground.
- **Device status**. The screen status (on/off) and battery level.
- **App install/uninstall events**. Report deltas between the current and previously reported sets of installed apps. For each installed app we use the `PackageManager` API [pma20] to report (1) install time, (2) last update time and therefore update events, (3) required permissions. Further, we report the MD5 hash of the APK file of the app.

RacketStore declares two broadcast receivers to detect turn-off events and to launch the app upon reboot.

Data Buffer Module: Processing Snapshots. The *data buffer* module (see Figure 5.2) processes both types of snapshots leveraging the device’s *local storage*. Every time a snapshot is taken, the data is written to different accumulating files depending on the snapshot’s type. Every time the accumulation file grows to 8 KB and 100 KB in size for the slow and fast types of snapshots respectively, they are compressed and a new accumulation file is created where the following snapshots continue to be recorded. We selected these threshold values based on the observed battery and bandwidth consumption, which we sought to minimize. Then, the alarm that fires every 2 minutes looks for any existing compressed files in the mobile app’s directory and sends them to our server. Upon file reception, the server returns the md5 hash of the received data in order for the mobile app to validate the transfer with its own hash calculation. If both hashes are equal, the app deletes the file from disk. This transfer mechanism allows for resilient communication.

RacketStore requires participants to grant two permissions that we have included in the `AndroidManifest.xml` file: `PACKAGE_USAGE_STATS` and `GET_ACCOUNTS` [and]. However, declaring the first permission implies intention to use the API and the user still needs to grant the permission through the Settings application which we open upon sign-in. On the other hand, the `GET_ACCOUNTS` permission is shown as a regular Android permission with a pop up at run time and it allows access to the list of accounts on the device.

5.2.2 Web Application

We have built a web application that supports RacketStore on the server side. We implemented the *Sign-in REST API* that 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 *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 (§ 5.3.1). The *Backend API* is comprised of three subsystems: (1) a review crawler that scraps reviews from Google Play given an app name and a device model (see § 5.5), (2) a Google ID crawler that maps Gmail accounts to a unique Google ID (see § 5.5) and (3) a usage monitor that triggers when no activity is detected from the app installs, i.e., sends email reminders to participants whose devices have not sent data to our servers in the last 12 hours.

Finally, we developed an internal dashboard 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.

5.3 Methods

Our study was conducted in two steps. The first step of our study involved the deployment of RacketStore with a group composed of both ASO workers and regular Google Play users, and understanding their usage patterns through quantitative analysis. In the second step, we conducted interviews with 13 ASO workers and regular users of Google Play to further advance our understanding of their usage.

5.3.1 Deployment of RacketStore

Recruitment of ASO Admins. We recruited ASO admins from the Facebook groups that we found to be dedicated to product promotion. Specifically, we posted calls to recruit participants who would be able to install and provide reviews on the 7 Facebook groups dedicated to Google Play app promotion. Many group members commented on our posts, expressed their interest, and asked us communicate with them over the Facebook inbox for further details. We contacted 49 such members over the Facebook inbox. We shared with them the details of the recruitment instructions (see section 7). We asked them to reply if they were interested, and also provide the following information: (a) whether they were ever hired to write reviews for Google Play apps before, (b) if they were currently involved in app search optimization jobs, (c) briefly describe the nature of those jobs, and (d) how many accounts they had control over, and how many mobile devices they owned. A total of 24 workers agreed to participate providing those pieces of information. We sent them the participant ID (§ 5.2.1) alongside a YouTube video that explained how to sign up for RacketStore. 17 workers ended up taking part in our studies. Each of these 17 participants claimed to control large groups of ASO workers and to be able to install RacketStore on at least 100 devices.

Recruitment of Exchange Workers. Nine out of the 16 Facebook groups that we identified, offered exchange reviews across different platforms including Google Play, Google Maps, Yelp, Amazon, and the App Store. The posters asked members to review their products in exchange of them providing reviews for those members' products. We responded to 97 exchange reviewers through their posts and subsequently over Facebook inbox, and asked them to participate in our study. A total of 38 people agreed to participate and installed RacketStore. In exchange, we had to provide feedback for their products. We installed the apps that they asked us to review, used them, and provided an honest review that represented our actual experiences with the apps. We also mentioned in the reviews that those were sponsored reviews.

The 17 ASO admins are from Bangladesh (9), Pakistan (5) and India (3), and the 38 exchange workers are from Bangladesh (19), India (8), Pakistan (8), Morocco (2) and Tunisia (1).

Recruitment of Regular Users. Further, we have recruited regular Android device users through commercial advertisements on Instagram (see Figure 5.4). We chose Instagram in order to avoid our ads being seen by the members of the above Facebook groups, leading to us again recruiting ASO workers. Our ads point to a landing page (see Figure 5.5(a)) that explained the purpose of our study. We posted the ads intermittently between December 17, 2019 and April 15, 2020, spending a total of 79.23 USD. Since cultural differences could affect patterns in mobile device use [WK05, cul12], 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

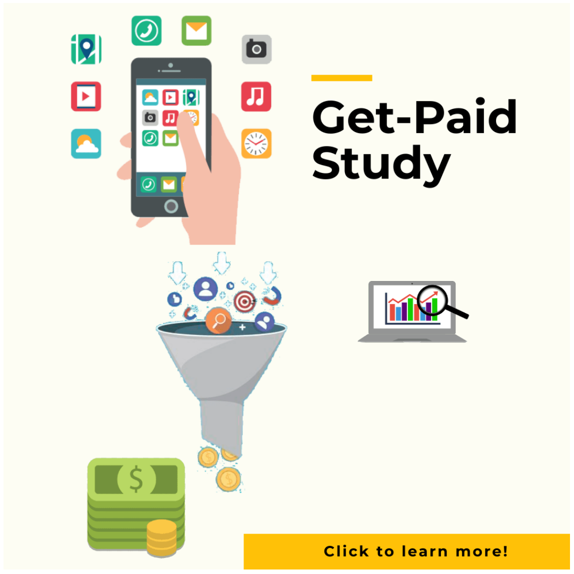
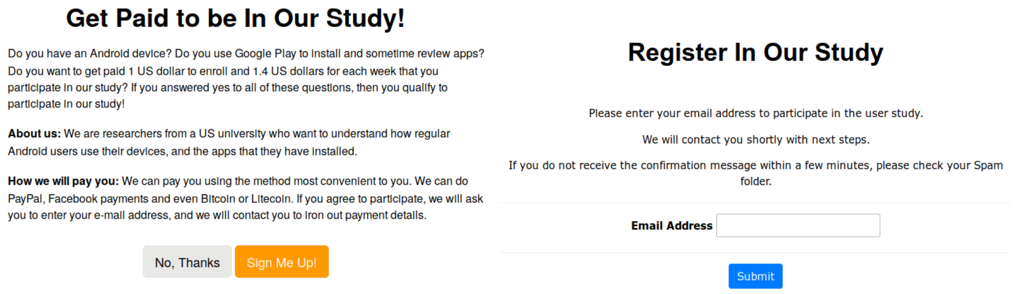


Figure 5.4: Ad shown to audience on Instagram Feed, Explore, and Stories. Upon clicking, users are sent to a website where the study is explained.

who are from Bangladesh (Chittagong, Dhaka, Rajshahi), India (Bangalore, Mumbai, New Delhi), Myanmar (Yangon), Pakistan (Islamabad), and Vietnam (Hanoi), 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, 61,748 users were reached by our ads that were shown a total of 136,022 times. 2,471 of these users landed on our web page after clicking on the Instagram ads. The landing page introduced our study, explained the payment method (i.e., Paypal, Bitcoin or Litecoin), and provided them with the option to either withdraw or sign up by acknowledging and agreeing with the terms and conditions that included the consent form explaining in detail the information that we would collect from their phones. If the user consented, we asked them to register in the study by submitting their email address (see Figure 5.5(b)). A total of 614 participants consented and sent us their e-mail addresses. We then sent an automatic confirmation email along with the Google Play link to download the RacketStore app (§ 5.2) and a six-digit unique participant ID that the participant



Get Paid to be In Our Study!

Do you have an Android device? Do you use Google Play to install and sometime review apps? Do you want to get paid 1 US dollar to enroll and 1.4 US dollars for each week that you participate in our study? If you answered yes to all of these questions, then you qualify to participate in our study!

About us: We are researchers from a US university who want to understand how regular Android users use their devices, and the apps that they have installed.

How we will pay you: We can pay you using the method most convenient to you. We can do PayPal, Facebook payments and even Bitcoin or Litecoin. If you agree to participate, we will ask you to enter your e-mail address, and we will contact you to iron out payment details.

Register In Our Study

Please enter your email address to participate in the user study.

We will contact you shortly with next steps.

If you do not receive the confirmation message within a few minutes, please check your Spam folder.

Email Address

(a) (b)

Figure 5.5: 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, and we then contact them with next steps.

would need to type-in to the app. RacketStore received 233 installs from the 614 consenting participants.

Participant Payments. We paid each participant who installed RacketStore on a per device basis: 1 USD to install the app and 0.2 USD 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. Subsequently, the participant is not required to perform any other activities, e.g., open and interact with the app.

In summary, we recruited (1) 17 ASO administrators who provided 672 RacketStore installs (M = 33.47, SD = 44.66, Max = 150), (2) 38 individual exchange workers, and (3) 233 regular users. We note that for the 569 devices provided by the ASO admins we do not have information about the type of worker who controls them, e.g., organic, team-based, etc.

5.3.2 Interviews

To deepen our understanding of our findings, we conducted a small scale follow-up interview study. We have developed semi-structured interview questionnaires for both ASO workers and regular users (see section 7). For the workers, our goal was to understand their perception of various problems that we discovered. On the other hand, for regular users, an additional goal was to confirm the labels, i.e., to make sure that they were indeed the kind of users who never provided paid reviews. For this, we reached back to our participants, and asked them if they would be interested in the follow-up interviews.

A total of five admin workers (two from Pakistan, two from Bangladesh, one from India), five organic workers (four from India, one from Bangladesh) and three regular users (two from India, one from Pakistan) agreed and participated in the interview study. We note that this interview study was conducted between February, 2020 and April, 2020, when the COVID-19 pandemic was unfolding throughout the world, and it was often difficult to publicize our call among the pandemic related posts. Also, the pandemic might have affected the life and work of our participants. These issues might explain the low participation in our interview study. However, based on the rigor of qualitative studies, instead of quantity, we focus on the depth of our data, that revealed relevant insights for our study.

The interviews took 15-30 minutes, were conducted over Whatsapp and Skype, in English, Bengali, and partial Hindi as appropriate by the members of our research team. The interviews were audio recorded with the permission of the participants. The interview scripts were later anonymized, translated, and transcribed by the members of our research team. We used Grounded Theory methods [CB07] to develop our understanding around the perception, usage, and challenges for paid reviews on Google Play.

Participant Payments. We paid interview participants 5 USD for each 15 minutes of their time. Thus, our payments are consistent with hourly rates advertised by ASO workers in crowdsourcing sites, exceed the minimum US salary (7.25 USD per hour) and the *average* hourly salary in e.g., Bangladesh (359 BDT $\approx$ 4.23

USD [Sal19]), India (201 INR $\approx$ 2.66 USD [Sal19]) and Pakistan (775 PKR $\approx$ 4.80 USD [Sal19]) at the time of writing.

In the rest of the chapter, we use A1,..,A5 to denote the five ASO admins who participated in our interviews, O1,..,O5 to denote the five organic workers, and R1, R2, R3 to denote the regular users.

5.3.3 Ethical Considerations

The full study procedure was examined and approved by the Institutional Review Board (IRB) of a major North American university. For the purpose of clarification, we highlight here a few ethical considerations that were essential for conducing this research:

Honesty and Consent. We did not use any deception in our study. Before collecting data, we explained to the participants the identity of the researchers, the research objectives, the data that we wanted to collect, and the potential impacts on participants. Please check the recruitment message and introductory script in 7. Participation in this study was completely voluntary.

Data Protection. We used GDPR [Par16] recommended pseudonymisation for data processing and statistics, and other generally accepted good practices for privacy preservation. After data collection, we deleted all device-to-identity links and only generated aggregated statistics that allowed us to validate our assumptions. We also responsibly disclosed the Google vulnerability that allowed us to map e-mail accounts to Google IDs, through Google's Vulnerability Reward Program [GVR]. No PII of our participants was disclosed outside the research team.

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. However, previous work by Rahman et al. [RHR$^+$19] explains why studying the ASO workers does not impact their livelihood. Nonetheless, we disclosed and explained this possibility to our participants, and we did not hear any concern from any of them.

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 [Ban18a] and has never applied to ASO work. A similar situation is also present in many other countries in the Global South including India, Pakistan, and Vietnam. Hence, the job of our participants was not illegal or unethical according to their own law of the land.

We also asked ASO admins 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, ASO admin A2 said that "*What we're doing is a legal and right job. So no need to be afraid*".

5.4 App and Device Usage Features

We posit that the data snapshots collected by RacketStore can be used to (1) detect apps that are installed in order to be promoted. and (2) to detect differences in behavior between devices controlled by different types of workers.

App Usage Features. To investigate the first hypothesis, we use data collected from a device D to extract features that define how the user uses an app A installed on D. In the following, let $I_D = [T_I, T_U]$ denote the install interval of RacketStore on device D, i.e., defined by the first install and last uninstall times.

- **# Accounts Review**. Number of accounts registered on device D from which app A was reviewed in Google Play (1 feature). Also, the number of accounts registered on D that reviewed app A, (1) before RacketStore was installed, (2) while it was installed, i.e., during I_D and (3) after RacketStore was uninstalled (3 features).
- **Install to Review Time (App)**. Length of interval between the time when A was installed on D and when it was reviewed by an account registered on D (3 features: mean, median and standard deviation over all the accounts on D that reviewed A).
- **Install to Review Time (User)**. Length of interval between the time when an app A was installed and when it was reviewed by an account registered on D, for all the apps installed and that have been reviewed from the device (3 features: mean, median, and standard deviation).
- **Inter-review times**. Time difference between all consecutive reviews posted for app A from accounts registered on device D (3 features: mean, median, and standard deviation).
- **Opened**. Whether app A was opened on multiple days on device D (1 feature, boolean).
- **Daily App Usage**. Number of snapshots per day when the app was the on-screen app (3 features: mean, median, std dev, during I_D).
- **Daily Snaps**. Number of snapshots captured per day from device (3 features: mean, median, std dev, over interval I_D).
- **Inner Retention**. App A install duration (1 feature computed as percentage of entire I_D interval).

- **Persistence Retention**. Whether the app was still installed after uninstalling RacketStore (1 feature, boolean).
- **Previously Installed**. Whether the app was installed before the installation of RacketStore (1 feature, boolean).
- **Normal/Dangerous Permissions**. Number of normal and dangerous permissions requested (2 features).
- **VT Flags**. Number of flags raised by VirusTotal AV tools that categorize the app as malicious, suspicious, or undetected (3 features).
- **# Installs/Uninstalls**. The number of times the app was installed and uninstalled during I_D (2 features).
- **Granted/Denied Permission**. The number of permissions requested by the app that have been granted and denied by the user (2 features).
- **App Size**. Average size of the app computed across all its versions found on VirusTotal (1 feature).

Device Usage Features. To evaluate the second hypothesis, we introduce several features that model the use of a device D: (1) *app suspiciousness*, i.e, the number of apps on D whose usage was flagged as suspicious by the above app usage classifier, divided by the total number of apps installed on D, (2) the number of Gmail accounts registered on D, (3) the average number of reviews per account written from those accounts, and (4) the total number of reviews written from accounts on D.

5.5 Data

We have collected data between October 2019 and April 2020 from 943 devices that installed RacketStore.

Snapshot Fingerprinting and Coalescing. To properly analyze the data collected from participating devices, we needed to map each captured device snapshot to a single device. As mentioned in § 5.2, 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 admins, with different assigned participant ids, who shared some devices. This can occur for instance if (1) the workers are employed by the same organization, thus have access to a common set of devices, or (2) the admins have access to sets of workers that overlap. 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 capture device IMEI since it requires an additional dangerous permission which we wanted to avoid, and it only applies to cellphones.

To address this problem, we used the following process to fingerprint snapshots. Specifically, we first grouped all captured 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. We found that 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 for registered accounts that exceeds 0.53, had low similarity for installed apps.

After this process, we had 803 unique devices: 549 devices controlled by ASO workers, 31 devices controlled by exchange reviewers, and 223 devices controlled by regular participants recruited through Instagram ads (§ 5.3.1). At the completion of the study, we had a total of 592,045 slow snapshots and 57,770,204 fast snapshots (§ 5.2.1). RacketStore was compatible with 298 unique device models from 28 Android manufacturers (top 5 most popular: Samsung, Huawei, Oppo, Xiaomi, Vivo).

Google Play Review Dataset and Google Vulnerability. The review crawler component of the Backend API of RacketStore's web app (see Figure 5.2 and $ 5.2.2) 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. We collected a total of 110,511,637 reviews from a total of 12,341 apps. Each review includes metadata, e.g., the user's Google ID, the review timestamp (with 1s granularity) and star rating.

To identify the reviews written by accounts registered on participant devices, we needed to map e-mail accounts to Google IDs. This is done in the Google ID crawler component of the Backend API in RacketStore's web app, see Figure 5.2 and § 5.2.2. Google Play protects reviewer identities by not publishing direct contact information (such as e-mail address) on its website [gpla]. We have discovered however that the Gmail service can be exploited to reveal the IDs behind Google Play reviews. Explicitly, we found that responses of Gmail's email search functionality embed the same Google ID previously found in the now defunct Google Plus. As in the case of Google Plus, this ID is left un-rendered by the browser and thus difficult to detect without careful inspection of the webpage code. Moreover, the removal of certain email search parameters allowed for even more information disclosure. Specifically, we performed email queries that removed all the `requestMask.includeContainer` family of parameters. The answers to these requests resulted in additional user sensitive information leakage such as email classification, image url, as well as the account's obfuscated GAIA (Google Accounts and ID Administration) ID [gai], "in-app reachability" and age range.

VirusTotal Report Dataset. We used the VirusTotal research license reports [Tot12] to analyze the presence of malware on the participant devices. We used the *snapshot collector module* (§5.2.1) to collect 18,079 distinct hashes corresponding to 9,911 unique mobile app identifiers installed in 713 participant devices: 511 devices of workers, 164 devices ofregular users and 38 devices ofworkers. We collected reports for these hashes in VirusTotal; 12,431 hashes were available in VirusTotal.

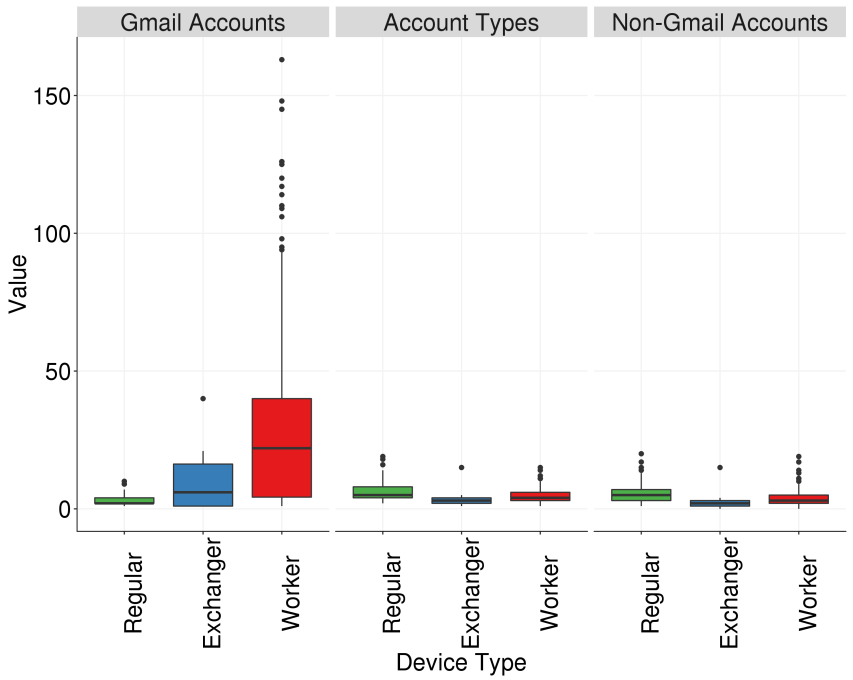


Figure 5.6: Comparison of number of e-mail accounts registered on devices controlled by worker, exchange worker, and regular participants. Worker devices tend to have more Gmail accounts, but fewer account types and non-Gmail accounts than regular devices.

5.6 Findings

We first report findings from the quantitative studies and interviews, with a focus on providing an intuition behind the features with potential to classify the usage of apps and devices. We then report the performance achieved by the app and device usage classifiers.

5.6.1 Registered Accounts

ASO workers are known to control or claim to control, many sockpuppet accounts which they use to post paid activities. Three of the five ASO admins that we interviewed (A1, A2, A4), claimed to only have 2-4 Google accounts; they justify it by the fact that their worker teams are responsible for posting reviews thus need more accounts. A5 claimed to control 10 accounts. However, A3 claimed to control 50 Gmail accounts, which he explained by the fact that he used to be a review-posting worker before becoming an admin. All the ASO admins claimed not to share their accounts with their workers or other admins.

Similarly, three of the five organic workers also claimed to control 1-2 accounts. For instance, O4 said "*I have two accounts. One account is mine another is my mom's.*" However, O3 claimed to control 30 accounts and another claimed to have "*many accounts*". In contrast, two of the regular participants (R1 and R3) claimed to only have one Gmail account, while R2 claimed to control 3 accounts.

We asked interviewed workers if they knew of a limit on the number of accounts that one can register on their devices. Four organic workers and two ASO admins said that they were not aware of any such a limit. One ASO admin believed the limit to be 15-20 accounts, while another admin mentioned that they knew people who had logged into 50 Gmail accounts from a single device.

Figure 5.6.1 (left) compares the number of Gmail accounts registered on 145 regular, 378 worker and 12 exchange worker devices that have reported such information. Not all devices reported it, due to not receiving permissions or enough snapshots. We observe that 13 worker devices have more than 100 Gmail accounts registered, with a maximum of 163 accounts per device. This is in contrast to regular devices that have a maximum of 10 accounts registered. Worker devices have an average of 28.87 accounts registered per device (M = 22, SD = 29.59, max = 163), which is higher than for exchange devices and regular devices.

Further, regular devices have more diversity on the types of services for which they have registered accounts, especially when compared against exchange workers, see Figure 5.6.1 (center). On average, regular devices have registered accounts for 6 services (max = 19), mostly for different social networks (Facebook, What-

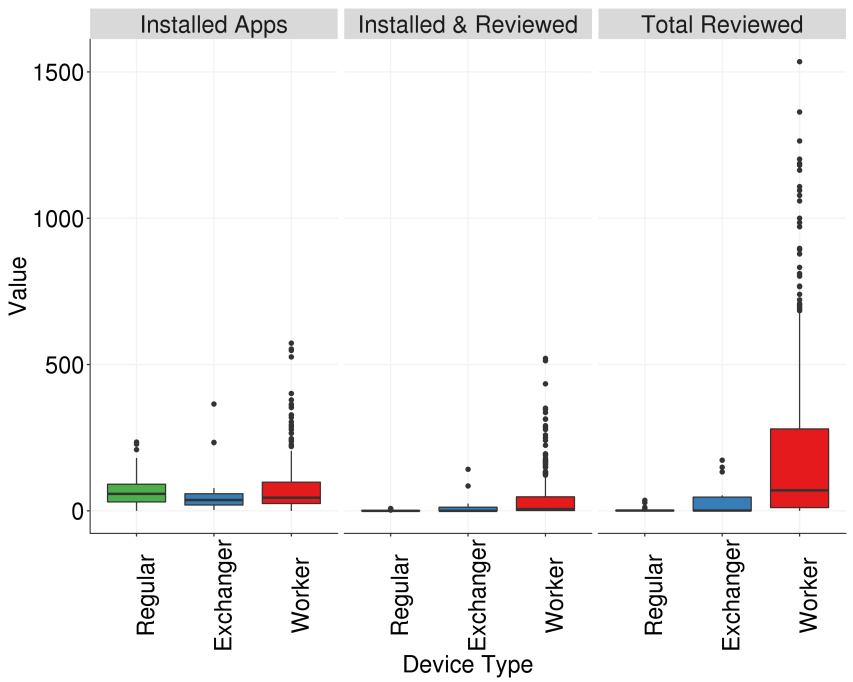


Figure 5.7: Number of installed apps, installed and reviewed, and total number of reviewed apps across all device types.

sApp, Telegram, etc). In contrast, worker and exchange worker devices have accounts mainly for Google services and other services useful for ASO work, e.g., dualspace.daemon, freelancer.

5.6.2 Installed Apps

Apps Installed and Reviewed. Figure 5.6.2 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, 381 worker and 19 exchange worker devices that reported these data. We first observe that each category of participants have a comparable number of apps currently installed. On average, we observe 65.45, 77.56 and 70.73 apps installed on regular, worker and exchanger worker devices respectively. This observation is

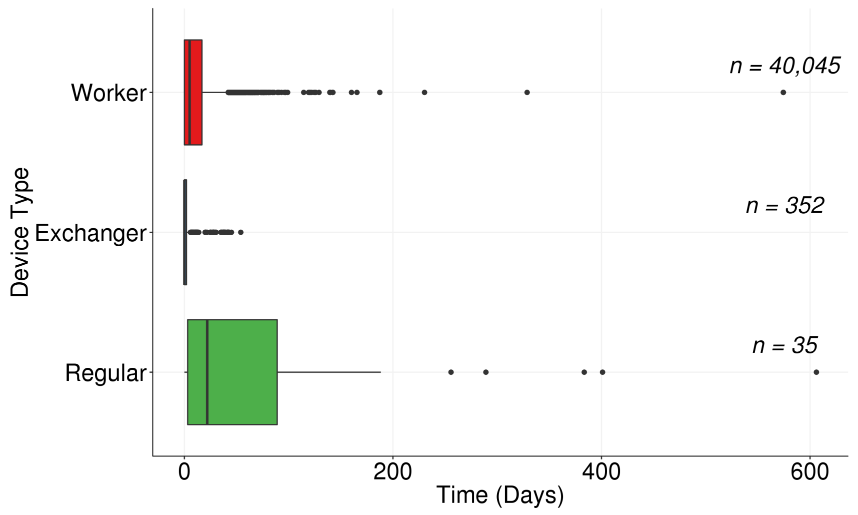


Figure 5.8: Distribution of time between app install and app review, for regular, exchange and worker devices. Each point is one review. Unlike regular users, worker-controlled accounts post many more reviews and tend to do it soon after installation.

expected as the number of installations is limited by the device's resource limitations across all participant categories. However, we observe that posting reviews does not follow this trend. On average, worker and exchange worker devices have posted reviews for 40.51 and 15.94 of the apps that they have currently installed. In contrast, on average, regular users only retain installation of 0.7 apps for which they have written reviews. If we include reviews for apps that are currently un-installed, the contrast is even more dramatic. On average, a worker and exchange worker device is responsible for a total of 208.91 and 34.26 reviews respectively. However, on average, regular user devices have posted only 1.91 reviews. For the case of worker devices, we have observed 11 devices responsible for more than 1000 total reviews. In contrast, the maximum number of total reviews generated by a regular devices is only 36.

Install-to-Review Time. Figure 5.6.2 shows the distribution of the time between app install and app review, for regular, exchange 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. First, we observe the difference in the number of reviews posted from accounts registered on these devices for apps that provided an installation time: accounts on regular devices only wrote 35 reviews, while those on worker devices posted 40,045 reviews.

Further, we observe that unlike regular users, workers (including exchange workers) tend to review apps much sooner after installation. In the case of workers, we observe an average of 10.7 days of waiting time between installation and review (M = 5.02 days, SD = 29.92 days, max = 3015.09 days). We have observed 4 cases of reviews posted after more than 1,000 days from 2 workers and for 2 apps (Facebook and Easypaisa), and only 25 cases with waiting times bigger than 100 days. These rare cases of prolonged waiting times are expected of apps used for personal purposes that have been pre-installed in the device. Similarly, exchange workers wait on average only 5.06 days to post a review (M = 466.56 sec, SD = 11.22 days, max = 54.07 days) which is consistent with expected a ASO activity. In contrast, regular users wait for 92.1 days to post a review on average (M=26.89 days, SD=146.36 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 inconsistent with paid promotion services.

App Permissions.

We studied the distribution of permission requirements for unique apps found on each participant categories. Figure 5.6.2 shows the number of dangerous permissions versus the total number of permissions for each app found exclusively on regular, worker and exchange 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.

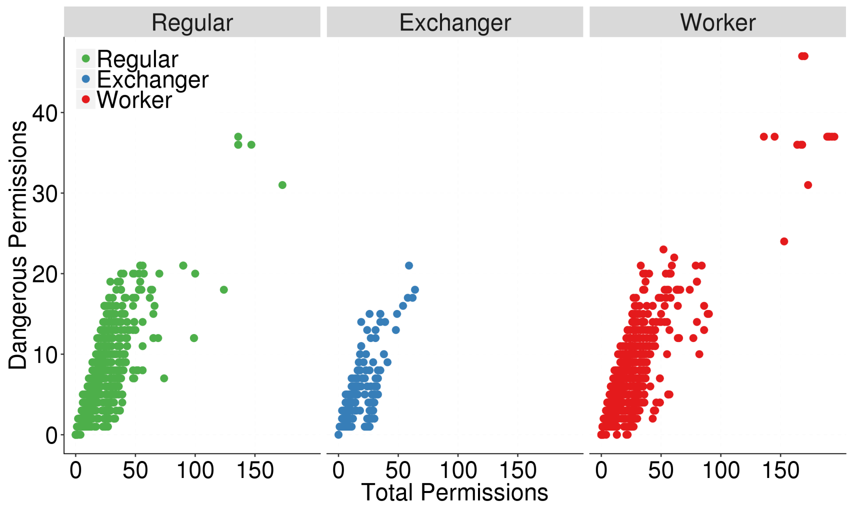


Figure 5.9: Comparison of exclusive app permissions for regular, worker and exchanger participants. Worker devices host apps with the largest ratio of dangerous versus total number of permissions.

All five organic workers and ASO admins that we interviewed, claimed to grant all permissions requested by the apps that they install. However, A1 claimed selective granting, i.e.,

"*Permissions are given based on the client request. If client does not ask, we do not give all permissions*".

Three of the organic workers and one ASO admin said there are permissions that they grant grudgingly, e.g., O4 who claimed "*I don't like location permission because it violates my privacy*", while O5 (and A4) said "*I don't like the ones that are related to personal data*". In contrast, two of the regular participants (R2 and R3) claimed not to grant all requested permissions. R2 claimed that he avoids granting location permissions, while R3 mentioned contact, images and phone storage permissions.

Third-Party App Stores. We observed that participant devices had apps installed that were not available in Google Play. We conjecture that either Google has removed those apps, they were removed by their developers, or the apps were installed from third-party app stores. We asked interview participants if they install apps from other app stores. All three regular participants, four of the five ASO admins, and one organic workers said that they only install apps from Google Play. A5 however said that they have installed apps from third-party stores, i.e.,

"*The client gives us a link, we go there and install that app*".

When asked why the app was not in Google Play, A5 claimed to just follow orders: "*App owner didn't publish there. I will do as the client says.*"

Three organic workers claimed to install apps also from other app stores, and to do it only for personal reasons, e.g., to play games (Dream11, O5), avoid subscription fees (Netflix, Hotstar, O1). O1 and O4 conjectured that Google does not host such apps because they violate Google's policy, and O5 because they are not secure. For instance, O1 said:

"*I use a modded* [1] *version of the apps that are not in the Google Play Store. You do not have to open an account to use these. Maybe Google Play Store's policy prohibits the use of such apps. 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.* "

5.6.3 Malware

Of the 12,431 unique hashes of apks of apps for which we collected VirusTotal reports (§ 5.5), 177 were flagged malicious by more than one VirusTotal AV tool. We found at least one of these flagged apps in 183 unique devices: 119 devices controlled by workers organized by 11 admins, 61 devices of 59 regular users and 3 devices of

[1] A *mod apk* 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, see `https://www.quora.com/What-is-a-modded-APK`.

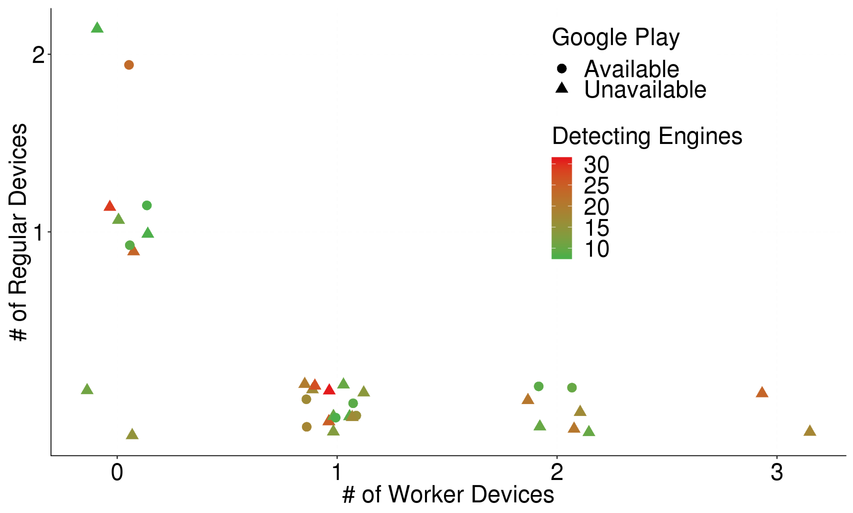


Figure 5.10: Comparison of malware occurrence in regular versus worker devices. Each point corresponds to a unique app apk hash, that raised at least 7 flags in VirusTotal. Worker devices host more unique malware which tends to be present on more devices than for regular users. Google has removed most malicious apps from the store, however they are still installed in worker and regular devices.

3 exchange workers. Thus, admins seem to be more likely to have at least one of their worker devices infected: 11/12 (91.6%) vs. 59/156 (37.8%). However, we find a 23.3% infection rate among worker devices versus a 37.2% infection rate among regular user devices.

This seems to suggest that worker devices tend to have safer behaviors, perhaps due to keeping apps installed for shorter intervals than regular users (see Appendix § 5.6.5). In contrast, regular users have been shown to be at a disadvantage when facing and judging malicious apps [BCI$^+$15].

To study the infection degree of each device, we compared the occurrence of the most malicious malware samples (flagged by more than 7 VirusTotal engines) in regular user devices versus worker devices. Figure 5.6.3 shows that malicious samples are more likely to appear in several worker devices when compared to regular users. Moreover, most malicious samples have also been removed from Google Play, which further validates their malicious classification.

Further, we found 70 unique mobile app identifiers with at least one malicious VirusTotal engine flag, that received at least one review from our participants: 62 of these apps were reviewed by workers, 2 apps reviewed by exchange workers, and 9 apps reviewed by regular users

Anti Virus (AV) Apps. We have identified 250 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 789 participant devices that sent at least one snapshot. We found only 19 devices that installed 15 AV apps: 7 worker devices working with 4 admins, 7 on regular user devices, 1 on an exchanger device and 4 unknown (i.e., either Google testing our infrastructure or some participants who managed to bypass our invitation code).

Participant Feedback. We asked interview 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 ASO admins (A2, A3) were not concerned about malware or privacy leaks and did not have AV apps installed. A3 for instance said

"*I am confident on the ability of my phone to prevent any mishap*".

A1, A4 and A5 and their worker teams are concerned about malware. A4 claimed however not to use AV apps. A1 and A5 claimed to use AV apps. A5 said it is the device's built-in AV app. A1 said to feel safe. Among the organic worker participants, only O4 and O5 said to be concerned about malware. O1 and O4 claimed to have an AV app installed on their devices. O1 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

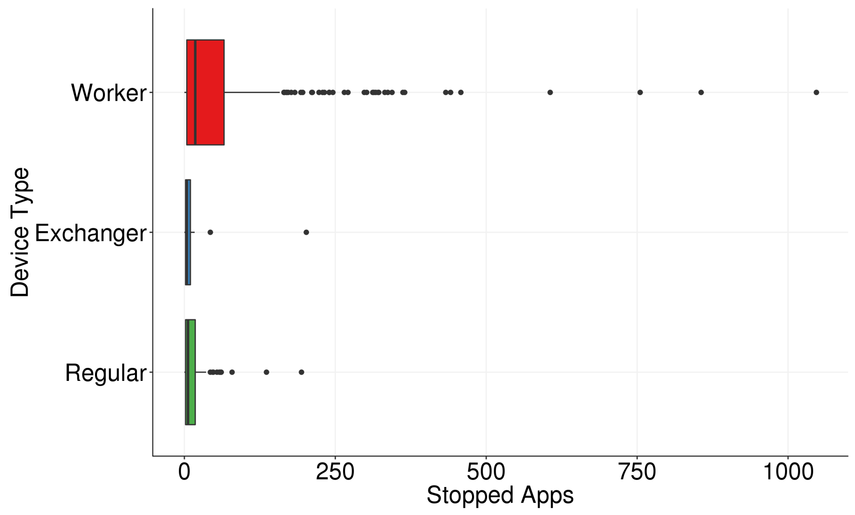


Figure 5.11: Number of stopped apps for regular and worker devices. Worker devices tend to have more stopped apps, but we also observe substantial overlap with regular devices.

"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."

Regular participant R1 claimed not to be concerned about installing malware or about privacy leaks, but claimed to have an AV app installed and to have previously installed malware apps. R2 and R3 were concerned about malware and privacy leaks, but only R2 had an AV app installed and to have detected malware.

5.6.4 Stopped Apps

We have further studied the number of apps that are stopped on the devices of regular and worker participants. Android devices allow users to stop apps, instead of uninstalling them. Figure 5.6.4 shows that some worker devices have significantly more stopped apps than regular devices. We conjecture that this occurs because ASO workers need to keep apps installed, e.g., to provide *retention installs*, but

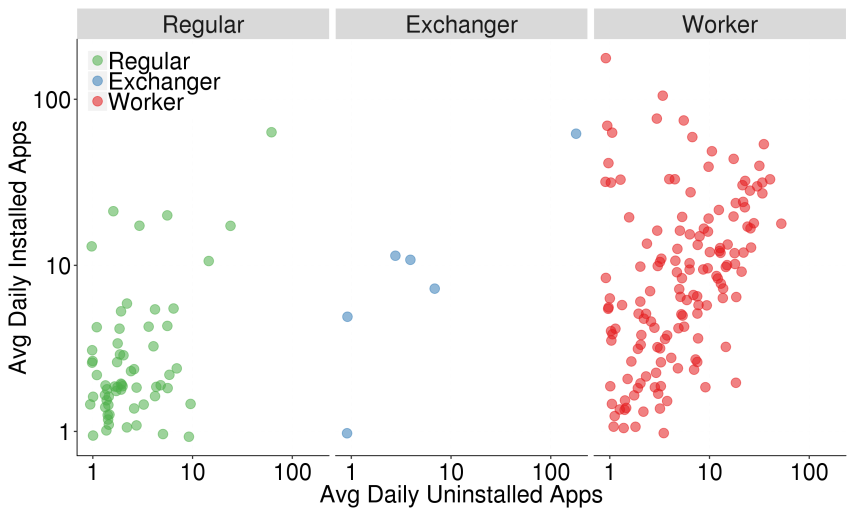


Figure 5.12: App churn: Scatterplot of average number of daily installs vs. average number of daily uninstalls for regular, worker and exchange worker controlled devices. Each dot is one device. Most regular devices install and uninstall less than 10 apps per day, while many worker devices install or uninstall more than 10 apps per day.

prefer to stop apps that misbehave. Two ASO admin participants and four organic workers claimed to never stop apps. A1 said

“*Sometimes the apps get hanged due to lack of storage on my phone.*”,

while O4 said

“*The quality of some apps was bad, I stopped those apps.*”.

5.6.5 App Churn: Install and Uninstall Events

Figure 5.6.5 shows the average number of daily install events and daily uninstall events for 304 participant devices (203 worker, 93 regular and 9 exchanger worker devices), computed over all the days when RacketStore was installed. Worker participants tend to install apps more often compared to regular users. Concretely, worker devices have an average of 14.2 daily installs (M = 4.5, SD = 28.82), exchange worker devices have an average of 15.38 installs (M = 6.0, SD = 42.05)

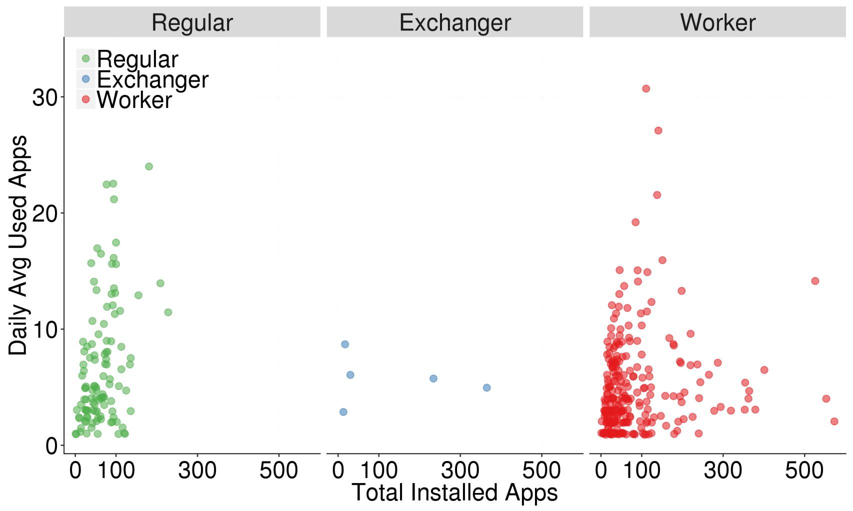


Figure 5.13: Scatterplot of the average number of apps used per day per device and the number of apps installed in a device, for all regular and worker devices. We observe substantial overlap between regular and worker devices.

and regular devices had an average 4.27 daily installs (M = 2.0, SD = 13.25). We recorded fewer daily uninstalls, thus confirming that devices have a tendency to retain apps: worker devices recorded an average of 9.29 daily uninstalls (M = 3.0, SD = 14.54), exchange worker devices had an average of 21.27 daily uninstalls (M = 2.0, SD = 58.75) and regular devices had an average of 3.69 daily uninstalls (M = 1.0, SD = 13.67). The highest number of app installs was recorded by a worker device (313) and the highest number of uninstalls was from an exchange worker device (198). We observe however that (1) many worker devices have a low daily app churn, making them hard to distinguish from regular devices and (2) some regular devices have a higher daily app churn, making it easy to confuse them for worker devices, based on this feature alone.

5.6.6 Number of Apps Used Per Day

Figure 5.6.6 shows for each of the 141 regular, 380 worker and 19 exchange worker device 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. Only 5 of the 38 exchange worker devices granted the permission that allows RacketStore to access the foreground app thus we cannot draw substantial conclusions about these devices. However, 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 leads us to believe that the daily number of used apps cannot accurately distinguish between worker and regular devices.

5.6.7 Classifier of App Usage

We found that 247 devices sent less than one day of snapshot data to our servers. We conjecture that these may be due to early uninstalls or incompatibilities. We focus the following analysis on only the remaining 524 devices: 355 controlled by workers, 19 by exchange workers and 150 controlled by regular users.

For each such device, we have computed (1) the *active interval* that measures the time difference between the last and first snapshot, and (2) the number of *active days* that captures the number of days for which the device sent data to our servers. Figure 5.6.7 shows the scatterplot of the 524 devices: each point shows the active interval and active days of one device. On average, workers kept RacketStore installed and active (active interval) for 7.22 days (M = 2.31 days, SD = 11.48 days, max = 92.17 days), exchange workers for 1.88 days (M = 0.98, SD = 2.45, max =

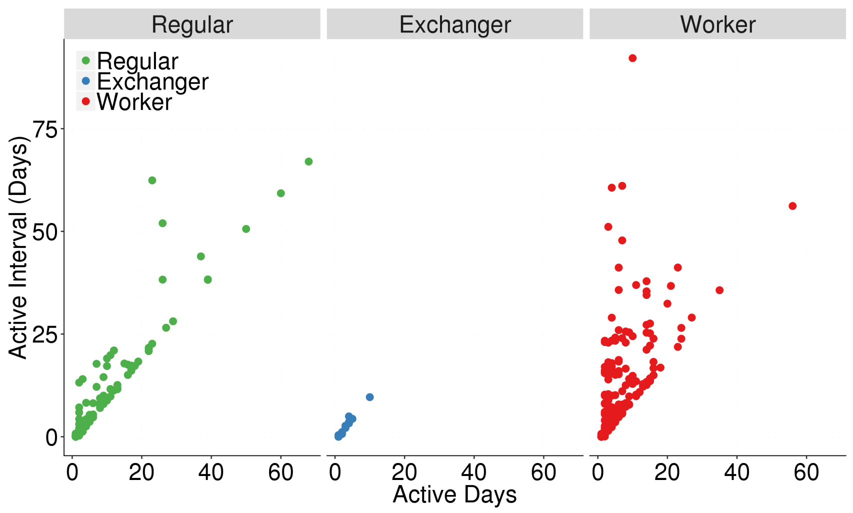


Figure 5.14: Scatterplot of active interval vs active days for 524 devices (green for regular devices, red for worker devices and blue for exchange worker devices). 357 devices (237 workers, 108 regular users and 12 exchange workers) kept RacketStore installed for more than 1 day, while 9 devices (2 workers and 7 regular users) active for 1-2 months.

9.63) while regular users for 9.32 days (M = 3.07 days, SD = 15.46 days, max = 86.55 days). Further, workers have kept the app installed, on average, for 4.6 days (M = 3 days, SD = 5.7 days, max = 56 days), exchange workers for 2.6 days (M = 2, SD = 2.19, max = 10) while regular users for 7.96 days (M = 3 days, SD = 11.09 days, max = 68 days).

Figure 5.15 is the scatterplot of the average number of snapshots per day vs. the number of active days over the 771 devices. Larger dots correspond to multiple devices. The average number of daily snapshots captured from regular devices is 9430.71 (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.

In the following experiments we only consider devices from which we have received at least two days of fast snapshots and slow snapshots. This dataset includes

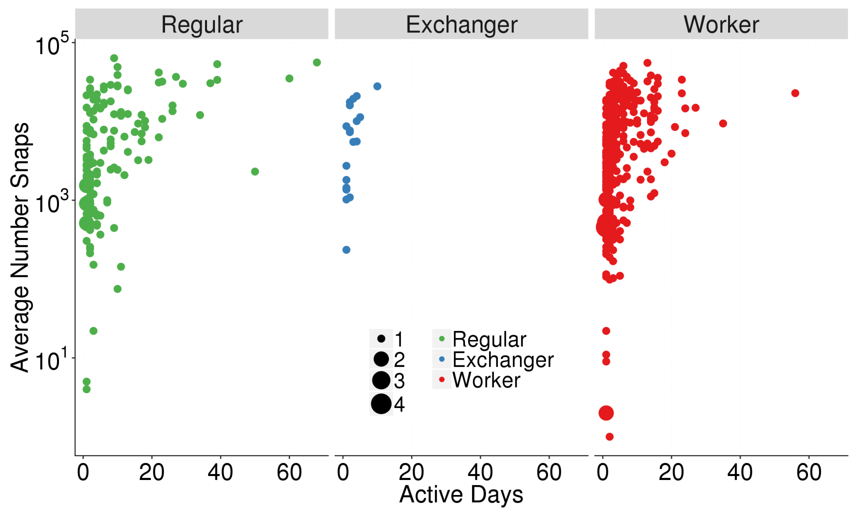


Figure 5.15: Scatterplot of average number of snapshots captured per day vs active days over regular (green), exchange worker (blue) and worker (red) devices. Dot size indicates the number of overlapping devices. Most devices report at least 100 snapshots per day.

166 worker, 88 regular and 12 exchange worker devices for a total of 266 devices. For the other devices we lack enough representative data to be able to make a reasonable decision.

We have randomly selected 20% of the devices controlled by the workers of each of the 17 participating ASO admins, and that have reported at least 2 days of RacketStore snapshots. In total, 38 worker-controlled devices. Further, we have selected 37 regular devices, that account for 42% of our set of regular devices that have reported at least 2 days of RacketStore snapshots.

We used these 38 worker-controlled and 37 regular devices to extract a set of train-and-validate apps for the app usage classifier (§ 5.4). Specifically, we have built a dataset of apps with "suspicious" and "regular" usage. We say that an app has suspicious usage if (1) it was installed on at least 5 of the worker-controlled devices, (2) it was not installed on any of the regular devices, and (3) it has received at most 10,000 reviews. That is, popular apps that are also installed by regular users, are not considered to be suspicious. Further, we say that an app has regular usage if it was not installed on any worker-controlled device but was installed in at least one regular device, and has received at least 15,000 reviews.

We have selected 1,041 suspicious usage apps from the ones installed on the 38 worker-controlled devices from the above device training set. We have selected 474 regular usage apps from the apps installed on the above 37 training regular devices. The train-and-validate dataset for the app usage classifier consists then of 2,994 suspicious instances and 345 regular instances. An instance in the dataset is a tuple that contains (1) an app and a device on which the app has been installed, (2) features extracted from the use of the app on the device (§ 5.4), and (3) a label, i.e., 1 for suspicious usage app and 0 for regular usage app.

Performance of App Usage Classifier. We have used the above train-and-validate dataset of app usage to evaluate the performance of supervised learning algorithms trained with the features introduced in § 5.4. For this, we have used 10-fold cross-validation over the 2,994 suspicious app usage instances and 345 regular instances. Table 5.1 shows the precision, recall and F1-measure of several algorithms. Extreme Gradient Boosting outperform the other algorithms, achieving an F1-measure of 99.72%. Figure 5.16 shows the top 10 most important features in classifying app usage (§ 5.4) as measured by the mean decrease in Gini [Bre01]. A higher decrease in Gini indicates higher variable importance. We observe the impact of the number of accounts registered on the device that have reviewed the app and the average time between install and review.

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 5.1: 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.

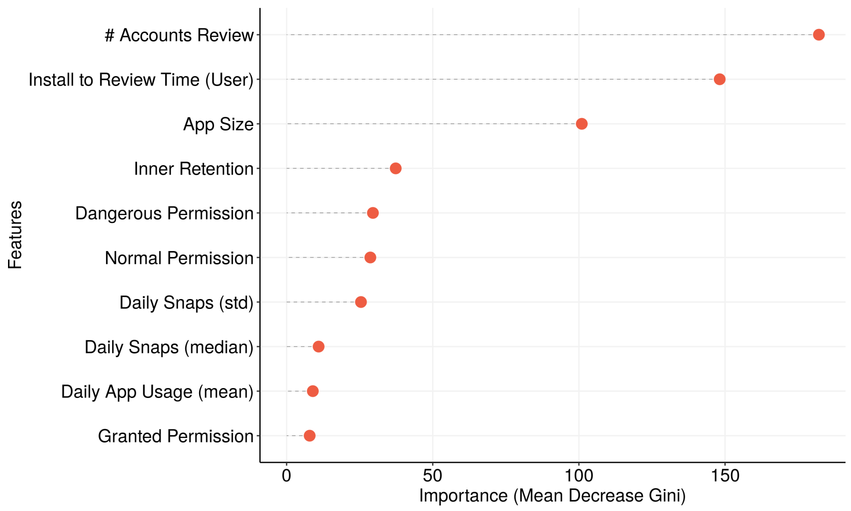
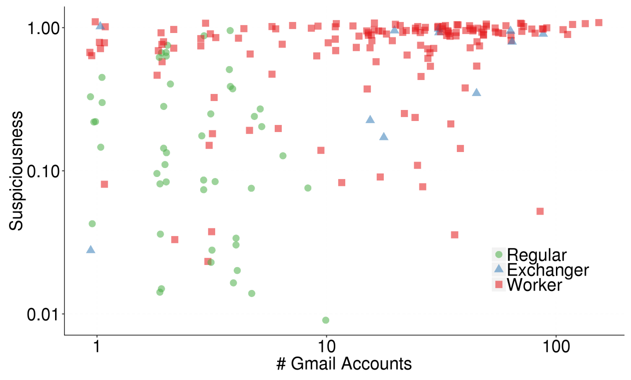
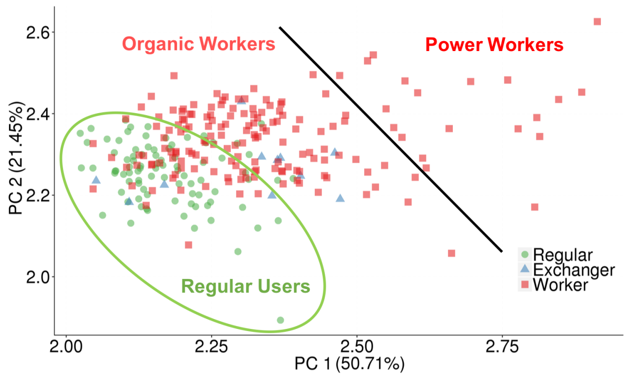


Figure 5.16: Top 10 most important features, measured by mean decrease in Gini. The number of accounts that have reviewed the app from the device and the average time between install and review are the most important features.



(a)



(b)

Figure 5.17: (a) Scatterplot of 266 devices on *app suspiciousness* vs. number of Gmail accounts registered on the device. Device types are shown with different shapes and colors. x and y axes shown in log scale. We observe distinctive clusters of worker and regular devices. (b) Scatterplot for all devices on their first two principal components computed from original features, annotated with cluster information from Table 5.2.

5.6.8 Evaluation of Device Usage

We explore the ability of the device usage features (§ 5.4) to differentiate between the 166 ASO worker, 12 exchange worker and 88 regular devices (total of 266 devices) that have reported snapshots over at least 2 days.

Figure 5.17(a) shows the scatterplot of app suspiciousness vs. number of registered Gmail accounts, and Figure 5.6.8 shows the scatterplot of average number of reviews written from an account vs. total number of accounts.

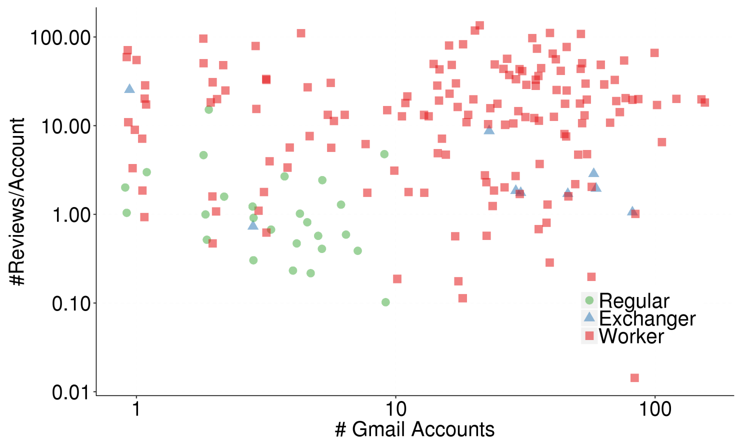


Figure 5.18: Scatterplot for 266 devices that reported at least two days of snapshots: the average number of reviews per registered account vs. number of Gmail accounts.

Participant/Cluster	**1**	**2**	**3**	**4**
Regular	0	**83**	5	0
Exchange	1	4	7	0
Worker	0	40	**107**	**19**

Table 5.2: Distribution of devices across clusters found using k-means with k=4. Cluster 4 corresponds to devices controlled by power workers (yellow cell), cluster 3 is dominated by organic worker devices, and cluster 2 is dominated by regular devices and inexperienced or novice workers.

We used k-means to cluster these devices. Table 5.2 shows the resulting clusters for $k = 4$. Figure 5.17(b) shows the scatterplot over these devices on the first two principal components that explain 72.16% of variability in the data set. We annotate the figure with the cluster information from Table 5.2.

Cluster 4 consists only of 19 **power worker** devices, with a high number of registered Gmail accounts (Mean = 60.47, M = 44, SD = 41.04) and high Gmail to total accounts ratio (Mean = 0.92, M = 0.94, SD = 0.06), that record many daily installs (Mean = 55.24, M = 46.33, SD = 44.64), are each responsible for posting an average of 2,341 reviews (M = 2,356, SD = 1,437.05, max = 5,962) and also used suspiciously an average of 412.2 apps (M = 376.0, SD = 149.82).

Cluster 3 most likely contains the organic worker devices. The 107 worker devices had each posted 583.88 reviews on average (M = 220.50, SD = 829.68, max = 583.88) and had registered 28.87 Gmail accounts on average (M = 27, SD = 23.85). Similarly, the 7 exchange workers in cluster 3 are each responsible for 83.57 reviews (M=70, SD=47.28, max=163) and had registered 44.29 Gmail accounts on average (M = 43, SD = 27). While these numbers suggest ASO worker activity (see § 5.6.2) we also observe that these devices mix in a decent amount of personal use of apps: the worker devices had used an average of 21.11% of their apps for personal tasks, and the exchange workers used 19.72% of their apps for personal purposes. The 5 miss-classified regular users in cluster 3 only posted an average of 6 reviews (M = 2, SD = 1), however, they present an unusual ratio of suspicious to total number of apps, i.e., 77% (M = 73%, SD = 14%).

Cluster 2 is likely the "regular" device cluster. It contains most of the regular devices but also includes 40 worker and 4 exchange worker devices. The regular devices have only 2.638 Gmail accounts registered on average (M = 2, SD = 2.70, max = 10) and have a low ratio of suspiciously used apps (Mean = 0.08, SD = 0.14, max = 0.08). Similarly, the 44 worker devices have very low levels of activity: on average, they have only posted 25.68 reviews in Google Play (M = 3, SD = 61.67, max = 388), have a low number of registered Google accounts: (Mean = 15.04, M = 5.5, SD = 19.12, max = 78) and use very few apps suspiciously (Mean = 16%, M = 9%, SD = 18%, max = 65%). Such "novice" worker devices thus have a small impact on the app store. We conjecture that once they gain more expertise they will move to the organic or power worker clusters.

5.7 Discussion and Limitations

Relevance of Proposed Work. Distinguishing devices of organic workers from those used exclusively for app promotion can help app stores implement appropriate responses to detected fraud. For instance, while the accounts registered on devices dedicated to app promotion could be closed, app stores could only filter or demote reviews posted from accounts registered on organic devices, and only for the apps they are suspected to have promoted.

Recruitment Bias. We performed our studies with a limited set of devices controlled by ASO workers and regular users. We have selected participants from a subset of countries and cities. Our worker recruitment process is further biased, since we contacted participants through a subset of 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. Similarly, our Instagram recruitment process reached 61,748 Instagram users from select locations but who speak English, are of restricted age, show interests related to Google Play and Android applications, and were willing to participate after approving the consent form.

Thus, while we have identified ASO admins, team workers, organic workers and exchange workers, we may be missing other types of professional app promoters. A larger scale recruitment process could help us identify more fraud worker types and more diverse regular users. However, the data that we collected reveals the potential to classify and differentiate the behaviors of various worker types even from regular users of similar location and age.

Generalization of Results. While several of the workers that we recruited claimed promotion expertise in multiple sites, e.g., Google Maps, Yelp, Amazon, TrustPilot, and Justdial, we have not evaluated their activities in such platforms. Thus, we do not claim that our findings apply to other sites or other types of ASO work.

Universal Adoption and Worker Defenses. Our approach can be effective if the RacketStore app is installed on the devices of all app market users. We envision however workers attempting to develop strategies to avoid detection by the fraud detection and classification techniques that we developed in this thesis. RacketStore may capture changes in worker strategies, however, our solutions may require adjustments to handle them. We note however that attempting to convert their app usage patterns to mimic those of personal usage apps will require substantial effort from workers. For instance, we expect that workers will have to keep apps installed for long intervals, use them regularly and review them after a long install interval. We also expect that workers will need to reduce the number of reviews written from multiple accounts registered on the same device. Thus, such efforts will reduce the amount of fraud that can be posted from a single device. We also expect that other features will further restrict worker options. For instance, the number of apps opened per day will experience observable increase if the worker needs to promote and periodically interact with many apps.

Data Size Limitations. The relatively small size of our datasets of suspicious and regular app and device usage may impact the accuracy of our classifiers. One approach to address this limitation is to use generative adversarial networks [GPAM+14] to synthesize realistic training datasets. Further, Jan et al. [JHH+20] have introduced solutions to synthesize even unseen and future data in order to identify bots with updated strategies.

CHAPTER 6

FRAUD DE-ANONYMIZATION FOR FUN AND PROFIT

6.1 Introduction

The persistence of search rank fraud in online, peer-opinion systems, made possible by crowdsourcing sites and specialized fraud workers, shows that the current approach of detecting and filtering fraud is inefficient. We introduce a fraud de-anonymization approach to disincentivize search rank fraud: attribute user accounts flagged by fraud detection algorithms in online peer-opinion systems, to the human workers in crowdsourcing sites, who control them. We model fraud de-anonymization as a maximum likelihood estimation problem, and introduce UODA, an unconstrained optimization solution. We develop a graph based deep learning approach to predict ownership of account pairs by the same fraudster and use it to build discriminative fraud de-anonymization (DDA) and pseudonymous fraudster discovery algorithms (PFD).

To address the lack of ground truth fraud data and its pernicious impacts on online systems that employ fraud detection, we propose the first cheating-resistant *fraud de-anonymization validation* protocol, that transforms human fraud workers into ground truth, performance evaluation oracles. In a user study with 16 human fraud workers, UODA achieved a precision of 91%. On ground truth data that we collected starting from other 23 fraud workers, our co-ownership predictor significantly outperformed a state-of-the-art competitor, and enabled DDA and PFD to discover tens of new fraud workers, and attribute thousands of suspicious user accounts to existing and newly discovered fraudsters.

Results. We conducted the fraud de-anonymization validation protocol, through a user study with 16 human fraud workers, who revealed control of a total of 230

Google Play accounts. The participants confirmed control of 91% of the user accounts newly discovered by UODA. Further, on 942 ground truth attributed user accounts that we collected from other 23 fraud workers, both DDA and UODA achieved precision and recall that exceed 90%, and attributed thousands of new accounts to these fraudsters.

We introduce intuition, and empirically evaluate the impact of features used by our co-ownership predictor. Our predictor outperformed the F1-measure of state-of-the-art, ELSIEDET's Sybil social link builder [ZXL+18] by more than 12 percentage points, on ground truth attributed data. Further, the PFD algorithm identified thousands of accounts not previously known to be fraudulent, grouped into communities according to common ownership by fraudsters. We analyzed 1.1 billion pairs of reviews from these communities and report orthogonal evidence of fraud, including communities with more than 80% of accounts involved in review text plagiarism. In summary, our contributions are the following:

- **Fraud de-anonymization**. Model fraud de-anonymization as a maximum likelihood estimation problem. Develop UODA, an unconstrained optimization fraud de-anonymization algorithm [§ 6.3].
- **Co-ownership predictor**. Introduce a graph based deep learning approach to predict ownership of account pairs by the same fraudster [§ 6.5]. Leverage the predictor to build DDA, a discriminative fraud de-anonymization [§ 6.6] and PFD, a pseudonymous fraudster discovery algorithm [§ 6.7].
- **Human fraud de-anonymization oracles**. Develop the first protocol to provide human-fraud-worker-based performance evaluation of fraud de-anonymization algorithms [§ 6.9]. Evaluate proposed solutions using data collected through this protocol [§ 6.11].

6.2 Problem Definition

The insight that multiple fraud workers usually target a single subject suggests that a binary classification of fraud, e.g., fake vs. honest reviews, fraudulent vs. genuine accounts [XZLW16, FLCS15, MKL$^+$13, FML$^+$13, HTS16, LFW$^+$17], is insufficient to understand and model fraud. Instead, we study the *fraud de-anonymization problem* which deals with attributing fraudulent accounts and fake reviews to the crowdsourcing accounts of the fraud workers who control and post them, respectively.

Formally, let $\mathcal{U}$ be the set of all user accounts, and let $\mathcal{S}$ be the set of all subjects hosted in the online peer-opinion system. We say that a user account is fraudulent or *fraudster-controlled* if it was opened by a fraudster to mainly perform fraudulent activities in the online system, i.e., to target subjects from $\mathcal{S}$.

Moreover, let $U^* \subseteq \mathcal{U}$ be the set of all fraudster-controlled accounts in an online system, and let $\mathcal{W}$ be the set of all fraud worker accounts in crowdsourcing sites. In addition, let $W^* = \{(W_l, U_l, S_l)|\ W_l \in \mathcal{W}, U_l \subseteq U^*, S_l \subseteq \mathcal{S}, l = 1 \ldots f\} \subset \mathcal{V}$ be a known set of f search rank fraud worker profiles where $\mathcal{V}$ is the universe of all worker profiles. A profile consists of a crowdsourcing account id (W_l), an incomplete set of user accounts (U_l) known to be controlled by W_l in the peer-opinion system, and the incomplete set of subjects (S_l) known to have been fraudulently reviewed by W_l. Section 6.9 describes a protocol to identify crowdsourced fraud workers and build seed profiles for them.

Ideally, we want to attribute each account in U^* to the fraudster who controls it. However, some accounts in U^* may not be controlled by any of the known fraudsters in W^*. To address this issue, we formulate two distinct problems: fraud de-anonymization and pseudonymous fraudster discovery:

Fraud De-Anonymization. Build a function $FDA\colon U^* \setminus \cup_{l=1}^{f} U_l \mapsto W^*$, that, given a user account $u \in U^*$ suspected of participation in search rank fraud, returns the fraud worker in W^* most likely to control u. In Section 6.3.1 we expand this definition in a maximum likelihood estimation (MLE) based framing of the problem.

Pseudonymous Fraudster Discovery. Build a function $PFD\colon U^* \setminus \cup_{l=1}^{f} U_l \mapsto \mathcal{V} \setminus W^*$ that, given a set of fraudster-controlled accounts that were not assigned to one of the known fraudsters by the FDA function, returns a new set of fraudster profiles from $\mathcal{V} \setminus W^*$ that control these accounts.

Unlike standard de-anonymization, the adversarial process of identifying users from data where their Personally Identifiable Information (PII) has been removed [NS08], the fraud de-anonymization problem seeks to attribute detected search rank fraud to the humans who posted it. A solution to this problem will enable peer-review services to identify the impactful crowdsourcing fraudsters who target them, and provide appealing fraud feedback proof to customers, e.g., links to the crowdsourcing accounts responsible for boosting a product's rating. Furthermore, accurate fraud de-anonymization will allow online services and law enforcement to retrieve banking information and real identities of fraudsters. Thus, fraud de-anonymization may provide counter-incentives for crowdsourcing workers to participate in fraud jobs, and for product developers to recruit them.

In Section 6.3 and 6.6, we introduce unconstrained optimization and discriminative fraud de-anonymization algorithms, respectively, while in Section 6.7 we propose a pseudonymous fraudster discovery algorithm. In Section 6.8, we show how DE-TEGO iteratively invokes a pseudonymous fraudster discovery algorithm followed by a fraudster de-anonymization algorithm, to expand knowledge of fraud workers and the accounts they control.

6.3 Unconstrained Optimization Based De-Anonymization

We first propose a maximum likelihood based de-anonymization approach motivated by a realistic generative model of review posting behavior. Next, we compute the likelihood of each worker having generated a given suspicious fraudulent review history. We then find the worker who maximizes such likelihood.

6.3.1 Definitions and Approach

We postulate a probabilistic review-posting model from accounts controlled by fraudsters, inspired by Su et al. [SSGN17]. Specifically, we assume that a fraudulent account u controlled by a fraudster profile $(W, U, S) \in W^*$ is likely to review subjects in a pairwise-disjoint family of sets over $\mathcal{S}$, $\mathcal{F}_W = \{\Omega_1, \Omega_2, \ldots, \Omega_m\}$ $(\Omega_i \cap \Omega_j = \emptyset \; \forall \; i \neq j)$ with different multiplicative factors $r_1, r_2, \ldots, r_m$ describing u's responsiveness to each Ω_i. Further, we assume that the review history of a user account is described by a sequence of independent and identically distributed random variables $R_1, R_2, \ldots, R_n$ where $R_k \in \mathcal{S}$ represents the k-th subject reviewed from the account. Therefore, a fraudulent account's review posting behavior is characterized by $\mathcal{F}_W$and r_i for all $i = 1 \ldots m$.

Let $\{p_j\}$ be a probability measure over the sample space $\mathcal{S}$, related to the popularity of the subjects: $p_j \geq 0, \sum_{j=1}^{|\mathcal{S}|} p_j = 1$. For any fraudster profile $(W, U, S) \in W^*$, we define random variable $R_k(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$ with values in $\mathcal{S}$ and with the probability distribution:

$$\mathbb{P}(R_k = s_j) = \begin{cases} \frac{r_1 p_j}{c} & \text{if } s_j \in \Omega_1 \\ \ldots & \\ \frac{r_m p_j}{c} & \text{if } s_j \in \Omega_m \\ \frac{p_j}{c} & \text{if } s_j \in \bigcap_{i=1}^{m} \Omega_i^C \end{cases} \tag{6.1}$$

where $c = \sum_{i=1}^{m} r_i \sum_{s_j \in \Omega_i} p_j + \sum_{s_j \in \bigcap_{i=1}^{m} \Omega_i^C} p_j$ and $\mathbf{r} = [r_1, \ldots, r_m]^\intercal$ is the vector of multiplicative factors. Specifically, the probability that the k-th review targets subject s_j is proportional to factor r_m if subject s_j satisfies Ω_m's membership properties. Otherwise, this probability is simply given by the ratio p_j/c.

Let $R_1(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, $R_2(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, ..., $R_n(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, be a review history suspected to be fraudulent. Given a set of candidate workers, each described by a family of sets $\mathcal{F}_{\mathbf{W}}$, the fraudster de-anonymization problem derives the maximum likelihood estimates $\hat{\mathbf{r}}$ and $\hat{\mathcal{F}_{\mathbf{W}}}$ of the function:

$$\mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r}) = \left(\prod_{i=1}^{m} \prod_{R_k \in \Omega_i} \mathbb{P}(R_k \mid \mathcal{F}_{\mathbf{W}}, \mathbf{r}) \right) \prod_{R_k \in \bigcap_{i=1}^{m} \Omega_i^C} \mathbb{P}(R_k \mid \mathcal{F}_{\mathbf{W}}, \mathbf{r}) \tag{6.2}$$

where $\hat{\mathcal{F}_{\mathbf{W}}}$ is the family of sets associated with the worker most likely linked with the given review history.

6.3.2 UODA

We introduce UODA, an unconstrained optimization based de-anonymization approach that maximizes the function in Equation 6.2 without any constraints on the multiplicative values $r_1, \ldots, r_m$. Theorem 6.3.1 characterizes the solution for the fraudster de-anonymization problem under this unconstrained setting.

Theorem 6.3.1. *Let $\mathcal{S}$ be the set of subjects hosted by the online service, and $\{p_j\}$ be a probability measure on $\mathcal{S}$ ($p_j \geq 0, \sum_{j=1}^{|\mathcal{S}|} p_j = 1$). Let $\mathcal{C} = \{\mathcal{F}_{W_1}, \dots, \mathcal{F}_{W_f}\}$ be a collection of family sets for each fraud worker, where $\mathcal{F}_{W_l}$=$\{\Omega_{l1}, \Omega_{l2}, \dots, \Omega_{lm}\}$. For any $\mathcal{F}_W \in \mathcal{C}$, define a random variable $R_k(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$ taking values in $\mathcal{S}$ and obeying the probability distribution in Equation* (6.1). *Given a review history $R_1(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, $R_2(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, $\dots$, $R_n(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$ suspected to be fraudulent, the maximum likelihood estimates $\hat{\mathbf{r}}$ and $\hat{\mathcal{F}_{\mathbf{W}}}$ are:*

$$\hat{r}_t = \frac{q_t\,(1 - \sum_{i=1}^m P_i)}{P_t\,(1 - \sum_{i=1}^m q_i)} \quad \text{for } t = 1, \dots, m \tag{6.3}$$

and

$$\hat{\mathcal{F}_{\mathbf{W}}} = \operatorname*{argmax}_{\mathcal{F}_W \in \mathcal{C}} \left[\sum_{i=1}^m q_i \ln\left(\frac{q_i}{P_i}\right) - \left(1 - \sum_{i=1}^m q_i\right) \ln\left(\frac{1 - \sum_{i=1}^m P_i}{1 - \sum_{i=1}^m q_i}\right) \right] \tag{6.4}$$

where $q_i = |\{k \mid R_k \in \Omega_i\}|/n$ *and* $P_i = \sum_{s_j \in \Omega_i} p_j \quad \text{for } i = 1, \dots, m$

Intuition. Equation (6.4) from Theorem 6.3.1 attributes a user account to the worker profile in W^* most likely responsible for the account's review history $R_1(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, $R_2(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$, $\dots$, $R_n(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$. The Ω sets partition worker's reviews into groups of subjects that have different characteristics (features). q_i is the fraction of subjects in the account's review history that are in the investigated worker's Ω_i. P_i is the total popularity of all the subjects in the set Ω_i. The first term of Equation (6.4) reveals that the $\hat{\mathcal{F}_{\mathbf{W}}}$ associated worker most likely to control the suspect account has a family of Ω sets for which most of q_i are large and P_i are small; that is, many of the subjects in the account's review history appear in the worker's sets Ω_i that are neither too big or popular.

Proof. Setting $R_k = s_k$, we rewrite Equation (6.2) as:

$$\mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r}) = \prod_{k=1}^{n} \left(\sum_{i=1}^{m} \frac{r_i p_k}{c} X_{\Omega_i}(s_k) + \frac{p_k}{c} X_{\bigcap_{i=1}^{m} \Omega_i^C}(s_k) \right)$$

when using indicator functions $X_{\Omega_i}(s)$ for $i = 1, \ldots, m$, i.e. $X_{\Omega_i}(s) = 1$ if $s \in \Omega_i$, and $X_{\Omega_i}(s) = 0$ otherwise. We can then write the log-likelihood function as follows:

$$\begin{aligned}
\ln \mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r}) &= \sum_{k=1}^{n} \ln \left(\sum_{i=1}^{m} \frac{r_i p_k}{c} X_{\Omega_i}(s_k) + \frac{p_k}{c} X_{\bigcap_{i=1}^{m} \Omega_i^C}(s_k) \right) \\
&= \sum_{k=1}^{n} \left(\sum_{i=1}^{m} X_{\Omega_i}(s_k) \ln \left(\frac{r_i p_k}{c} \right) + X_{\bigcap_{i=1}^{m} \Omega_i^C}(s_k) \ln \left(\frac{p_k}{c} \right) \right) \\
&= n \left(\sum_{i=1}^{m} q_i \ln(r_i) + \ln(p_k) - \ln(c) \right)
\end{aligned}$$

We can further rewrite c:

$$\begin{aligned}
c &= \sum_{i=1}^{m} r_i \sum_{s_j \in \Omega_i} p_j + \sum_{s_j \in \bigcap_{i=1}^{m} \Omega_i^C} p_j \\
&= \sum_{i=1}^{m} r_i \sum_{s_j \in \Omega_i} p_j + \left(\sum_{s_j \in \mathcal{S}} p_j - \sum_{i=1}^{m} \sum_{s_j \in \Omega_i} p_j \right) \\
&= \sum_{i=1}^{m} P_i (r_i - 1) + 1
\end{aligned}$$

Therefore,

$$\ln \mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r}) = n \left(\sum_{i=1}^{m} q_i \ln(r_i) + \ln(p_k) - \ln \left(\sum_{i=1}^{m} P_i (r_i - 1) + 1 \right) \right)$$

The first-order necessary conditions are:

$$\frac{\partial \ln \mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r})}{\partial r_i} = \frac{-n P_i}{\sum_{i=1}^{m} P_i (r_i - 1) + 1} + \frac{n q_i}{r_i} = 0 \quad \text{for } i \in [m] \qquad (6.5)$$

We can also write (6.5) as the $m \times m$ non-homogeneous system of linear equations:

$$[P_i(1-q_i)]r_i - q_i \sum_{d \neq i} P_d r_d = q_i \left(1 - \sum_{i=1}^{m} P_i\right) \quad \text{for } i \in [m] \tag{6.6}$$

To solve the system of equations (6.6), we introduce the following lemma, whose proof we present at...

Lemma 6.3.2. *The system of linear equations*

$$[P_i(1-q_i)]r_i - q_i \sum_{d \neq i} P_d r_d = q_i \left(1 - \sum_{i=1}^{m} P_i\right) \quad \text{for } i \in [m]$$

has solutions given by $r_t = \frac{q_t\left(1-\sum_{i=1}^m P_i\right)}{P_t\left(1-\sum_{i=1}^m q_i\right)}$

This enables us to write c as:

$$\begin{aligned}
c &= \sum_{i=1}^{m} P_i(r_i - 1) + 1 \\
&= \sum_{i=1}^{m} P_i \left(\frac{q_i}{P_i} \frac{(1-\sum_{i=1}^m P_i)}{(1-\sum_{i=1}^m q_i)} - 1 \right) + 1 \\
&= \frac{\sum_{i=1}^m q_i(1-\sum_{i=1}^m P_i) + (1-\sum_{i=1}^m q_i)(1-\sum_{i=1}^m P_i)}{1-\sum_{i=1}^m q_i} \\
&= \frac{1-\sum_{i=1}^m P_i}{1-\sum_{i=1}^m q_i}
\end{aligned}$$

Thus, the value of $\mathbf{r}$ at which $\ln \mathcal{L}(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$ reaches its maximum must also maximize the function $L(\mathcal{F}_{\mathbf{W}}, \mathbf{r})$ defined as:

$$\begin{aligned} L(\mathcal{F}_{\mathbf{W}}, \mathbf{r}) &= \sum_{i=1}^{m} q_i \ln(r_i) - \ln(c) \\ &= \sum_{i=1}^{m} q_i \ln(r_i) - \ln\left(\frac{1 - \sum_{i=1}^{m} P_i}{1 - \sum_{i=1}^{m} q_i}\right) \\ &= \sum_{i=1}^{m} q_i \ln\left(\frac{q_i}{P_i}\right) - \left(1 - \sum_{i=1}^{m} q_i\right) \ln\left(\frac{1 - \sum_{i=1}^{m} P_i}{1 - \sum_{i=1}^{m} q_i}\right) \end{aligned}$$

□

In Section 6.11.2 we instantiate UODA for two features that define the Ω sets.

6.4 Proof of Lemma 6.3.2

Proof. We note that (6.6) can be expressed in matrix form as:

$$(\operatorname{diag}(\mathbf{p}) - \mathbf{q}\mathbf{p}^\intercal)\mathbf{r} = \left(1 - \sum_{i=1}^{m} P_i\right)\mathbf{q} \tag{6.7}$$

where $\mathbf{p} = [P_1, \ldots, P_m]^\intercal$, $\mathbf{q} = [q_1, \ldots, q_m]^\intercal$, $\mathbf{r} = [r_1, \ldots, r_m]^\intercal$ and $\operatorname{diag}(\mathbf{p})$ is the $m \times m$ diagonal matrix with $\operatorname{diag}(\mathbf{p})_{ii} = P_i$.

We also note that:

$$\begin{aligned} \mathbf{A} &= \operatorname{diag}(\mathbf{p}) - \mathbf{q}\mathbf{p}^\intercal \\ &= \operatorname{diag}(\mathbf{p}) - \mathbf{q}\mathbf{1}^\intercal \operatorname{diag}(\mathbf{p}) \\ &= (\mathbf{I} - \mathbf{q}\mathbf{1}^\intercal) \operatorname{diag}(\mathbf{p}) \end{aligned}$$

and therefore:

$$
\begin{aligned}
\det(\mathbf{A}) &= \det((\mathbf{I} - \mathbf{q}\mathbf{1}^\intercal)\operatorname{diag}(\mathbf{p})) \\
&= \det(\mathbf{I} - \mathbf{q}\mathbf{1}^\intercal)\prod_{i=1}^{m} P_i \\
&= \left(1 - \sum_{i=1}^{m} q_i\right)\prod_{i=1}^{m} P_i
\end{aligned}
$$

where $\mathbf{1} = [1, \ldots, 1]^\intercal$ and the last equality follows from Sylvester's determinant theorem.

Let $\mathbf{A_t}$ be the matrix formed by replacing the t-th column of $\mathbf{A}$ by the column vector $(1 - \sum_{i=1}^{m} P_i)\,\mathbf{q}$. Thus,

$$
\mathbf{A_t} = \left[\mathbf{a_1}, \ldots, \left(1 - \sum_{i=1}^{m} P_i\right)\mathbf{q}, \ldots, \mathbf{a_m}\right]
$$

where $\mathbf{a_t}$ represents the $t-$th column of matrix $\mathbf{A}$. We also note that

$$
\begin{aligned}
\mathbf{a_t} &= P_t\mathbf{e_t} - P_t\mathbf{q} \\
\mathbf{q} &= \mathbf{e_t} - \frac{1}{P_t}\mathbf{a_t}
\end{aligned}
$$

where $\mathbf{e_t}$ denotes the vector with a 1 in the t-th coordinate and 0's elsewhere. By properties of the determinant, it is plain that:

$$\begin{aligned}
&\det(\mathbf{A_t}) = \\
&\left(1 - \sum_{i=1}^{m} P_i\right) \det([\mathbf{a_1}, \ldots, \mathbf{q}, \ldots, \mathbf{a_m}]) \\
&= -\frac{(1 - \sum_{i=1}^{m} P_i)}{P_t} \det([\mathbf{a_1}, \ldots, \mathbf{a_t} - P_t\mathbf{e_t}, \ldots, \mathbf{a_m}]) \\
&= -\frac{(1 - \sum_{i=1}^{m} P_i)}{P_t} (\det(\mathbf{A}) - P_t \det([\mathbf{a_1}, \ldots, \mathbf{e_t}, \ldots, \mathbf{a_m}])) \\
&= -\frac{(1 - \sum_{i=1}^{m} P_i)}{P_t} (\det(\mathbf{A}) - P_t(-1)^{t+t}\mathrm{Minor}(\mathbf{A})_{tt}) \\
&= -\frac{(1 - \sum_{i=1}^{m} P_i)}{P_t} \left[\left(1 - \sum_{i=1}^{m} q_i\right) \prod_{i=1}^{m} P_i - P_t \left(1 - \sum_{i \neq t} q_i\right) \prod_{i \neq t} P_i\right] \\
&= -\frac{(1 - \sum_{i=1}^{m} P_i)}{P_t} \left[\left(1 - \sum_{i=1}^{m} q_i - 1 + \sum_{i \neq t} q_i\right) \prod_{i=1}^{m} P_i\right] \\
&= \frac{q_t(1 - \sum_{i=1}^{m} P_i) \prod_{i=1}^{m} P_i}{P_t}
\end{aligned}$$

By Cramer's rule it follows that:

$$r_t = \frac{\det(\mathbf{A_t})}{\det(\mathbf{A})} = \frac{q_t\,(1 - \sum_{i=1}^{m} P_i)}{P_t\,(1 - \sum_{i=1}^{m} q_i)}$$

We are left to prove that $\mathrm{Minor}(\mathbf{A})_{tt} = \left(1 - \sum_{i \neq t} q_i\right) \prod_{i \neq t} P_i$, but this follows from the construction of $\mathbf{A}$. Take

$\mathbf{p}_{-t} = [P_1, \ldots, P_{t-1}, P_{t+1}, \ldots, P_m]^\intercal$ and

$\mathbf{q}_{-t} = [q_1, \ldots, q_{t-1}, q_{t+1}, \ldots, q_m]^\intercal$, we then have:

$$\begin{aligned}
\det(\mathbf{A}_{-t,-t}) &= \det(\mathrm{diag}(\mathbf{p}_{-t}) - \mathbf{q}_{-t}\mathbf{p}^\intercal_{-t}) \\
&= \left(1 - \sum_{i \neq t} q_i\right) \prod_{i \neq t} P_i = \mathrm{Minor}(\mathbf{A})_{tt}
\end{aligned}$$

□

6.5 Co-Ownership Predictor

We develop a co-ownership predictor function $cowPred \colon \mathcal{U} \times \mathcal{U} \mapsto \{0, 1\}$ that determines if two user accounts are controlled by the same fraud worker. Specifically, given two user accounts u_i and u_j, $cowPred(u_i, u_j) = 1$ if u_i and u_j are controlled by the same fraudster. $cowPred$ uses several features, that model similarity of behaviors between the input accounts. One such feature is extracted by DeepCluster, a semi supervised learning approach that we propose to cluster user accounts.

6.5.1 DeepCluster

DeepCluster leverages DeepWalk features [PARS14] extracted from co-review graphs. Given a subject s and its reviewer set $\mathcal{U}_s \subset \mathcal{U}$ (i.e., accounts who reviewed it), we define its **co-review graph** to be a weighted graph $G_s = (V_s, E_s)$, where $V_s = \mathcal{U}_s$ and $(u_i, u_j) \in E_s$ iff users u_i, u_j have reviewed the same $w(u_i, u_j)$ subjects other than s itself. Further, given a set of co-review graphs $\mathcal{G} = \{G_1, \ldots, G_k\}, G_i = (V_i, E_i)$, we define their **union fraud graph** to be the union of all the individual co-review graphs, viz., $V = \cup V_i$ and $E = \cup E_i$ for $1 \leq i \leq m$.

DeepCluster, see Algorithm 1, clusters co-review graph nodes (user accounts) based on their DeepWalk features [PARS14], that go beyond their 1-hop neighbors and are based on random walks in the union fraud graph. DeepCluster precomputes the DeepWalk features of each account in the union fraud graph (line 1). We discuss the choice of DeepWalk parameters in § 6.11. For each subject s_i, $i \in [k]$, DeepCluster extracts all its users' features (line 3), and uses any fraud account detection algorithm, e.g. [RRCL17, ACF13] to filter out the subject's honest reviewers and their accounts (line 4). DeepCluster then uses a clustering algorithm (e.g., K-means) to group the fraudulent candidate accounts of subject s_i, $i \in [k]$ (line 5).

Algorithm 1: DeepCluster identifies communities of fraudulent accounts who targeted input subjects $s_1, .., s_k$, based on the similarity of their DeepWalk features extracted from the union fraud graph of the subjects.

```
  Input  : $CoR[1 \dots k]$; # Co-review graphs of reviewers of subjects
           $s_1, \dots, s_k$;
           $DWParams$; # Best DeepWalk parameters;
           $UFG$; # Union Fraud Graph over CoR[];
  Output: $clusters[1 \dots k][\,]$; # Best clusters for $s_1, \dots, s_k$
1 $UFeatures[\,][\,] = UFG.DWFeatures(DWParams)$
2 for $i = 1$ to $k$ do
3     $candidates[\,][\,] = CoR[i].V \ltimes UFeatures$
4     $candidates[\,][\,] = FilterHonest(candidates[\,][\,])$
5     $clusters[i] = getBestClusters(candidates)$
6 end
7 return $clusters[\,][\,]$
```

6.5.2 Features

DeepCluster returns k cluster sets, one set for each of the k subjects s_i (line 7). We use these clusters to extract *cowPred*'s first feature, **Co-cluster weight**: The number of times that u_i and u_j have appeared in the same cluster identified by DeepCluster. We further introduce several other features:

• **Co-review weight**. The co-review weight of two accounts is computed over their commonly reviewed subjects. Specifically, if $\mathcal{S}_k$ is the set of subjects reviewed by u_k, we define the co-review weight of u_i and u_j as $|\mathcal{S}_i \cap \mathcal{S}_j|$.

• **Inter-review times**. We define the *date difference* attribute for a subject $s_k \in \mathcal{S}_i \cap \mathcal{S}_j, i \neq j$ as $\Delta_{Tij}(s_k) = |dt(u_i, s_k) - dt(u_j, s_k)|$, where $dt(u, s)$ denotes the date on which user u performed an activity on subject s. Let the multiset $L_{ij} = \{\Delta_{Tij}(s_k)\}_{k=1}^{|\mathcal{S}_i \cap \mathcal{S}_j|}$. L_{ij} is a multiset, thus can contain duplicate elements. We compute the minimum, mean, median, maximum, mode, and standard deviation over L_{ij}, and obtain a vector of review-time related features in $\mathbb{R}^6$. Further, we define the *unique lockstep* feature, $u_L \in \mathbb{N}$, to be the number of unique ways (with respect to review-posting time) in which two accounts were used across subjects, i.e., the number of unique elements in the multiset L_{ij}.

• **Rating difference**. We define the *rating difference* predictor as $\Delta_{Rij}(s_k) = |R(u_i, s_k) - R(u_j, s_k)|$, where $R(u, s)$ is the rating assigned by user u to subject s. We use the multiset $L_{Rij} = \{\Delta_{Rij}(s_k)\}_{k=1}^{|\mathcal{S}_i \cap \mathcal{S}_j|}$ to derive minimum, mean, median, maximum, mode, and standard deviation for this feature over all the subjects in the intersection and obtain a vector of rating features in $\mathbb{R}^6$. Further, we also extract its number of unique elements $u_R \in \mathbb{N}$.

Intuition. Accounts with high co-review and co-cluster weights are more likely to be controlled by the same fraudster. They have not only reviewed many subjects in common, but they also have similar neighbors (as identified by DeepWalk and DeepCluster) in the individual co-review graphs of those subjects.

For the inter-review features, the statistics computed over L_{ij} leverage the observation that fraudsters synchronize the activities of the accounts that they control, e.g., in a "lockstep" behavior [BXG+13, SMJ+15, TZX+15]. Since fraudsters need to meet tight deadlines [SLK15], we expect $\Delta_{Tij}(s_k)$ to be lower for user accounts controlled by the same worker (fake review "burstiness" assumption [MKL+13, FML+13, HTS16, LFW+17, HSB+16a, BSLL+16]). Further, we expect the unique lockstep u_L to be lower for pair of accounts governed by the same fraudster.

For the rating difference features, we expect u_R to be lower for pair of accounts controlled by the same worker, which would imply that both accounts tend to post the same rating for their common subjects. In Section 6.11.4 we use regularized logistic regression to provide further insights into the impact of these features. We train the co-ownership predictor on the 16 features above and use it to devise a fraud de-anonymization and a pseudonymous fraudster discovery algorithm.

6.6 DDA: Discriminative De-Anonymization

We introduce a discriminative de-anonymization solution (DDA), a classifier that approximates the function FDA: $U^* \setminus \cup_{l=1}^{f} U_l \mapsto W^*$ defined in Section 6.2. We exploit the intuition that in DeepCluster, accounts in a union fraud graph that are controlled by the same fraudster, form a densely connected subgraph, or cluster. Knowledge that some accounts in such a cluster are controlled by a fraud worker, would allow one to attribute the other accounts in that cluster, to the same worker. However, our experiments revealed that clusters often contain accounts controlled by different fraudsters, as fraudsters tend to collaborate in search rank fraud jobs.

To disambiguate this fraud attribution problem, we leverage the co-ownership predictor, of Section 6.5. Specifically, DDA analyzes the clusters returned by DeepCluster (see Section 6.5.1). Some of the clusters may consist of both un-attributed accounts and user accounts known to be controlled by a fraud worker profile in W^*. DDA separately processes each un-attributed account u in such clusters. First, it creates links (u, u_w), for each account u_w controlled by a worker w in u's cluster. Then, it uses $cowPred(u, u_w)$ to determine if u and u_w share the same owner. Note that u may appear in multiple clusters, computed by DeepCluster for multiple subjects. DDA extracts $|W^*|$ features for u: for each fraudster profile in W^*, the feature consists of the number of nodes controlled by that fraudster, to whom u has a link according to $cowPred(u, u_w)$. DDA uses these features to train a supervised learning algorithm.

6.7 PFD: Pseudonymous Fraudster Discovery

Following the fraud attribution process (e.g., UODA or DDA), we are left with suspected fraudulent user accounts that have not been attributed to any of the known

Algorithm 2: Detego system iteratively attributes new fraud to known fraudsters and discovers new fraudsters.

```
   Input  : $W^*[\,][\,]$; # seed worker profiles
   Output: $W^*[\,][\,]$; # extended worker profiles
1  $S = W^*.getProducts()$; $f = W^*.size()$;
2  while (S.notEmpty()) do
3      $U = S.getReviewerAccounts()$;
4      $< W^*[1..f], U_N >= FDA.(U, S, W^*)$;
5      $W^*[f+1, ..f+k] = PFD(U_N)$;
6      $S = W^*.getFreshProducts()$; $f = W^*.size()$;
7  end
8  return
```

fraudsters. We introduce now the pseudonymous fraudster discovery (PFD) algorithm that groups these un-attributed accounts into communities likely controlled by the same, albeit not yet discovered, fraudster.

PFD uses the co-ownership predictor of Section 6.5 to build a co-ownership graph $G_c = (V_c, E_c)$ over the unknown accounts. Nodes V_c are fraudster-controlled but un-attributed user accounts, while an edge in E_c exists between two nodes if the accounts are controlled by the same worker as predicted by *cowPred*. PFD then recursively applies a Karger [Kar93], weighted min-cut inspired algorithm to partition the co-ownership graph into two subgraphs. These subgraphs are more densely connected than the original graph and connected through links of minimal total weight. We use triangle density $\rho(G) = \frac{t(V)}{\binom{|V|}{3}}$ for an un-weighted graph $G = (V, E)$, where $t(V)$ is the number of triangles formed by the edges in E.

6.8 Putting It All Together

We introduce Detego, a fraud attribution and fraudster discovery system (see Algorithm 2). Detego takes as input a seed set W^* of f known fraudster profiles, which include user accounts known to be controlled by each fraudster. Detego expands this seed data, iteratively attributing more accounts to the known fraudsters, and identifying new fraudsters.

Detego identifies the subjects S reviewed by the accounts controlled by the seed fraudsters (Algorithm 2, line 1), then retrieves all the user accounts U who reviewed these subjects (line 3). The accounts in U include accounts controlled by the f fraudster profiles in W^*, as well as accounts controlled by other, not yet identified fraudsters, and also honest accounts. Detego uses a fraud de-anonymization (FDA) algorithm, e.g., either UODA or DDA to (1) attribute accounts from U to the fraudster profiles in W^* (line 4), and (2) identify the other, non-attributed accounts from U, denoted by U_N. Detego uses the PFD algorithm (line 5) to group the accounts from U_N into communities belonging to k new fraudsters. It then continues to iterate over newly discovered subjects, reviewed by these new fraudsters or by the previously known fraudsters (line 6), and over newly identified fraudsters, e.g., using the techniques described in Section 6.9.

6.9 Fraud De-Anonymization Oracles

We observe that ASO workers know the user accounts that they control, and introduce a novel approach to validate fraud de-anonymization solutions, that converts human workers into FDA oracles. In Section 6.10 we use this approach to evaluate UODA.

Algorithm 3 outlines our validation protocol, where m, n, q are integer parameters. The protocol consists of 2 main interaction steps. In the first step, we ask each participant, i.e., recruited human fraud worker, to reveal m user accounts that they control in Google Play, by sending their Google e-mail addresses associated with these accounts (Algorithm 3, line 1). We then use a depth-2 breath first search

Algorithm 3: Interaction protocol with human fraud workers, to provide ground truth performance evaluation for fraud de-anonymization algorithms.

```
Input  : $P$; # User study participant;
         $m$, $n$, $q$ ; # Numbers of accounts
Output: $A[]$; # Accounts attributed to $P$;
1  $A$ = $P$.revealAccounts($m$);
2  $Data[]$ = BFS($A$, 2);
3  $newAccounts[n]$ = FDA($A$, $Data$);
4  $ACAccounts[q]$ = genAttentionCheckAccounts();
5  $Q$ = genQuestionnaire($newAccounts$, $ACAccounts$);
6  $Answers$ = send($A$.randomAccount(), Q);
7  if Answers.passAttentionCheck() then
8      if newAccounts.getConfirmed().verifyOwnership() then
9          $A$.add(newAccounts.getConfirmed());
10     end
11 end
12 return $A$
```

approach to collect (1) all the apps reviewed by the m accounts and (2) all the reviewers of these apps (line 2). We apply a fraud de-anonymization solution (see next section) to identify n new, *candidate accounts*, i.e., other Google Play accounts suspected to be controlled by the same participant (line 3).

For the second interaction step, we have designed a questionnaire that asks the participant to confirm if they control each of these n candidate accounts, see Figure 6.1. Specifically, for each account, we show the account's profile photo and name, and ask the participant if they control the account. We provide 3 options, "Yes", "No" and "I don't remember".

Participant validation. We have proposed the following tests to validate attention and honesty:

- **Attention check**. In addition to the n candidate accounts, we add to the questionnaire q other *test* accounts (line 4), for which we know the answer: (1) accounts that we know that the participant controls, i.e., picked randomly from

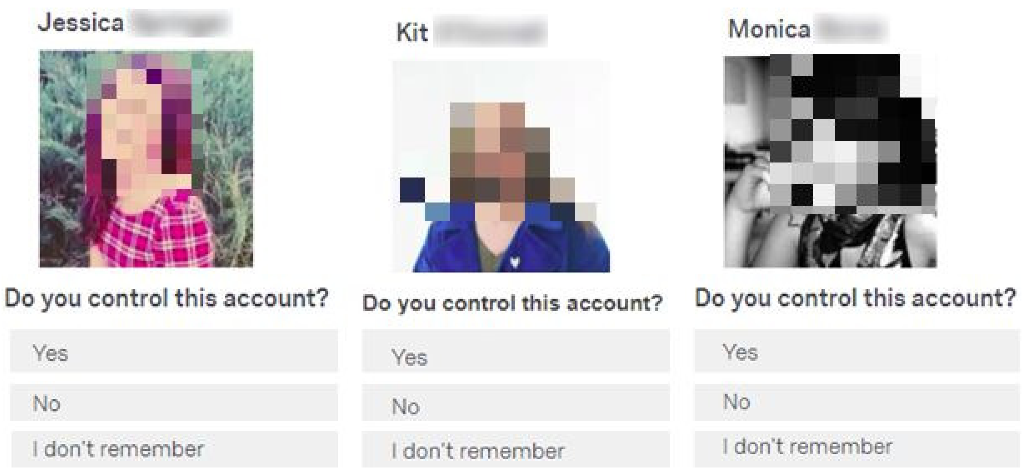


Figure 6.1: Anonymized screenshots of 3 questionnaire pages, for accounts (left) revealed in step 1 to be controlled by the participant, (center) synthetic account not controlled by participant, and (right) detected by UODA to be controlled by the participant.

among the m accounts revealed in the first step, and (2) synthetic accounts that we know that the participant does not control, i.e., accounts that we either created or extracted from Google Play and that has significant activities in Google Play (photos, videos, followers). We then present the questions for the $n + q$ candidate and test accounts, in randomized order (line 5).

• **E-mail knowledge**. Each Google Play account A has an associated e-mail address E. Given E, one can easily retrieve the account A. However, E is not public, and, given only knowledge of A, one cannot find E. We leverage this observation to ask each participant to reveal the e-mail address E of each Google Play account A that they claim to control. We use E to find the corresponding account A'. The participant fails this test if A' does not exist or $A' \neq A$.

• **E-mail based validation**. To certify that participants control the accounts that they claimed to control, we pick a random account from among the m accounts revealed in the first step, and send the questionnaire (line 6).

• **Token and e-mail based validation**. To verify ownership of accounts confirmed in the questionnaire (line 8), we choose randomly one of the n accounts

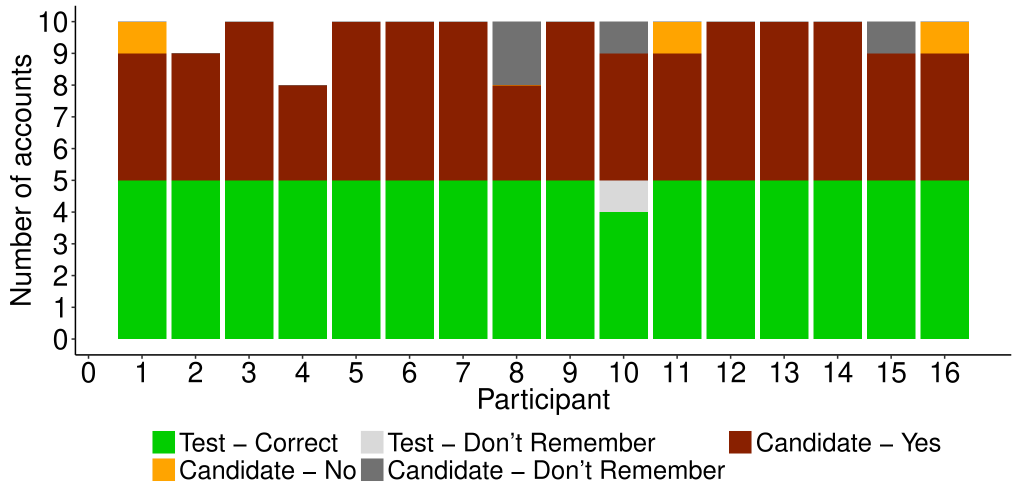


Figure 6.2: Results of UODA on data validated by 16 human fraud worker participants. UODA achieves an overall precision of 91%.

confirmed, and send to its corresponding e-mail address, a random, 6 character code. The accounts verify if and only if the participant can retrieve the code.

6.10 User Study

We have recruited 16 ASO workers from India (4), Bangladesh (4), UK (2), Egypt (2), USA (1), Pakistan (1), Indonesia (1), and Morocco (1), 12 male and 4 female, who claimed to control between 40 to 500 accounts (M=211, SD=166). We have used these participants to evaluate the performance of UODA. We have set m=10, n=5 and q=5, thus each participant reveals 10 accounts controlled in Google Play, then further confirms or denies control of 5 other UODA detected accounts, and 5 test accounts. To run UODA, we have used the 10 accounts revealed by each participant in the first step, to collect (via BFS) 718 apps, 265,724 reviewers and 341,993 reviews in total. We collected up to 175 apps, 37,056 reviews and 22,848 reviewers from a single worker. We pay $10 to each participant.

Ethical considerations. We have developed IRB-approved protocols to ethically interact with participants and collect data. We have not asked the participants to post fraud on Google services. We restricted the volatile handling of emails and photos of accounts revealed by participants, to the validation process. We have immediately discarded them after validation. We believe that this information cannot be used to personally identify fraudsters: recruited fraudsters control between 40-500 accounts each (M=211, SD=166) thus any such account is unlikely to contain PII. Further, since we do not preserve these emails and photos, their handling does not fall within the PII definition of NIST SP 800-122. Under GDPR, the use of emails and photos without context, e.g., name or personal identification number, is not considered to be "personal information".

In the following we first detail the instantiation of UODA that we evaluated, then describe the results of the user study.

6.10.1 UODA Parameters

We evaluate UODA (see § 6.3) using two features, defined by the sets (1) $C_{l\geq} = \{(s,s') \in S_l \mid cr(s,s') \geq b_1\}$, where $cr(s,s')$ is the number of reviewers shared by subjects s and s' and (2) $U_{l\geq} = \{s \in S_l \mid u_l(s) \geq b_2\}$,where $u_l(s)$ is the number of accounts controlled by worker W_l who has reviewed subject s. Specifically, these features define the family of sets $\mathcal{F}_{W_l}$ with m=4:

$$\begin{aligned}
\Omega_{l1} &= \{s \in S_l \mid s \in C_{l\geq} \setminus U_{l\geq}\} \\
\Omega_{l2} &= \{s \in S_l \mid s \in U_{l\geq} \setminus C_{l\geq}\} \\
\Omega_{l3} &= \{s \in S_l \mid s \in C_{l\geq} \cap U_{l\geq}\} \\
\Omega_{l4} &= \{s \in S_l \mid s \in (C_{l\geq} \cup U_{l\geq})^C\}
\end{aligned} \tag{6.8}$$

The rationale behind this selection of Ω sets is that fraudsters are hired to provide large number of reviews for different subjects. Thus, a fraudulent account u

controlled by a fraudster profile $(W, U, S) \in W^*$ is more likely to post reviews for subjects that were reviewed by other accounts under its control, see e.g. [JL08, MLG12, TMG+13, ZXL+18].

6.10.2 Results

Figure 6.2 shows that 15 of the 16 participants have provided correct responses to all 5 test accounts. The remaining participant answered "I don't remember" for a single test account, known not to be controlled by the participant. We have thus decided to keep the data from all participants. Further, for participants 2 and 4, UODA found less than 5 suspected accounts (i.e., 4 and 3 respectively).

We observe that 10 out of 16 participants have confirmed control (and passed our verification) of all UODA proposed accounts. 5 participants confirmed control of 4 out of 5 UODA recommended accounts and 1 participant confirmed control of only 3 accounts out of 5 UODA recommended accounts. UODA's precision ($\frac{TP}{TP+FP}$, where TP is the number of true positives and FP is the number of false positives) is thus 91%, i.e., 7 unconfirmed accounts among 77 predicted. We note that for 3 out of the 7 unconfirmed accounts, the participants did not remember if they control them or not.

6.11 Empirical Evaluation

6.11.1 Attributed Account Data

We have recruited an additional set of 23 fraud workers and performed only the first step of the fraud de-anonymization validation protocol of § 6.9, where we asked each participant to reveal at least 15 accounts that they control in Google Play.

Figure 6.3 shows the number of accounts (bottom, red segments) revealed by each of the 23 workers, between 22 and 86 accounts revealed per worker, for a total of 942 attributed fraud accounts.

We have selected the top 640 *fraud apps*, that received the highest percentage of reviews from accounts controlled by the 23 fraudsters, and crawled their reviews once every 2 days, over a 6 month period. The 640 apps had between 7 to 3,889 reviews. Half of these apps had at least 51% of their reviews written from accounts controlled by the 23 fraudsters. On the whole, the 640 apps have received 159,469 reviews, of which 17,575 were written from the above 942 attributed fraud accounts.

In the following, we use this data to evaluate the ability of developed solutions to (1) attribute *unknown* accounts to existing seed workers and (2) reveal hidden relationships among reviewers towards uncovering previously unknown fraudulent workers.

6.11.2 DeepCluster Parameter Tuning

We have built the union fraud graph over the user accounts who reviewed the 640 fraud apps. To run DeepWalk, we transform this union fraud graph into a non-weighted graph, where we replace an edge between nodes u_i and u_j with weight $w_{ij} = w(u_i, u_j)$, by w_{ij} non-weighted edges between u_i and u_j. This ensures that the probability of DeepWalk choosing node u_j as next hop while at node u_i is proportional to w_{ij}. The resulting union fraud graph has 56,950 nodes and 34,742,730 edges (5,858,940 unique edges) and consists of 202 disconnected components.

Algorithm 4 shows the pseudocode for the grid search process that we used to identify the best performing DeepWalk parameters on the union fraud graph: $d = 300, t = 100, \gamma = 80, w = 5$, see § 6.11.6. d is the number of dimensions when

Algorithm 4: DeepWalk parameter tuning. For each parameter set, compute Deepwalk embeddings on the union fraud graph and run stratified cross validation (SCV) using a learning algorithm Alg and only seed accounts as part of the training and validation set (lines 3-5). We save the best performing configuration (lines 6-8).

```
   Input  : $CRG$ # Co-review Graph
            $S$ # seed accounts
            $Alg$ # learning algorithm
   Output: $DWParams$ # Best DeepWalk parameters
1  $F_{max} = 0$, $DWParams = \emptyset$
2  $ParamSet = \text{Generate.Grid}(\{t, d, \gamma, w\})$
3  for $p \in ParamSet$ do
4      $D = S \ltimes CRG.DWFeatures(p)$
5      $F = \text{SCV}(D, Alg)$
6      if $F > F_{max}$ then
7          $DWParams = p$
8      end
9      $F_{max} = \max\{F, F_{max}\}$
10 end
11 return $DWParams$
```

representing nodes in the graph, t is the maximum length of a random walk, γ is the number of random walks started from each node, and w is the the number of neighbors used as the context in each iteration of its SkipGram component.

We have used K-means as clustering algorithm in DeepCluster (see § 1) considering that we have prior knowledge about the number of workers who targeted each subject. We identified the optimum K value required by K-means for each subject s_i experimentally, as follows. Iterate for values of K ranging from 2 to $|W_i|$ where $|W_i|$ is the number of distinct workers known to have targeted subject s_i. Since K-means is susceptible to local optima, we run it 100 times on the embeddings of the co-review graph of subject s_i, and assess the quality of the returned clusters. We use a quasi-F1 score that gages how good a cluster configuration is with regards to our ground truth. We also adjust for the number of accounts in each cluster and

Approach	**Algorithm**	**Precision**	**Recall**	**F1**
UODA	Top 1	85.11%	82.59%	83.83%
	Top 2	92.05%	90.32%	91.11%
	Top 3	94.23%	92.91%	93.57%
DDA	KNN	94.28%	93.35%	93.81%
	MLP	94.90%	94.10%	94.50%
	RF	94.37%	93.31%	93.84%

Table 6.1: Performance of UODA and DDA on ground truth data set. DDA performs better. However, with only 2 features, UODA reaches an F1 of 83%.

compute the weighted average across all clusters in one cluster configuration.

6.11.3 Fraud De-Anonymization

We compare the ability of the UODA and DDA algorithms to de-anonymize the ground truth attributed account dataset of § 6.11.1. For this, we first set randomly aside 75% of the seed accounts from each worker into a set G_T (Ground Truth) and let the remaining 25% accounts be the T_T (Testing Truth) set. For DDA, we train the co-ownership predictor using accounts in G_T, then apply the predictor to all accounts in T_T and extract as features the number of nodes in each class (known fraudster) to whom the account has a link according to the co-ownership predictor. Finally, we train a classifier on these features using stratified 10-fold cross validation.

For UODA, following the G_T/T_T split, we compute the Ω sets as described in (6.8) using accounts in G_T and test the algorithm on the review histories of all accounts in T_T. We fix the same $b_1 = 10$ and $b_2 = 15$ (obtained through a grid search) across all the workers. Then, given an account u in T_T, we select as candidate the worker whose partition maximizes the function in Equation 6.4, i.e., we evaluate such function 23 times (one for each worker) and attribute u to the worker that maximizes it. Note that to evaluate the function, we need P_i: the popularity

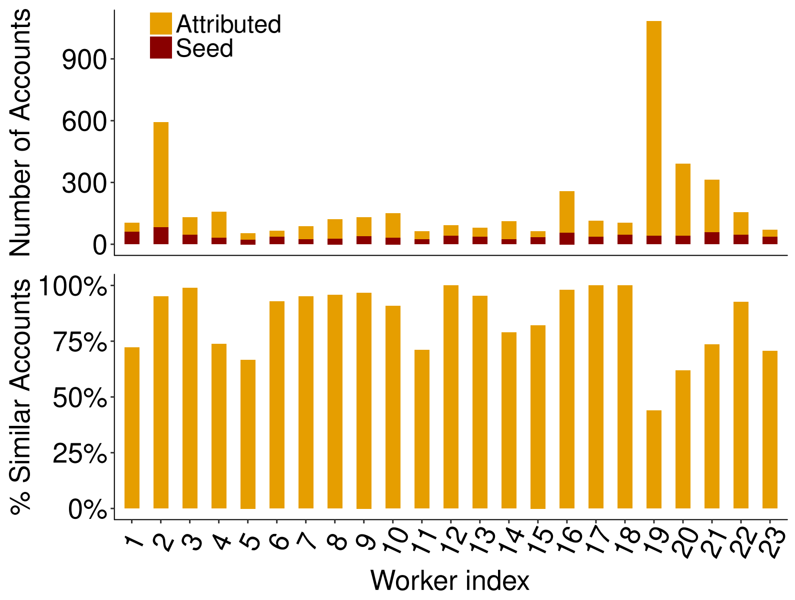


Figure 6.3: (Top) Distribution of seed and DDA attributed accounts across the 23 fraudulent workers. DDA attributed 3,547 accounts to these fraudsters, 3.7 times more than the size of the seed set. (Bottom) Per worker percentage of newly attributed accounts suspected of self-plagiarism. Almost all ($\geq$ 90%) of the newly attributed accounts for 13 out of 23 fraud workers have self-plagiarized reviews.

volume of all the subjects in each Ω_i. We approximate $P_i = \epsilon \sum_{s_j \in \Omega_i} R(s_j)$ where $R(s_j)$ is the number of reviews that subject s_j received from fraudster accounts in the G_T set and ϵ was set to mimic a probability distribution on $\mathcal{S}$. In practice, we have evaluated multiple values for ϵ, and chose $\epsilon = 10^{-6}$ as best performer.

Table 6.1 compares UODA and DDA results after 10 different random G_T/T_T splits. We observe that DDA achieves an F1 measure of 94.5%, outperforming UODA's top 1 choice. UODA's performance, however, significantly increases when allowed to make mistakes. Specifically, Top 2 UODA achieves an average F1 of 91.11% while Top 3 UODA achieves an average F1 of 93.57%.

Fraud Attribution in the Wild. We have further trained DDA on all the ground truth information (both G_T and T_T sets). We then applied the trained DDA to 3,681 accounts that appeared in at least one seed cluster but never appeared in an unknown cluster of the 640 suspicious apps (§ 6.11.1). Figure 6.3 (top) shows the distribution of 3,547 of these accounts attributed to the 23 fraud workers. Only 134 accounts were not assigned to any fraud worker. To validate this result, we computed the review's Jaccard similarity between each newly attributed $\hat{U}_l$ account and all seed U_l accounts, using the review's $k-$shingle representation as defined in [Bro97].

Figure 6.3 (bottom) shows the proportion of newly assigned accounts $u \in \hat{U}_l$ that have at least one review similar ($J(R_{\hat{u}}, R_u) \geq 0.5$) to those of accounts in its respective seed set. We have set $k = 3$ and considered only reviews with at least 10 characters in length. We observe that 13 out of 23 fraud workers have around 90% of their new attributed accounts with similar reviews to the ones written by its seed accounts. Likewise, 22 out of 23 fraudsters have at least 50% of their accounts with similar reviews. These results confirm DDA's outcome and previous work on crowdsourced review manipulation, e.g., [KCA17].

6.11.4 Co-Ownership Predictor

We evaluate the performance of the co-ownership predictor *cowPred* of Section 6.5, and compare it against ELSIEDET's state-of-the-art solution [ZXL+18]. For this, we build training data as follows. First, create complete graphs from among seed attributed accounts found in clusters across all the product space, i.e., create a link (u, v) for $u, v \in C_j$ where C_j is a cluster in product j. Then, using the 942 accounts of § 6.11.1, generate "positive" links (class 1) when both accounts in the link are known to be controlled by the same fraudster and "negative" links (class 0) when controlled by different fraudsters. Finally, for each link (u, v), extract the 16 features described in Section 6.6 and append its class. Our training set consists of 17,695

Solution	**ML Algo.**	**Precision**	**Recall**	**F1**
cowPred	**GBM**	**96.40%**	**96.94%**	**96.67%**
	RF	96.30%	97.01%	96.65%
	SVM	93.75%	95.34%	94.54%
	RLR	93.72%	94.42%	94.07%
	NB	88.44%	95.66%	91.91%
ELSIEDET	Grid search	82.41%	85.92%	84.13%

Table 6.2: Performance of our co-ownership predictor *cowPred* vs. ELSIEDET [ZXL$^+$18] on ground truth data. *cowPred* significantly outperforms ELSIEDET.

pairs of user accounts, 79.5% of which are controlled by the same fraudster.

We use this data to train several supervised learning algorithms and select the top performer as the co-ownership predictor. Specifically, we used several sampling strategies and supervised learning algorithms that train on the features of the co-ownership predictor: Gradient Boosting Machine (GBM), Random Forests (RF), Support Vector Machine (SVM), Regularized Logistic Regression (RLR), and Naive Bayes (NB). We also set aside 20% of the 17,695 links as a test set to assess the quality of the co-ownership predictor after training with 10-fold CV. Further, to evaluate the impact of class imbalance, we compared the *no sampling* strategy against strategies of *undersampling* and *oversampling.* For the *undersampling* strategy, we created a 50-50 training set with 2,901 links for each class. For the *oversampling* strategy, we used the SMOTE algorithm [CBHK02] and created synthetic data along the line segments joining any or all of the k minority class nearest neighbors.

cowPred's results were very similar for the no sampling and oversampling strategies, outperforming the undersampling strategy. Thus, in the following we present results only for the no sampling strategy.

The ELSIEDET co-ownership predictor. We compare *cowPred* against the state-of-the-art ELSIEDET's Sybil social link builder [ZXL$^+$18]. ELSIEDET builds social links between Sybil user accounts based on their similarity: (i) whether their reviews were posted for the same app, (ii) within a fixed time window ΔT, and (iii) were either 1-star or 5-star. Accounts u and v are considered to form a Sybil social link iff $sim(u, v) \geq \beta$, where β and ΔT are parameters. Zheng et al. [ZXL+18] manually tuned these parameters, as they observed that several supervised learning techniques were not sensitive to different thresholds employed. We have improved on this manual tuning process, by implementing a grid search to obtain the best parameters $\Delta T^*, \beta^*$, using the same training set used for our *cowPred* predictor. We compute performance for ELSIEDET based on whether links (u, v) were predicted to be controlled by the same worker.

Comparison results. Table 6.11.4 compares *cowPred*'s performance on the test set, for the best performing supervised learning algorithms evaluated, against ELSIEDET's Sybil social link builder, with best parameters $\Delta T^* = 30$ and $\beta^* = 0.01$. For *cowPred*, GBM and RF achieved the best overall results. *cowPred* significantly outperformed ELSIEDET, with an F1-measure of 96.67% vs. 84.13%. While ELSIEDET was designed for a different type of social network (i.e., Dianping, Yelp), and a different adversary type (elite reviewer), we believe that *cowPred*'s advantage stems from its use of features extracted from common review behaviors exhibited by Sybil accounts. We note that we were not able to compare *cowPred* against other related solutions, e.g., Kumar et al.'s sockpuppet pair detection approach [KCLS17], as they leverage features not available in Google Play, such as community features (downvotes and upvotes).

Feature Insights via Regularized Logistic Regression. In order to understand the impact of and confirm the intuition behind the *cowPred* features (see § 6.5.2), we train *cowPred* on the entire data set (17,695 links) using a regularized logistic regression model [FHT10]. Figure 6.4 shows the relative importance of the

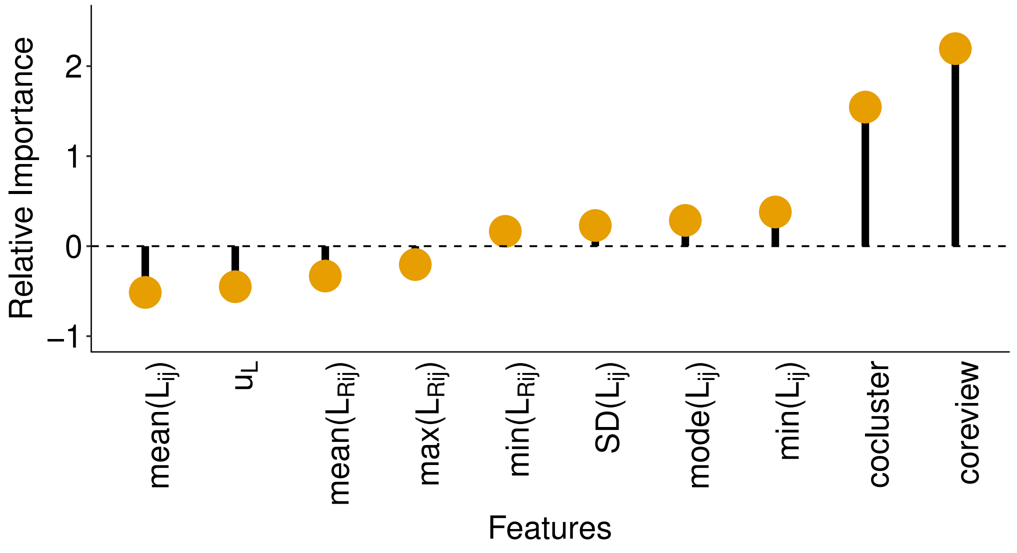


Figure 6.4: Relative importance (shown as $sign(y) * log(1 + abs(y))$) for statistically significant features in the co-ownership predictor using logistic regression. *Co-review* and *co-cluster* have the highest positive impact, while the mean date difference on L_{ij} and the unique lockstep u_{ij} have the largest negative weight.

statistically significant variables after applying Wald Chi-Squared test. We measure importance as the value of the coefficients corresponding to the trained model.

We observe that the co-review and co-cluster features have a strong positive effect on the probability of two accounts being controlled by the same worker. The higher their values the more likely it is that two accounts are owned by the same underlying worker. Similarly, a positive weight for $mode(L_{ij})$ and $min(L_{ij})$ (see § 6.5.2) suggests that if a long period of time between reviews is repeated across most of the commonly reviewed apps then it is more likely that the two accounts are handled by the same worker. However, the unique lockstep feature u_L shows a negative effect, i.e., the larger its value, the less likely it is that both accounts belong to the same worker. Equivalently, contrary to the burstiness assumption, the time difference for all reviews in common are rarely similar. The sign effects of $mean(L_{ij})$ and $SD(L_{ij})$ are less intuitive. We conjecture these sign effects are

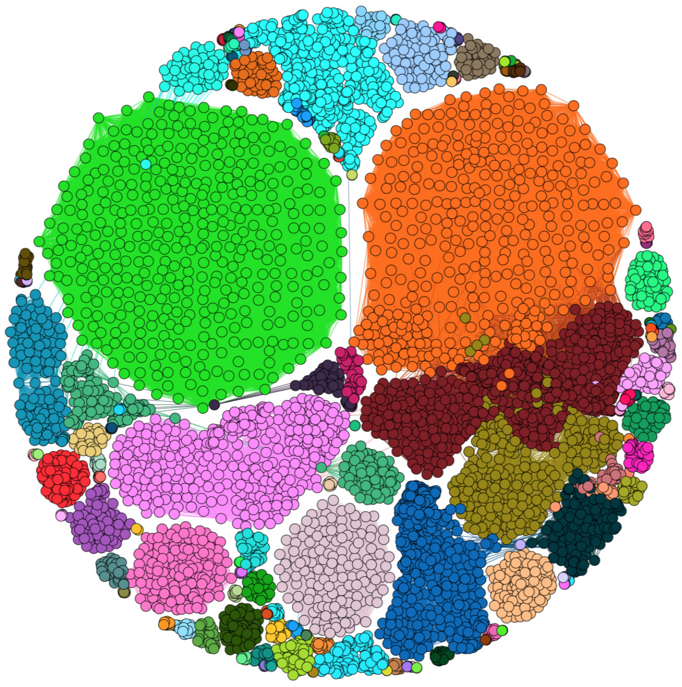


Figure 6.5: Co-ownership (co-w) graph over 5,548 user accounts who reviewed 640 apps involved in fraud. Two accounts are connected if they were predicted to be controlled by the same fraudster. Partition algorithm identified 129 user account components, each potentially controlled by a different fraudster. The largest cluster has 962 nodes and 54 components have more than 10 nodes.

the result of existing correlation across all variables. Further, $mean(L_{R_{ij}})$ impacts negatively the probability of co-ownership. Hence, accounts controlled by the same worker tend to award similar star rating to their commonly reviewed apps. However, we notice that rating features have the least significant effect. This observation implies that most workers post either positive or negative reviews.

6.11.5 Pseudonymous Fraudster Discovery

We applied the $cowPred$ predictor with no sampling strategy and GBM with Bernoulli loss function. We used 279,431 links from 5,690 unknown (un-attributed) user accounts that reviewed 640 suspicious apps. These accounts occurred in clusters without seed accounts (unknown clusters). The resulting co-ownership graph consists

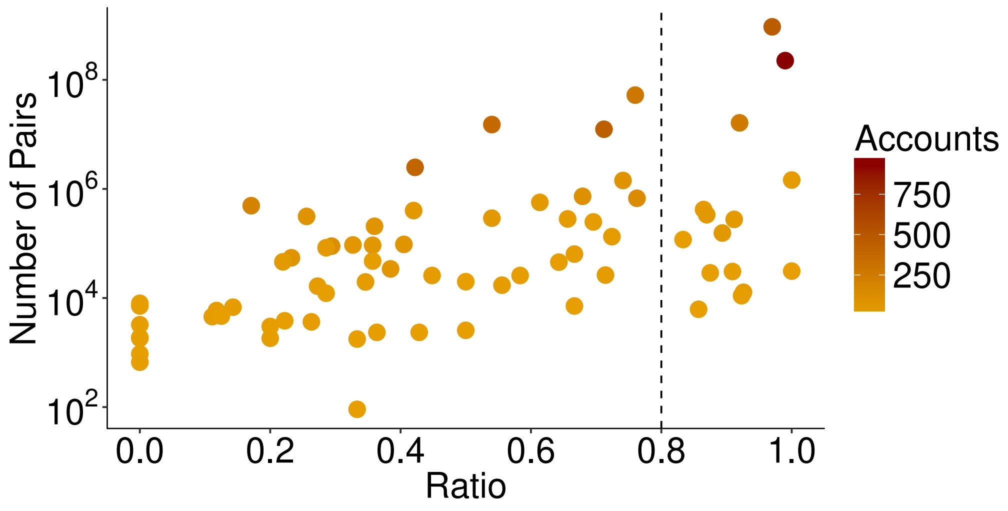


Figure 6.6: Scatterplot for 71 fraudster communities (shown as dots) discovered by PFD: the percentage of users who wrote reviews that are at least 50% Jaccard similar to other reviews (x axis) vs. the number of review pairs (in log scale) in each component (y axis). 15 communities have at least 80% of their user accounts suspected of plagiarism.

of 5,548 user accounts and 97,448 edges. Figure 6.5 shows 129 components identified by PFD. We conjecture that each of these dense components is controlled by a different fraudster. In the following, we validate this conjecture.

Result Validation. We use orthogonal evidence of fraud to validate the dense components of Figure 6.5. Specifically, we inspect reviews' text written by accounts in each cluster. Upon manual investigation, we found many suspicious behaviors, including **singular coincidence**: The review *"this game is Really cute and awesome. I think this is so addicting cause when my kid play this game; i can't resist her to playing it."* was posted from three different accounts in the same component for three different apps on the same date; **the enthusiastic reviewer**: A user account posted the review: *"Try it guys for who never use this app.. I'm enjoy and love app...thanks very much.. because i really enjoy with this app..."* for 40 apps in two days; and **the lazy high-level editors**: We found 12 accounts in one component that used the review *"[App Name] It is very exciting. I like it Nice app! Beautiful* *screenshot. Very interesting It is useful. I like it so much"* as a template to post reviews for 8 apps. The fraudster would tailor this template by adding the name of the app as a prefix.

In addition, similar to the validation in § 6.11.3, we have computed the Jaccard similarity for every pair of reviews using their text's k-shingle representation with $k = 3$. We performed this calculation over each of the 71 detected components with at least 6 accounts. This experiment generated a total of 1.1 billion Jaccard pairs from 118,281 reviews belonging to 5,364 accounts. Moreover, we evaluate the possibility that accounts responsible for reviews with low similarity are generated by accounts not engaged in review manipulation. Specifically, we first computed, a, the number of user accounts in a component that posted reviews with Jaccard similarity at least 0.5 to other reviews in that component. Next, we computed, b, the total number of accounts for each of the selected components. Finally, we computed the ratio a/b. Figure 6.6 highlights fifteen components (1967 users) with ratio greater than 0.8. Very few components have a ratio below 0.3. This result suggests that, even for large components, users that generated very dissimilar reviews are in fact also engaged in review manipulation that reuse high amounts of text.

6.11.6 DeepWalk Based Fraud Attribution Evaluation

We evaluate the fraud attribution capabilities of DeepWalk [PARS14] on the union fraud graph described in section § 6.11.2. In the following, we describe the process we used to identify the parameters that achieve its best performance.

We have run DeepWalk starting from each of the 942 accounts controlled by the 23 workers (that are part of the union fraud graph). Thus, for each such account we extract d DeepWalk features. We then used stratified 10-fold cross validation (each

Algo	**Precision**	**Recall**	**F-measure**
RF	88.3%	85.8%	87.1%
SVM	**90.0%**	**88.7%**	**89.3%**
k-NN	88.0%	86.1%	87.1%
MLP	89.4%	88.3%	88.8%

Table 6.3: DeepWalk performance with several supervised learning algorithms ($d = 300$, $t = 100$, $\gamma = 80$, and $w = 5$). SVM has consistently outperformed the other algorithms in all our subsequent experiments.

fold contains one tenth of the accounts controlled by each worker) to evaluate the ability of standard supervised learning algorithms to use these features to attribute accounts to the workers who control them.

We note that our experiments with different number of random walks per node, showed no significant difference performance changes when γ ranged from 12 to 137. In the following experiments we set γ=80.

Supervised learning algorithm selection. We conducted a random search on the DeepWalk's parameters (walk length t, embedding size d and window size w) and SVM consistently achieved the best performance. DeepWalk and the polynomial SVM combination achieved the best performance for $d = 300$, $t = 100$, $\gamma = 80$, and $w = 5$ (see next paragraphs). Table 6.3 shows the performance of several supervised learning algorithms in attributing fake accounts to the fraudsters who control them, when using DeepWalk extracted features. Thus, in the following we only use SVM for the DeepWalk features.

Random walk length (t). We have evaluated DeepWalk's fraud attribution performance under randomly chosen values for the walk length, ranging from 33 to 175. We have set d=64, γ=10 and w=5. Table 6.4 shows the mean precision, recall and F-Measure under different values, over the 23 workers. It shows that a longer walk length does not imply better performance: the F-measure peaks at $t = 117$. We

t	**Precision**	**Recall**	**F-meas.**
33	72.7%	71.3%	71.9%
45	74.2%	73.3%	73.7%
71	79.7%	77.9%	78.8%
117	**81.9%**	**80.3%**	**81.1%**
126	81.0%	79.5%	80.2%
143	80.9%	79.5%	80.2%
175	82.0%	80.1%	81.1%

Table 6.4: DeepWalk performance for several t values. Best performance at $t = 117$.

d	**Precision**	**Recall**	**F-meas.**
100	83.3%	82.3%	82.8%
104	83.0%	81.6%	82.3%
108	84.3%	82.8%	83.5%
180	87.0%	85.8%	86.4%
198	87.1%	85.5%	86.3%
298	88.7%	87.5%	88.1%
434	**90.7%**	**89.3%**	**90.0%**

Table 6.5: DeepWalk performance for several d values; Best achieved at $d = 434$.

conjecture that as fraudster controlled accounts are close in the co-review graph, longer walks stray from the community controlled by the fraudster and capture information also from other communities. Thus, in the subsequent experiments we set $t = 117$.

Representation size (d). We have evaluated DeepWalk when setting its output representation size (the number of feature) to random values ranging from 100 to 450. Table 6.5 shows the mean precision, recall and F-measure over the 23 workers for various values of d. It shows that a longer representation size is beneficial. Thus, in the following experiments we set d to the maximum we evaluated, 434.

Window size (w). We have also evaluated DeepWalk for several window size values, ranging from 5 to 100. Table 6.6 shows that larger window sizes lead to lower performance. We conjecture that this occurs since a larger context includes

w	**Precision**	**Recall**	**F-meas.**
5	**90.7%**	**89.3%**	**90.0%**
10	89.3%	88.0%	88.7%
15	88.3%	87.1%	87.7%
20	87.0%	85.5%	86.2%
25	85.6%	83.9%	84.7%
50	78.7%	76.4%	77.5%
100	69.0%	66.5%	67.7%

Table 6.6: DeepWalk performance for several w values; Best achieved at $w = 5$.

also information about nodes controlled by other users, either honest or fraudulent. Thus we set $w = 5$ as best performing window size.

6.12 Discussion and Limitations

Underground fraud markets. If successful, the fraud de-anonymization approach proposed in this thesis may drive fraudsters to underground markets. This is however compatible with our objectives, to degrade fraudster capabilities and real-life impact. Further, we observe that DETEGO's ground truth collection and solution validation approach, identifies and leverages intrinsic vulnerabilities in the developer-to-fraudster interactions, i.e., developers need to verify claimed fraudster expertise and fraudsters need to make a profit. Even underground markets need to provide basic functionality that includes worker expertise and developer reputation verifications, and payment mechanisms. When underground fraud markets become accessible to regular developers, they will also be accessible to researchers, who can exploit the same vulnerabilities for ground truth collection and fraud de-anonymization validation purposes.

Evasion strategies. Fraudsters can try to game the DETEGO system. For instance, a fraudster can use multiple sets of disjoint accounts and never use them while reviewing the same app. We observe however that DETEGO introduces a tradeoff between the fraud operation's efficiency and its detectability. Decreasing account reuse decreases profits, as reputable accounts are often preferred in search rank fraud jobs [ZXL+18, SWE+13, CDHH18]. Increasing account reuse exposes the fraud operation to DETEGO detection and attribution. Thus, DETEGO forces fraudsters to minimize account reuse and reduces review fraud incentives.

Further, an adversarial developer who wants to boost the average rating of her app, needs to commision a number of fake reviews that is linear in the number of the app's honest reviews [RCB+14]. Such behavior however affects the temporal distribution of the app's reviews [RCB+14], which makes it detectable, i.e., through the inter-review-time and rating-difference features of DETEGO.

Importance of seed fraud data. DETEGO can effectively provide fraud de-anonymization only in the presence of seed ground truth information about accounts controlled by known fraudsters. Future work may explore the ability of cross-site identity linking attacks [AGL17, BBG+16, SYBT15, ZL13, JKJ13] (see § 3) to e.g., link reviews of detected Sybil communities to public profiles of crowdsourcing accounts.

Informed consent. To recruit 16 participants for the user study of Section 6.10, we have contacted 320 fraud workers. This small turnout may be due to a combination of factors, that include deserted accounts, lack of interest, and the online consent form used as part of our IRB approved validation process. We note that the 16 participants were honest (a single "I don't remember" among 80 test accounts). Future work may investigate the use of IRB approved deception to evaluate the impact of the consent form on the number of participants, their honesty, and the precision of fraud de-anonymization algorithms.

We believe that realization of consequences will not be a major factor in the recruitment process. Our results suggest that reward driven participation is enough for certain fraudsters. Proofs of expertise are normal in crowdsourcing sites, where they enable developers gain confidence when hiring workers. Thus, DETEGO's data collection (or variations) can blend in with regular recruitment of fraud. Further, the use of deception may increase the probability of successful recruiting.

Fraud account memorability. Search rank fraud workers can control hundreds of accounts in the online system, which can impact memorability. However, in our study, participants were able to correctly detect ground truth controlled and non-controlled accounts. The caveat is that we only presented participants with 5 test accounts. Future work should determine the maximum number of questions that we can ask participants, before factors like fatigue and boredom impact their honesty and accuracy.

I.i.d. assumption. UODA assumes that the review history of a fraudulent user account is independent and identically distributed, i.e., that an element in the sequence of reviews is independent of the element that came before it. A possible future work is to explore UODA assuming a Markovian review-posting model.

CHAPTER 7

CONCLUSIONS

Detecting search rank fraud in services like Google Play and Google Maps, is essential for building and relaying an accurate image of user perception of product quality. While Google employs a variety of techniques to detect and obstruct the creation of fraud, communities that specialize in search rank fraud continue to strive and successfully post, e.g., fake reviews. In this thesis, we posit that to be effective, fraud detection and classification efforts need to involve the organizations and individuals who contribute to search rank fraud.

In this thesis we present results from the first structured interview study of 18 ASO workers we recruited from 5 sites, concerning their fraud posting work in Google Play, and also a quantitative investigation with data that we collected from 39 other ASO workers recruited from the same sites. We report Google Play vulnerabilities, and new findings about the capabilities, behaviors and detection avoidance strategies claimed and exhibited by ASO workers.

Taken together, our study in chapter 4 is limited by the difficulty to recruit participants and the sensitivity of the data. The presented findings are hence needed to be understood as situated information and not as generalized facts. Since the nature of fraud detection research involves elimination of risks and vulnerabilities, the presented findings, even with all their limitations, provide new suggestions for future research. Further, given the observed ASO worker ability to adapt, we believe that future research should focus on collecting more such information from diverse sources, to extend and ensure the continued relevance of our findings.

In chapter 5 we have developed RacketStore, the first framework 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 distinguish ASO workers and the more organic exchange workers from regular users of the Google Play service. 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.

In chapter 6 we study the search rank fraud de-anonymization problem and show that it is different from the well studied fraud or spammer detection problem. We model fraud de-anonymization as a maximum likelihood estimation problem and develop an unconstrained optimization fraud de-anonymization algorithm. We introduce a graph based deep learning approach to predict co-ownership of fraudulent account pairs, and use it to build discriminative fraud de-anonymization and pseudonymous fraudster discovery algorithms. Further, we introduce the first protocol to involve human fraud workers in the task of evaluating the performance of fraud de-anonymization algorithms. We show that our solutions achieve high precision and recall on ground truth data, significantly outperform a state-of-the-art approach and are able to attribute thousands of new accounts to known crowdsourced fraudsters.

APPENDIX

Recruitment Material (Chapter 4)

We are researchers from FIU, a university in the US, looking for freelancers with provable App Search Optimization (ASO) expertise in Google Play, willing to participate in a survey. We will ask you questions about your experience working as an app search optimization (ASO) freelancer. We are conducting this survey part of an effort to increase our understanding of how the ASO process optimizes mobile apps.
Your participation in this study is confidential. 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 or not answer questions that you do not feel comfortable answering. If you agree to participate, please send me an e-mail at mrahm031@fiu.edu.

Figure 7.1: Recruitment message sent to each identified ASO worker.

Figure 7.1 shows the recruitment message that we sent to each ASO worker that we identified. Figure 7.2 shows the script that we read to each ASO worker who replied to the recruitment message and qualified for our study.

Thank you for agreeing to participate in this study. My name is Mizanur Rahman, and I am a student at FIU.
In this study, I would like to ask you questions about your experience working as an app search optimization, or ASO, freelancer. The questions will explore your perspectives on ASO strategies in Google Play. The study should take up to 1 hour.
If you decide to participate, you will be one of up to 100 people in this study. We will pay you $5 for every 15 minutes of your time, that is, $20 if we talk for 1 hour.

The benefits of your participation include receiving feedback on vulnerabilities that your strategies may have, and also helping us better understand and model the app search optimization process in Google Play.
Please note that some of the questions that we will ask you, may be upsetting. You can skip any questions you don't want to answer, or stop the study entirely, at any time. 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.
In addition, once we publish our results, other parties, including Google, may use them to try to develop techniques to detect your activities. We note that you already run this risk, even if you do not participate in our study. This is because other developers who hire you, may work for Google, and could use data that they collect from you, to directly impact your activities, e.g., block your accounts or remove the reviews that you write. However, we will never do this.

Please be assured that your participation in this study is confidential. We will keep the records of this study private and 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 you. We will store records 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.
Now, please read the consent form at the following link, `https://fiu.qualtrics.com/jfe/form/SV_8wYphZYyVQ4lTz7`, and tell me if you want to participate in the study. If you want to participate, please click on the button at the end of the form, that says "I consent". Before we begin, do you have any questions?

Figure 7.2: Introduction script read by interviewer to ASO workers who responded to the recruitment message, and qualified for the study, before starting the study.

Survey Questionnaire (Chapter 4)

Screening Questions.

1. How many user accounts do you control in Google Play?
2. For how long have you been active doing App Store Optimization for Google Play?
3. How many jobs have you worked on Google Play review approximately?
4. How much do you charge for a review?
5. How much do you charge for an app install?
6. Are you hired to only provide ratings without reviews? Is this cheaper?
7. Can you tell me what types of ASO services do you provide in Google Play?

ASO Work Process.

1. Assume that I hire you to post several reviews for an app. Can you walk me through your procedure, that is, how do you approach the task?
2. How do you select the devices that you will use?
3. How do you select the user accounts that you will use?
4. Are you already logged in on an account on each device, or do you have to login for this task?
5. Do you log into multiple accounts using the same device? Do you have a fixed list of accounts that you use from each device?
6. Is there a limit on how many accounts you can login from in a single device?
7. Do you need to install the app multiple times, once from each account, or if you install it once you can review it from all the accounts from that device?

   8. Do you type each review directly from the device, or do you write first all the reviews in a file and then you copy-and-paste them to each device?

   9. Do you have a routine (or procedure) to make sure that you use all these devices when writing reviews? For instance, do you go through the devices in a certain order?

1. Assume now that tomorrow I will hire you to post reviews for another app.

   2. Will you pick the same devices and accounts as in the first job? Will you post the reviews in the same order, from the same devices and user accounts, as in the first job?

   3. How else do you make sure that you don't write reviews from the same account twice, or that you don't forget to use one of your accounts and devices in the process?

Job Requirement.

1. Have you seen jobs that tell you for how long to post reviews, that is, for how many days? *If the answer is yes:*

   2. How many such jobs have you seen in the past month?

   3. What is the longest such time interval that you have seen in a job?

4. What do you think is the largest number of reviews that a freelancer could write for a single app?

5. Have you seen jobs that ask you to post a certain number of reviews per day? *If the answer is yes:*

   6. How many such jobs have you seen in the past month?

7. What is the average number of reviews that you think is better to post in a day for a single app?
8. Do you post them all at the same time, or at different times during the day?
9. Can you tell if an app that you are reviewing has been recently launched?
10. Have you seen jobs where developers/employers ask freelancers to post reviews for recently launched apps?
11. How many such jobs have you seen in the past month?
12. Were you ever re-hired by the same developer to review the same app at a later time? For instance, 2 weeks or 6 months after the first job?
13. How many times?
14. Were you ever hired by a developer to review more than one app?
15. Do you provide services that ensure that the rating of the app will stay above a threshold, for instance, above 3.5 or 4 stars?
16. Have you seen such jobs? How many in the past month?
17. How many such jobs have you worked on in the past month?
18. How much do you charge for this service?

Devices.

1. What devices do you use frequently?
2. Do you ever/do you prefer to use a browser to write reviews? or do you use emulator? What about virtual machines?
3. How many mobile devices do you have?
4. How do you get these devices? Are they old and cheap or new and expensive, or a mix of old and new?

5. How many brands of devices do you have?
6. Given that you control so many devices, how do you manage them so that you can easily access them physically? Do you keep all of them in a box, or on your desk, or on a rack?
7. Can you access them remotely, or do you need to manually access each of these devices?
8. Device compatibility: Did it ever happen that you were unable to install an app on the devices that you have - what did you do then?

User Accounts.

1. What do you do when the number of reviews requested is higher than the number of accounts you have?
2. Do you outsource?
3. Do you create new user accounts?
4. Do you purchase accounts?
5. Some people think it is important to create new user accounts when they start a new job. How many new accounts would you create on average per job?
6. How often do you our you team create new accounts?
7. Is there any effective way to pick a good name for a user account?
8. Are the names that you choose very different from each other?
9. Do you ever use a random name generator?
10. How do you get the pictures for the accounts you create?
11. One participant mentioned that they have purchased user accounts from a third party. Have you ever done that?

   12. Do you purchase such accounts when you start a new job?
   13. Do you purchase such accounts at random times?
   14. How many new accounts do you think you have purchased in total?

15. Do you access your Google Play accounts regularly? How do you do that?
   16. Do you access them manually?
   17. Do you use any script to access and maintain their accounts?
   18. Is it necessary to use proxy or VPN services to access accounts?
19. Since freelancers have so many user accounts in Google Play, what are common strategies to remember passwords for these accounts?
   20. Do you write the passwords down?
   21. Do you use the same password for multiple accounts?
   22. Do you use passwords that are easy to remember?
23. For how long do you usually use a Google Play account on average? Do you ever abandon a user account that you use in Google Play?
24. Do you ever create an account just to post one review and then you abandon the account?

Validation.

1. Do you ever need to verify newly created Google Play accounts?
   2. What is the preferred way to verify those accounts? Email or Phone?
   3. Do you buy and use virtual phone numbers, like Twilio or Google Voice?
   4. Do you activate the accounts' google plus profiles?
   5. Why sometimes the account's name is "A Google User"?

6. Do you upvote reviews written by you or your team from other accounts?
7. Did any employer ask you to prove that you posted the reviews? How do you provide such proof?

Avoidance Strategies.

1. Did you ever have a Google Play account that was deleted by Google?
2. How many times did this happen to you in the past two weeks?
3. Has Google Play ever erased reviews that you posted?
4. Do you remember how many times this happened in the past 2 weeks? From your experience, how long does it take for other people to see your reviews?
5. Do you write reviews for apps that you were not hired to review?
6. How do you select those apps?
7. Do you write reviews for apps that you personally like?
8. Do you write reviews for apps that you pick at random?
9. Do you ever use only new accounts to post reviews for an app in a job?
10. Do you use a mix of old and new accounts to post reviews for an app in a job?
11. How often did you do this in your past 5 jobs?
12. Have you seen jobs that explicitly request to use only old accounts that have a good reputation?
13. How many such jobs did you noticed in the past 1-3 month(s)?

App Installation.

1. Do you recommend using the app before reviewing it, or is it fine if you review it right after you install it?

2. Do you wait sometime between installing the app and reviewing the app? For how long?
3. Do you open the app before writing the review? Have you worked on jobs that asked you to interact with the app, before writing the review? For how long? How many such jobs did you get?
4. How long do you keep the app after posting the review?

Review Writing.

1. Do you write the text of the reviews yourself?
2. Do you ever get the review text from the developer? How many times in the past 2-3 weeks?
3. Have you ever seen jobs that ask you how long the review text should be? Do you remember that value?
4. Have you seen people copy parts of reviews that they posted for other apps?
5. Have you seen people copy parts of reviews written by others?
6. How long is an average review that you write, in terms of the number of words or characters?
7. Have you seen people copying reviews that they wrote in an old job to a new job?
8. Do you write reviews in first or third person?
9. Can you give me some example of review text that you write usually?
10. Were you ever hired to write bad reviews, with 1 or 2 star ratings? Do you remember how many such jobs you did?
11. What percentage of your reviews had 4 or 5 stars?

12. Do you ask someone else to post the reviews that you wrote?
13. Do you ask someone else to post the reviews that you wrote?
14. Do you ever use a script to post the reviews?
15. Do you know how to post a review for an app without installing the app?

Collaboration.

1. Do you work individually or are you a part of a team or business?
2. If part of a team, how large is your team or business?
3. How do people in the team communicate?
4. How do they distribute the profit?
5. Do you maintain any work hierarchical levels in your team? For example, do you have dedicated employees who find clients, write reviews, post reviews, validate accounts, get mobile devices, create accounts?
6. Do you know freelancers who communicate with each other to post reviews for the same job?
7. Do you know if freelancers communicate among them about when they will post their reviews?

General Questions.

1. What is your age?
2. What is the highest degree or level of education you have completed?
3. Do you have another job besides freelancing? Can you tell me what it is?

Recruitment Material (Chapter 5)

We have used the following recruitment message:

"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 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 most four weeks. 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?"

Survey Questionnaire (Chapter 5)

1. What is your age?
2. Where do you live?
3. How long have you been working on this type of job?
4. Are you the sole user of the device where you installed RacketStore?
5. If he shares with others: What applications do the other people use on the phone?
6. Do you use this device for personal reasons? For instance, do you check your e-mail on it?
7. Do you use this device to log into multiple accounts?
8. Do you use those accounts to post reviews? In which sites, Google Play, Google Maps?
9. How do you manage the id and passwords for many accounts?
10. Is there a limit on how many google accounts you can login from in a single device?
11. What made you uninstall the RacketStore app?
12. Have you ever stopped any apps that you installed, via the device settings?
13. If the answer is yes: Why do you do this - for instance, are you concerned that such apps are malware? Why would you not uninstall and just deactivate/stop this particular suspicious app?
14. If the app is stopped does Google still consider this app to be installed? That is, does this still count as a "retention install"?
15. Do you give all permissions requested by the apps that you install?

16. Are there any permissions that you don't like to grant?
17. Can you give examples of such permissions?
18. Are you concerned about installing malware apps on this device?
19. Do you have an anti-virus installed on it?
20. If the answer is yes: Have you ever found that an app that you installed was detected to be malware?
21. Are you concerned about the privacy of your data (contacts, login info, pictures, videos, text messages, location) that is on your phone?
22. Have you installed apps from Android app markets other than Google Play? Which app markets?
23. Can you give examples of apps that you download from such app markets?
24. Do you know why those apps are not on Google Play?
25. If they give example: Why did you install this app? (personal use vs review)

BIBLIOGRAPHY

[ACF13] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion Fraud Detection in Online Reviews by Network Effects. In *Proceedings of AAAI ICWSM*, 2013.

[AGL17] Athanasios Andreou, Oana Goga, and Patrick Loiseau. Identity vs. attribute disclosure risks for users with multiple social profiles. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 163–170, 2017.

[AHG$^+$17] Syed Ishtiaque Ahmed, Md. Romael Haque, Shion Guha, Md. Rashidujjaman Rifat, and Nicola Dell. 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, pages 906–918, 2017.

[air18] How artificial intelligence detects fake reviews. Scitech Europa, `https://tinyurl.com/ycjwtmfw`, 2018.

[Ako18] Tasneem Akolawala. Google Play Store Removes Millions of Fake Reviews and Bad Apps With New Anti-Spam System. Gadgets360, `https://tinyurl.com/ya6g2v9n`, 2018.

[and] Permissions overview. `https://bit.ly/2x4HKiW`.

[and17] Changes to device identifiers in android o. Android Developers Blog, `https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html`, 2017.

[Ank13] Google I/O 2013 - Getting Discovered on Google Play. `www.youtube.com/watch?v=5Od2SuL2igA`, 2013.

[ant20] Privacy, security, and deception. Developer Policy Center, `https://play.google.com/about/privacy-security-deception/user-data/`, 2020.

[App] Appbrain. `https://www.appbrain.com/info/about`.

[AQAA$^+$17] Muhammad AL-Qurishi, Mabrook Alrakhami, Atif Alamri, Majed Alrubaian, Sk Md Mizanur Rahman, and M Hossain. Sybil defense techniques in online social networks: A survey. PP:1–1, 01 2017.

[AR16] App Reviews. http://www.app-reviews.org, Last accessed November 2016.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[AS16] App Such. http://www.appsuch.com, Last accessed November 2016.

[AT12] Mishari Almishari and Gene Tsudik. Exploring linkability of user reviews. In *European Symposium on Research in Computer Security*, pages 307–324. Springer, 2012.

[AV16] Apps Viral. http://www.appsviral.com/, Last accessed November 2016.

[Ban18a] Journalists, activists in Bangladesh arrested under ICT Act for posting on social media. AccessNow, https://www.accessnow.org/bangladesh-ict-act/, August 2018.

[Ban18b] 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, May 2018.

[Bar18] Brian Barrett. Millions of Android Devices are Vulnerable Right Out of the Box. Wired, https://www.wired.com/story/android-smartphones-vulnerable-out-of-the-box/, 2018.

[BBG$^+$16] Michael Backes, Pascal Berrang, Oana Goga, Krishna P Gummadi, and Praveen Manoharan. On profile linkability despite anonymity in social media systems. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 25–35, 2016.

[BBM$^+$14] Elie Bursztein, Borbala Benko, Daniel Margolis, Tadek Pietraszek, Andy Archer, Allan Aquino, Andreas Pitsillidis, and Stefan Savage. Handcrafted fraud and extortion: Manual account hijacking in the wild. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 347–358, New York, NY, USA, 2014. ACM.

[BCI$^+$15] Antonio Bianchi, Jacopo Corbetta, Luca Invernizzi, Yanick Fratantonio, Christopher Kruegel, and Giovanni Vigna. What the app is that? deception and countermeasures in the android user interface. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 931–948. IEEE Computer Society, 2015.

[Blu] Bluestacks. `https://www.bluestacks.com/`.

[Bra18] Kyle Bradshaw. Play store's machine learning based anti-spam system removes millions of reviews per week. 9To5Google, `https://tinyurl.com/ya3b6xjg`, 2018.

[BRBMR17] Dearbhail Bracken-Roche, Emily Bell, Mary Ellen Macdonald, and Eric Racine. The concept of "vulnerability" in research ethics: an in-depth analysis of policies and guidelines. *Health research policy and systems*, 15(1):8, 2017.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[Bro97] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.

[BSLL$^+$16] Prudhvi Ratna Badri Satya, Kyumin Lee, Dongwon Lee, Thanh Tran, and Jason (Jiasheng) Zhang. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2365–2370, New York, NY, USA, 2016. ACM.

[BXG$^+$13] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks. In *Proceedings of the WWW*, 2013.

[CB07] Kathy Charmaz and Linda Liska Belgrave. Grounded theory. *The Blackwell Encyclopedia of Sociology*, 2007.

[CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[CDHH18] Nicholas Confessore, Gabriel Dance, Richard Harris, and Mark Hansen. The follower factory. *The New York Times*, Jan 2018.

[cei] Plan ceibal. `https://www.ceibal.edu.uy/en/institucional`.

[Cip16] Jason Cipriani. Google starts filtering fraudulent app reviews from Play Store. ZDNet, `https://tinyurl.com/hklb5tk`, 2016.

[cul12] Cell phone culture: How cultural differences affect mobile use. CNN Business, `https://www.cnn.com/2012/09/27/tech/mobile-culture-usage/`, 2012.

[CYYP14] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 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, pages 477–488, New York, NY, USA, 2014. ACM.

[DCFJ$^+$14] Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. Paying for likes?: Understanding facebook like fraud using honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 129–136, 2014.

[DM09] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[Dou02] John R. Douceur. The Sybil Attack. In *International workshop on peer-to-peer systems*, pages 251–260, 2002.

[Fer20] Nick Fernandez. It's 2020 and the Google Play Store still has a major fake review problem. Android Authority, =https://tinyurl.com/rgnh9wz, 2020.

[FHT10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[Fiv] Fiverr. `https://www.fiverr.com/`.

[FLCS15] Amir Fayazi, Kyumin Lee, James Caverlee, and Anna Squicciarini. Uncovering crowdsourced manipulation of online reviews. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 233–242, New York, NY, USA, 2015. ACM.

[flu20] Flutter. Flutter, `https://flutter.dev/`, 2020.

[FML+13] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Exploiting Burstiness in Reviews for Review Spammer Detection. In *Proceedings of AAAI ICWSM*, 2013.

[FNG] Fake Name Generator. Your Randomly Generated Identity. `https://www.fakenamegenerator.com/`.

[Fre] Freelancer. `http://www.freelancer.com`.

[Fre18] Freedom on the Net, Bangladesh. Freedom House, `https://freedomhouse.org/report/freedom-net/2018/bangladesh`, 2018.

[FTC] The FTC's Endorsement Guides: What People Are Asking. `https://tinyurl.com/p7hk9uz`.

[gad] Gadgets 360. `https://gadgets.ndtv.com/`.

[gai] Access control. `https://developers.google.com/issue-tracker/concepts/access-control`.

[GGF14] Stephan Günnemann, Nikou Günnemann, and Christos Faloutsos. 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, pages 841–850, New York, NY, USA, 2014. ACM.

[GLP+13] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *Proceedings of the 22nd international conference on World Wide Web*, pages 447–458, 2013.

[GP] Google Play Help – Supported Devices. `https://support.google.com/googleplay/answer/1727131?hl=en`.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[gpla] View & analyze your app's ratings & reviews. `https://support.google.com/googleplay/android-developer/answer/138230?hl=en`.

[gplb] Write a review on google play. Google Play Help, `https://tinyurl.com/yc9stfy3`.

[gsm] Gsmarena. `https://www.gsmarena.com/`.

[GVR] Google Vulnerability Reward Program. `https://www.google.com/about/appsecurity/reward-program/`.

[HGT$^+$17] Danny Yuxing Huang, Doug Grundman, Kurt Thomas, Abhishek Kumar, Elie Bursztein, Kirill Levchenko, and Alex C. Snoeren. Pinning down abuse on google maps. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1471–1479, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[HRRC18] Nestor Hernandez, Mizanur Rahman, Ruben Recabarren, and Bogdan Carbunar. Fraud de-anonymization for fun and profit. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 115–130, New York, NY, USA, 2018. ACM.

[HSB$^+$16a] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. BIRDNEST: bayesian inference for ratings-fraud detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, pages 495–503, 2016.

[HSB$^+$16b] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 895–904, New York, NY, USA, 2016. ACM.

[HTS16] Atefeh Heydari, Mohammadali Tavakoli, and Naomie Salim. Detection of fake opinions using time series. *Expert Syst. Appl.*, 58(C):83–92, October 2016.

[IVD$^+$10] Lilly Irani, Janet Vertesi, Paul Dourish, Kavita Philip, and Rebecca E. Grinter. Postcolonial computing: A lens on design and development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1311–1320, 2010.

[Jak18] Markus Jakobsson. Secure remote attestation. *IACR Cryptology ePrint Archive*, 2018:31, 2018.

[Jan18] Mark Jansen. Here's how the google play store detects fake ratings and reviews. Digital Trends, `https://tinyurl.com/yc5hvyq5`, 2018.

[JCB$^+$14] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Inferring strange behavior from connectivity pattern in social networks. In Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, Arbee L. P. Chen, and Hung-Yu Kao, editors, *Advances in Knowledge Discovery and Data Mining*, pages 126–138, Cham, 2014. Springer International Publishing.

[JHH$^+$20] S. K. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath. Throwing darts in the dark? detecting bots with limited data using neural data augmentation. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1729–1745, Los Alamitos, CA, USA, may 2020. IEEE Computer Society.

[JKJ13] Paridhi Jain, Ponnurangam Kumaraguru, and Anupam Joshi. @ i seek'fb. me': Identifying users across multiple online social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1259–1268. ACM, 2013.

[JL08] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 219–230, New York, NY, USA, 2008. Association for Computing Machinery.

[KAC19] Parisa Kaghazgaran, Majid Alfifi, and James Caverlee. Tomcat: Target-oriented crowd review attacks and countermeasures. In *International AAAI Conference on Web and Social Media, ICWSM*, 2019.

[Kar93] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, 1993.

[KCA17] Parisa Kaghazgaran, James Caverlee, and Majid Alfifi. Behavioral analysis of review fraud: Linking malicious crowdsourcing to amazon and beyond. In *Proceedings of ICWSM*, 2017.

[KCLS17] Srijan Kumar, Justin Cheng, Jure Leskovec, and V.S. Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 857–866, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[KCS18] Parisa Kaghazgaran, James Caverlee, and Anna Squicciarini. Combating crowdsourced review manipulators: A neighborhood-based approach. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 306–314, New York, NY, USA, 2018. ACM.

[KM16] Santosh KC and Arjun Mukherjee. On the temporal dynamics of opinion spamming: Case studies on yelp. In *Proceedings of the 25th International Conference on World Wide Web*, pages 369–379. International World Wide Web Conferences Steering Committee, 2016.

[Kna19] Helen Knapman. Fake five-star review farms are flooding amazon with positive comments, says which? The Sun, `https://tinyurl.com/yafthxdd`, 2019.

[KS18] Srijan Kumar and Neil Shah. False information on web and social media: A survey. *CoRR*, abs/1804.08559, 2018.

[LCM$^+$15] Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Proceedings of ICWSM*, pages 634–637. AAAI Press, 2015.

[LCNK17] Shanshan Li, James Caverlee, Wei Niu, and Parisa Kaghazgaran. Crowdsourced app review manipulation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1137–1140, New York, NY, USA, 2017. ACM.

[LFW$^+$17] Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, and Jidong Shao. Bimodal distribution and co-bursting in review spam detection. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1063–1072, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[LGWM15] Changchang Liu, Peng Gao, Matthew Wright, and Prateek Mittal. Exploiting temporal dynamics in sybil defenses. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 805–816, New York, NY, USA, 2015. ACM.

[LMC$^+$16] Yixuan Li, Oscar Martinez, Xing Chen, Yi Li, and John E Hopcroft. In a world that counts: Clustering and detecting fake social engagement at scale. In *Proceedings of the 25th International Conference on World Wide Web*, pages 111–120, 2016.

[LNJ$^+$10] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 939–948, New York, NY, USA, 2010. ACM.

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets.* Cambridge University Press, New York, NY, USA, 2nd edition, 2014.

[LWG14] Kyumin Lee, Steve Webb, and Hancheng Ge. The dark side of micro-task marketplaces: Characterizing fiverr and automatically detecting crowdturfing, 2014.

[LZ16] Michael Luca and Georgios Zervas. Fake it till you make it: Reputation, competition, and yelp review fraud. In *Management Sciences*, pages 3412–3427, 01 2016.

[Mah19] Sapna Maheshwari. When Is a Star Not Always a Star? When It's an Online Review. The New York Times, =https://tinyurl.com/snwjmdd, 2019.

[MDS$^+$19] Ariana Mirian, Joe DeBlasio, Stefan Savage, Geoffrey M. Voelker, and Kurt Thomas. Hack for hire: Exploring the emerging market for account hijacking. In *The World Wide Web Conference*, WWW '19, page

1279–1289, New York, NY, USA, 2019. Association for Computing Machinery.

[mic12] microWorkers. work & earn or offer a microjob. http://www.microworkers.com/, Retrieved on Nov. 2, 2012.

[MKL+13] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting Opinion Spammers Using Behavioral Footprints. In *Proceedings of ACM KDD*, 2013.

[MLG12] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of ACM WWW*, 2012.

[MLW+11] Arjun Mukherjee, Bing Liu, Junhui Wang, Natalie Glance, and Nitin Jindal. Detecting group review spam. In *Proceedings of the 20th international conference companion on World wide web*, pages 93–94. ACM, 2011.

[MoP] MoPeak. https://mopeak.com/buy-android-reviews/.

[MPG+12] Damon McCoy, Andreas Pitsillidis, Jordan Grant, Nicholas Weaver, Christian Kreibich, Brian Krebs, Geoffrey Voelker, Stefan Savage, and Kirill Levchenko. Pharmaleaks: Understanding the business of online pharmaceutical affiliate programs. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 1–16, Bellevue, WA, 2012. USENIX.

[MVLG13] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What Yelp Fake Review Filter Might Be Doing. In *Proceedings of the International Conference on Weblogs and Social Media*, 2013.

[NA16] Kazushi Nagayama and Andrew Ahn. Keeping the Play Store trusted: fighting fraud and spam installs. Android Developers Blog, https://android-developers.googleblog.com/2016/10/keeping-the-play-store-trusted-fighting-fraud-and-spam-installs.html, 2016.

[NAG+19] Shirin Nilizadeh, Hojjat Aghakhani, Eric Gustafson, Christopher Kruegel, and Giovanni Vigna. Think outside the dataset: Finding fraudulent reviews using cross-dataset analysis. In *The World Wide*

Web Conference, WWW '19, page 3108–3115, New York, NY, USA, 2019. Association for Computing Machinery.

[NLS+17] Shirin Nilizadeh, Francois Labrèche, Alireza Sedighian, Ali Zand, José Fernandez, Christopher Kruegel, Gianluca Stringhini, and Giovanni Vigna. Poised: Spotting twitter spam off the beaten paths. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1159–1174, New York, NY, USA, 2017. ACM.

[NPG+12] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 300–314, 2012.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[OMS16] Jeremiah Onaolapo, Enrico Mariconti, and Gianluca Stringhini. What happens after you are pwnd: Understanding the use of leaked webmail credentials in the wild. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, pages 65–79, New York, NY, USA, 2016. ACM.

[PAD+17] Rebecca S. Portnoff, Sadia Afroz, Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. Tools for automated analysis of cybercriminal markets. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 657–666, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[Par16] TE Parliament. Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union*, 2016.

[Par18] Simon Parkin. The Never-Ending War on Fake Reviews. The New Yorker, =https://tinyurl.com/y84hhrea, 2018.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[PCWF07] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: A fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 201–210, 2007.

[Peo] PeoplePerHour. `https://www.peopleperhour.com`.

[Per16] Sarah Perez. Amazon bans incentivized reviews tied to free or discounted products. Tech Crunch, `https://tinyurl.com/zgn9sq3`, 2016.

[Per19] Sarah Perez. Google Play is changing how app ratings work. Tech Crunch `https://techcrunch.com/2019/05/08/google-play-is-changing-how-app-ratings-work/`, 2019.

[PJM+14] Youngsam Park, Jackie Jones, Damon McCoy, Elaine Shi, and Markus Jakobsson. Scambaiter: Understanding targeted nigerian scams on craigslist. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[PLT09] Joyojeet Pal, Meera Lakshmanan, and Kentaro Toyama. "My child will be respected": Parental perspectives on computers and education in Rural India. *Information Systems Frontiers*, 11(2):129–144, 2009.

[pma20] Packagemanager. , `https://developer.android.com/reference/android/content/pm/PackageManager`, 2020.

[RA15] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 985–994, New York, NY, USA, 2015. ACM.

[Rah18] Md Mizanur Rahman. Search Rank Fraud Prevention in Online Systems. FIU Electronic Theses and Dissertations. 3909. `https://digitalcommons.fiu.edu/etd/3909`, 2018.

[rap] Rapidworkers. `https://rapidworkers.com/`.

[RCB+14] Mahmudur Rahman, Bogdan Carbunar, Jaime Ballesteros, George Burri, and Duen Horng (Polo) Chau. Turning the Tide: Curbing Deceptive Yelp Behaviors. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2014.

[Rei17a] Brian Reigh. Fake reviews on the Play store reportedly growing and getting smarter. *Android Authority*, April 2017.

[Rei17b] Brian Reigh. Fake reviews on the Play Store reportedly growing and getting smarter. Android Authority, `https://tinyurl.com/yc4fo9dk`, 2017.

[Reva] ReviewApp.Mobi. `https://reviewapp.mobi/`.

[Revb] Reviews-Up. `https://reviews-up.com/android-app-reviews/`.

[RHCC18] Mizanur Rahman, Nestor Hernandez, Bogdan Carbunar, and Duen Horng Chau. Search rank fraud de-anonymization in online systems. In *Proceedings of the ACM Conference on Hypertext and Social Media*, 2018.

[RHR+19] Mizanur Rahman, Nestor Hernandez, Ruben Recabarren, Syed Ishtiaque Ahmed, and Bogdan Carbunar. The art and craft of fraudulent app promotion in google play. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2437–2454, New York, NY, USA, 2019. Association for Computing Machinery.

[RJS17] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[RL16] Rank Likes. `http://www.ranklikes.com/`, Last accessed November 2016.

[Ros17] Eli Rosenberg. The Shed at Dulwich' was London's top-rated restaurant. Just one problem: It didn't exist. *The Washington Post*, Dec 2017.

[RRCL17] Mizanur Rahman, Ruben Recabarren, Bogdan Carbunar, and Dongwon Lee. Stateless puzzles for real time online fraud preemption. In *Proceedings of the ACM Web Science Conference (WebSci)*, 2017.

[Sal19] Salary explorer. `http://www.salaryexplorer.com/`, 2019.

[SB13] Kevin Springborn and Paul Barford. Impression fraud in on-line advertising via pay-per-view networks. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 211–226, Washington, D.C., 2013. USENIX.

[SCM11] Tao Stein, Erdong Chen, and Karan Mangla. Facebook Immune System. In *Proceedings of the 4th Workshop on Social Network Systems*, pages 8:1–8:8, 2011.

[SE15] Vlad Sandulescu and Martin Ester. Detecting singleton review spammers using semantic similarity. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 971–976, New York, NY, USA, 2015. ACM.

[SHM13] Thamar Solorio, Ragib Hasan, and Mainul Mizan. A case study of sockpuppet detection in wikipedia. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 59–68, 2013.

[SLK15] Jonghyuk Song, Sangho Lee, and Jong Kim. 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, pages 793–804, New York, NY, USA, 2015. ACM.

[SMJ$^+$15] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. EVILCOHORT: Detecting communities of malicious accounts on online services. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 563–578, Washington, D.C., 2015. USENIX Association.

[SR07] Brad Stone and Matt Richtel. The Hand That Controls the Sock Puppet Could Get Slapped. The New York Times, `https://www.nytimes.com/2007/07/16/technology/16blog.html`, 2007.

[SSGN17] Jessica Su, Ansh Shukla, Sharad Goel, and Arvind Narayanan. De-anonymizing web browsing data with social networks. In *Proceedings* *of the 26th International Conference on World Wide Web*, WWW '17, pages 1261–1269, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[Ste19] Rebecca Stewart. Instagram's fake follower purge has had 'little effect' on fraudulent influencers. The Drum, `https://tinyurl.com/y7ja52h5`, 2019.

[SWE+13] Gianluca Stringhini, Gang Wang, Manuel Egele, Christopher Kruegel, Giovanni Vigna, Haitao Zheng, and Ben Y. Zhao. Follow the green: Growth and dynamics in twitter follower markets. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 163–176, New York, NY, USA, 2013. ACM.

[SYBT15] Giuseppe Silvestri, Jie Yang, Alessandro Bozzon, and Andrea Tagarelli. Linking accounts across social networks: the case of stackoverflow, github and twitter. In *KDWeb*, pages 41–52, 2015.

[TGSP11] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. Suspended accounts in retrospect: An analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, page 243–258, New York, NY, USA, 2011. Association for Computing Machinery.

[TIB+14] Kurt Thomas, Dmytro Iatskiv, Elie Bursztein, Tadek Pietraszek, Chris Grier, and Damon McCoy. Dialing back abuse on phone verified accounts. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 465–476, New York, NY, USA, 2014. Association for Computing Machinery.

[TMG+13] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. 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, pages 195–210, Berkeley, CA, USA, 2013. USENIX Association.

[TMLS09] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association.

[Tot12] Virus Total. Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en*, 2012.

[Tri17] Robert Triggs. Flagship? mid-range? budget? find the best phone for you. AndroidAuthority, `https://www.androidauthority.com/flagship-mid-range-budget-best-phone-815330/`, 2017.

[TSM16] The Social Marketeers. `http://www.thesocialmarketeers.org/`, Last accessed November 2016.

[TZX$^+$15] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and Tong Zhang. Crowd fraud detection in internet advertising. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1100–1110, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[Upw] Upwork Inc. `https://www.upwork.com`.

[VGZ$^+$16] Giridhari Venkatadri, Oana Goga, Changtao Zhong, Bimal Viswanath, Krishna P. Gummadi, and Nishanth Sastry. Strengthening weak identities through inter-domain trust transfer. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 1249–1259, 2016.

[Vir20] Virustotal. , `https://developers.virustotal.com/reference#getting-started`, 2020.

[Wam17] Colleen Wamback. WPI Research Detects When Online Reviews and News Are a Paid-for Pack of Lies. *Worcester Polytechnic Institute*, November 2017.

[War17] Tom Warren. 41 percent of Android phones are vulnerable to 'devastating' Wi-Fi attack . The Verge, `https://www.theverge.com/2017/10/16/16481252/wi-fi-hack-attack-android-wpa-2-details`, 2017.

[WF19] Rolfe Winkler and Andrea Fuller. How companies secretly boost their glassdoor ratings. The Wall Street Journal, `https://tinyurl.com/yc7t2nk4`, 2019.

[WGF17] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. GANG: Detecting Fraudulent Users in Online Social Networks via Guilt-by-Association on Directed Graphs. In *Proceedings of ICDM*, 2017.

[WK05] C. Wei and B. E. Kolko. Studying mobile phone use in context: cultural, political, and economic dimensions of mobile phone use. In *IPCC 2005. Proceedings. International Professional Communication Conference, 2005.*, pages 205–212, July 2005.

[Woo17] Emma Woollacott. Amazon's Fake Review Problem Is Now Worse Than Ever, Study Suggests. *Forbes*, September 2017.

[Woo20] Emma Woollacott. Facebook And EBay Promise Crackdown On Fake Reviews. Forbes, =https://tinyurl.com/yb83nowr, 2020.

[WWZZ14] Gang Wang, Tianyi Wang, Haitao Zhang, and Ben Y. Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, pages 239–254, 2014.

[WXLY11] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. Review Graph Based Online Store Review Spammer Detection. *IEEE ICDM*, 2011.

[Xu13] Chang Xu. Detecting collusive spammers in online review communities. In *Proceedings of the Sixth Workshop on Ph.D. Students in Information and Knowledge Management*, PIKM '13, pages 33–40, New York, NY, USA, 2013. ACM.

[XWLY12] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 823–831, New York, NY, USA, 2012. ACM.

[XZ14] Zhen Xie and Sencun Zhu. Grouptie: Toward hidden collusion group discovery in app stores. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless &#38; Mobile Networks*, WiSec '14, pages 153–164, New York, NY, USA, 2014. ACM.

[XZ15a] Zhen Xie and Sencun Zhu. Appwatcher: Unveiling the underground market of trading mobile app reviews. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, pages 10:1–10:11, New York, NY, USA, 2015. ACM.

[XZ15b] Chang Xu and Jie Zhang. Combating product review spam campaigns via multiple heterogeneous pairwise features. In *Proceedings of the*

2015 SIAM International Conference on Data Mining, pages 172–180. SIAM, 2015.

[XZLW16] Zhen Xie, Sencun Zhu, Qing Li, and Wenjing Wang. 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, pages 374–385, New York, NY, USA, 2016. ACM.

[YA15] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *Proceedings of the 2015 ACM on Conference on Online Social Networks*, COSN '15, pages 97–97, New York, NY, USA, 2015. ACM.

[Yel12] Yelp tries public shaming to discourage businesses from gaming reviews and ratings. Digital Trends, `https://www.digitaltrends.com/social-media/yelp-cracking-down-on-fake-reviews/`, 2012.

[YGKX10] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. *IEEE/ACM Trans. Netw.*, 18(3):885–898, June 2010.

[YHZ$^+$12] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on Twitter. In *Proceedings of the World Wide Web (WWW)*, pages 71–80. ACM, 2012.

[YKA16] Junting Ye, Santhosh Kumar, and Leman Akoglu. Temporal opinion spam detection by multivariate indicative signals. In *Proceedings of ICWSM*, pages 743–746. AAAI Press, 2016.

[YKGF08] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham D. Flaxman. Sybilguard: Defending against sybil attacks via social networks. *IEEE/ACM Trans. Netw.*, 16(3):576–589, June 2008.

[YN18] Fei Ye and Kazushi Nagayama. In reviews we trust, making google play ratings and reviews more trustworthy. Android Developers Blog, `https://tinyurl.com/yb67uy73`, 2018.

[YVC$^+$17] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y. Zhao. Automated Crowdturfing Attacks and Defenses in Online

Review Systems. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1143–1158, 2017.

[YWW$^+$14] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):2, 2014.

[Zee] Zeerk. `https://zeerk.com/`.

[ZL13] Reza Zafarani and Huan Liu. Connecting users across social media sites: a behavioral-modeling approach. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 41–49, 2013.

[ZXL$^+$18] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith Ross. Smoke Screener or Straight Shooter: Detecting Elite Sybil Attacks in User-Review Social Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

VITA

NESTOR HERNANDEZ

2014 - Present	Ph.D., Computer Science Florida International University Miami, Florida
2019	M.S., Computer Science Florida International University Miami, Florida
2012 - 2014	Data Scientist Wunderman Buenos Aires Buenos Aires, Argentina
2011	B.Sc., Applied Mathematics Universidad Simon Bolivar Caracas, Venezuela

PUBLICATIONS AND PRESENTATIONS

M. Rahman, N. Hernandez, B. Carbunar, D. Chau. *Towards De-Anonymization of Google Play Search Rank Fraud.* IEEE Transactions on Knowledge and Data Engineering (TKDE), 2020

N. Hernandez, M. Rahman, R. Recabarren, S.I. Ahmed, B. Carbunar. *The Art and Craft of Fraudulent App Promotion in Google Play.* The 26th ACM Conference on Computer and Communications Security (CCS), London, UK, November 2019.

N. Hernandez, M. Rahman, R. Recabarren, B. Carbunar. *Fraud De-Anonymization for Fun and Profit* . The 25th ACM Conference on Computer and Communications Security (CCS), Toronto, Canada, October 2018.

M. Rahman, N. Hernandez, B. Carbunar, D. Chau. *Search Rank Fraud De-Anonymization in Online Systems.* The 29th ACM Conference on Hypertext and Social Media (HT), Baltimore, July 2018.