FairPlay: Search Rank Fraud and Malware Identification clinched alongside Google Play

Dr. Mohammed Abdul Waheed[1], Zakiya Tahseen[2], Prof. Zohara Begum[3]

Associate Professor[1], PG student[2], Assistant Professor[3]
Department of Computer Science and Engineering[1,]
VTU Centre for PG Studies, Kalaburagi, **India**.

Abstract—Androids open-source accessibility and accomplished achievement has resulted it in becoming the best accepted adaptable operating system. In turn, it has additionally admiring counterfeit behaviors and awful proliferation. To ascertain malware, all the antecedent techniques accept focused on activating assay of app executables and changeless assay of cipher and permissions. This cardboard introduces FairPlay, a malware apprehension framework that discovers and leverages traces larboard abaft by counterfeit developers, to analyze both malware and look rank artifice subjected to apps in Google play. The apprehensive app is articular by the accession of co-reviews, rankings and ratings. Finally accumulation all the activities of the front-end active apps, it achieves over 95 percent accurateness in free the amiable accepted datasets of malware, accepted and counterfeit apps. FairPlay detects abundant counterfeit apps that alike the Google's Bouncer's apprehension technology fails to discover. It has additionally helped in absolute a new blazon of analysis attack namely "co-ercive" –where users are affected into autograph absolute reviews and into additionally to install and analysis added apps.

Index Terms - Google play, android malware,counterfeit behavior, awful proliferation, search rank fraud, coercive reviews.

I. INTRODUCTION

Android has been the most popular open sources mobile operating system, the phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious applications(apps) developers. Fraudster uploads the malware-laden apps and deceptively boosts the search rank and popularity of their apps. The inspiration for such practices is impact: app popularity surges interprets into financial benefits and assisted malware burgeoning. Fake developers very often abuse crowdsourcing sites (e.g., Freelancer, BestAppPromotion, Fiverr), to utilize aggregation of willing laborers to submit deceiving collectively, emulating realistic, spontaneous exercises from inconsequential individuals (i.e., "Crowd-turfing"). This conduct is called "Search rank fraud".

Past portable malware identification fills in need concentrated around changing examination about app executables and additionally, Static dissection from claiming code and permissions. However, Late android malware dissection uncovered that malware evolves fast with sidestep anti-virus instruments.

In this paper, we try to recognize both malware and search rank duplicity subjects in Google play. This mix will be not arbitrary: we posit that pernicious developers depend on search rank duplicity to support those sway about their malware.

Dissimilar to existing solutions, we fabricate this fill in on the perception that fake and pernicious practices take off behind obvious indications for app business sectors. We uncover these terrible acts toward picking crazy such trails. For instance, the secondary cosset of setting up substantial Google play accounts powers, fraudsters will reuse their accounts over survey composing jobs. Making them likely will audit more applications in normal over. General clients' asset imperatives might urge fraudsters will to post positive reviews inside short time intervals. Real clients affected by malware might report card upsetting encounters. For their review increments in the number about asked for permissions starting with particular case versify of the next, which we will call "permission ramps", might demonstrate Favorable with malware. (Jekyll-Hyde) moves.

II. RELATED WORK

In [1] the creators gathered and described 1,200Android Malware tests, and announced the capacity of malware to rapidly advance and sidestep the recognition components of hostile to infection apparatuses. In [2] the creators utilized crowdsourcing to gather framework Call follows from genuine clients, at that point utilized a "partitional" grouping calculation to order kindhearted and noxious applications. In [3] the creators utilized static examination to productively distinguish high and medium hazard

applications. Past work has additionally utilized application consents to pinpoint malware [4] – [6]. Sarma et al. [4] utilize chance signs removed from application authorizations, e.g., uncommon basic consents (RCP) what's more; uncommon sets of basic authorizations (RPCP), to prepare SVM what's more, illuminate clients of the dangers versus benefits tradeoffs of applications. In this paper we demonstrate that FairPlay fundamentally enhances the execution accomplished by Sarma et al. [4]. Peng et al. [5] propose a score to gauge the hazard of applications, in light of probabilistic generative models, for example, Guileless Bayes. Yerima et al. [6] likewise utilize highlights separated from application authorizations, API calls and orders removed from the application executable. Sahs and Khan [7] utilized highlights removed from application authorizations and control stream charts to prepare a SVM classifier on 2000 considerate and under 100malicious applications. Sanz et al. [8] depend entirely on authorizations as wellsprings of highlights for several machine learning apparatuses. They utilize a dataset of around 300 honest to goodness and 300 malware apps.ssify benevolent and malignant applications. Google has sent Bouncer, a structure that screens distributed applications to distinguish and expel malware. Oberheide also, Miller [9] has dissected and uncovered points of interest of Bouncer (e.g., situated in QEMU, utilizing both static and dynamic examination). Bouncer isn't adequate - our outcomes demonstrate that 948 applications out of 7,756 applications that we downloaded from Google Play are identified as suspicious by at any rate 1 hostile to infection apparatus. Also, FairPlay distinguished suspicious conduct for applications that were not evacuated by Bouncer amid an over a half year long interim. Rather than breaking down application executable, FairPlay utilizes a social, phonetic and conduct approach in light of longitudinal application information. FairPlay's utilization of application authorizations contrasts from existing work through its emphasis on the fleeting measurement, e.g., changes in the quantity of asked for consents, specifically the "unsafe" ones. We watch that FairPlay recognizes and abuses another connection between malware and look rank extortion.

III. METHODOLOGY

3.1 Overview

FairPlay sorts out the investigation of longitudinal application information into the accompanying 4 modules, represented in Figure 1. The CoReview Chart (CoReG) module distinguishes applications audited in an adjoining time window by gatherings of clients with altogether covering survey chronicles. The Review Feedback (RF) module misuses criticism left by honest to goodness commentators, while the Inter Review Relation (IRR) module influences relations between audits, appraisals and introduce tallies. The Jekyll-Hyde (JH) module screens application authorizations, with an attention on risky ones, to distinguish applications that change over from considerate to malware. Every module delivers a few highlights that are utilized to prepare an application classifier. FairPlay moreover utilizes general highlights, for example, the application's normal rating, add up to number of audits, appraisals and introduces, for an aggregate of 28 highlights.

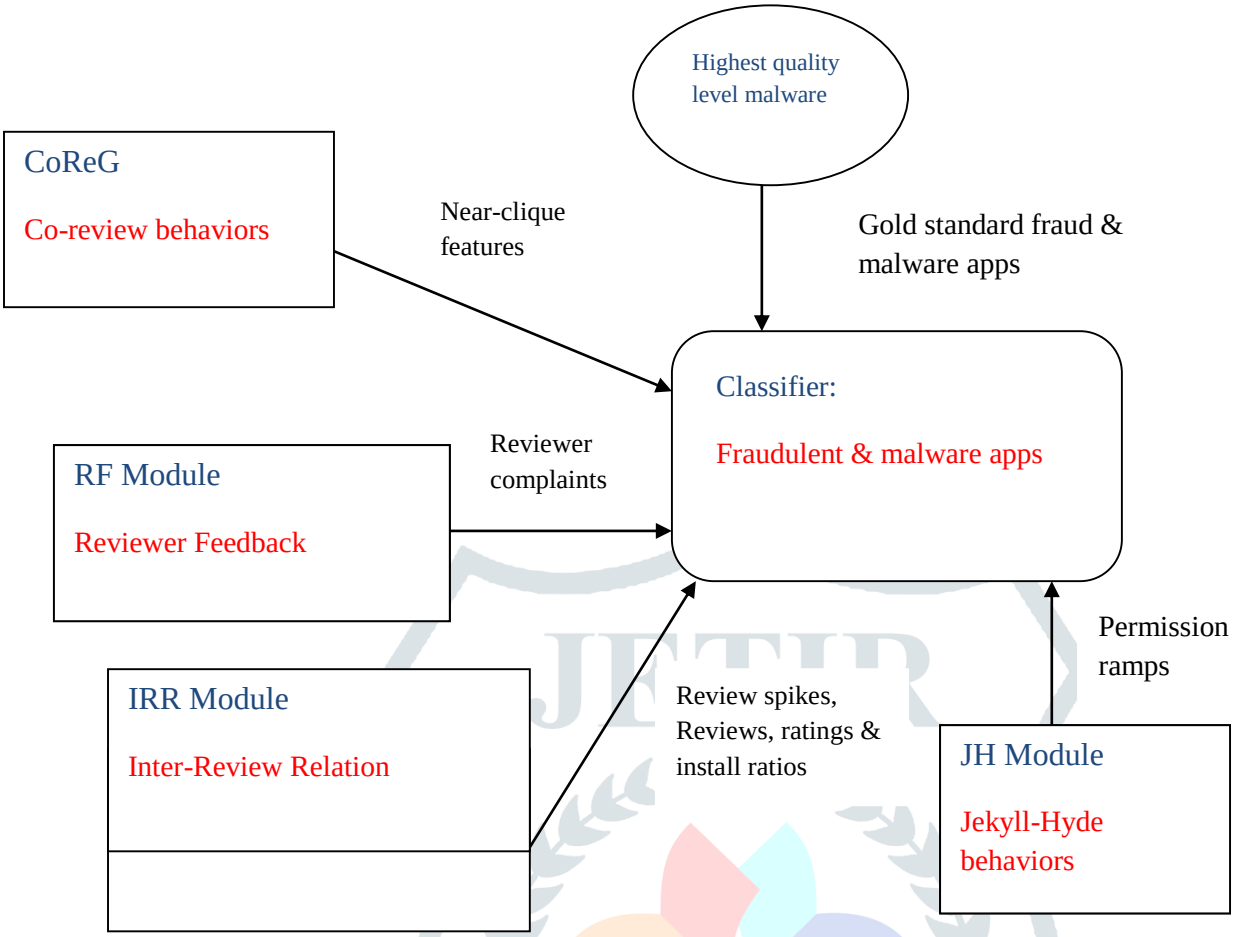

Figure 1: system architecture.

3.2 The Co-Review Graph (CoReG) Module:

This module abuses the perception that fraudsters who control numerous records will re-utilize them over various occupations. Its objective is then to recognize sub-sets of an application's analysts that have performed critical basic survey exercises before. In the accompanying, the co-survey diagram idea, formally show the weighted maximal coterie specification issue, at that point present an effective heuristic that use regular constraints in the practices of fraudsters. Co-Review Graphs, give the co-a chance to audit chart of an application, be where hubs relate to client accounts who evaluated the application, and undirected edges have a weight that shows the quantity of applications looked into in like manner by the edge's endpoint users. The coreview diagram idea normally distinguishes client accounts with critical past survey exercises. Maximal club list calculations, for example, connected to co-audit diagrams are not perfect to take care of the issue of recognizing sub-sets of an application's commentators with noteworthy past regular surveys. To begin with, fraudsters may not reliably utilize (or may even intentionally abstain from utilizing) every one of their records over all extortion occupations that they perform. What's more, Google Play gives deficient data (up to 4,000 surveys per application, may likewise recognize and channel extortion). Since edge data might be fragmented, unique coteries may now likewise be deficient. The Pseudo Clique Finder (PCF) Algorithm: a calculation that adventures the perception that fraudsters contracted to survey an application are probably going to post those audits inside moderately brief time interims (e.g., days). PCF, takes as info the arrangement of the audits of an application, composed by days, and a limit esteem. PCF yields an arrangement of recognized pseudo-fractions that were shaped amid bordering time spans.

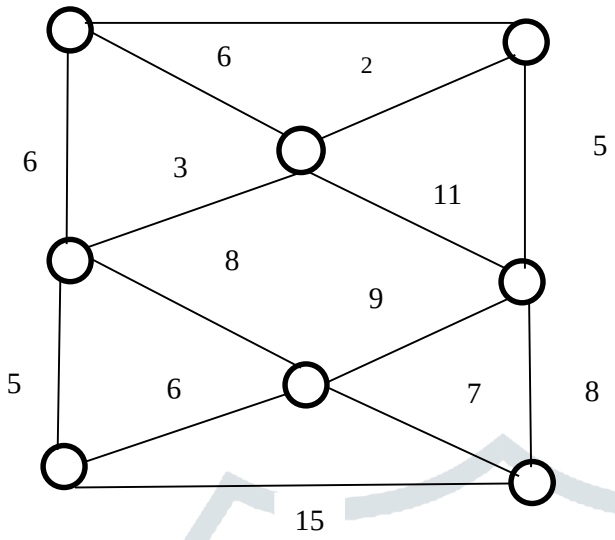

Figure 2: co-review graph

PCF Algorithm:

Information: days, a variety of day by day audits, and u, the weighted limit thickness

Yield: allCliques,set of all recognized pseudo-inner circles

1. for d: =0 d <days.size (); d++
2. Graph PC: = new Graph ();
3. BestNearClique (PC, days[d]);
4. C: = 1; n: =PC.size ();
5. for nd: = d+1; d <days.size () and c = 1; d++
6. BestNearClique (PC, days [nd]);
7. C: = (PC.size () > n); endfor
8. if (PC.size () > 2)
9. allCliques:=allCliques.add(PC); fi endfor
10. return
11. function bestNearClique (Graph PC, Set revs)
12. if (PC.size () = 0)
13. for root: = 0; root < revs.size(); root++
14. Graph candClique: = new Graph ();
15. CandClique.addNode(revs [root].get User ());
16. docandNode:=getMaxDensityGain (revs);

```
17.  if (density (candClique ({candNode}) u))
18.  CandClique.addNode (candNode); fi
19.  while (candNode! = invalid);
20.  if (candClique.density() > maxRho)
21.  maxRho: =candClique.density ();
22.  PC: = candClique; fi endfor
23.  else if (PC.size () > 0)
24.  docandNode:=getMaxDensityGain(revs);
25.  if (density (candClique [candNode) u))
26.  PC.addNode (candNode); fi
27.  while (candNode!= invalid);
28.  return
```

3.3 Reviewer Feedback (RF) Module Reviews:

Written by genuine users of malware and fraudulent apps may describe negative experiences. The RF module exploits this observation through a two-step approach: (i) detect and filter out fraudulent reviews, then (ii) identify malware and fraud indicative feedback from the remaining reviews.

Reviewer based features:

The expertise of User for app A, defined as the number of reviews User wrote for apps that are “similar” to A, as listed by Google Play. The bias of User towards A: the number of reviews written by User for other apps developed by A’s developer. In addition, we extract the total money paid by User on apps it has reviewed, the number of apps that User has liked, and the number of Google+ followers of User.

Text based features:

We used the NLTK library and the Naive Bayes classifier, trained on two datasets: (i) 1,041 sentences extracted from randomly selected 350 positive and 410 negative Google Play reviews, and (ii) 10,663 sentences extracted from 700 positive and 700 negative IMDB movie reviews.(i) since no app is perfect, a “balanced” review that contains both app positive and negative sentiments is more likely to be genuine, and (ii) there should exist a relation between the review’s dominating sentiment and its rating. Thus, after filtering out fraudulent reviews, we extract feedback from the remaining reviews.

3.4 Inter Review Relation (IRR) Module:

This module use worldly relations between reviews, and additionally relations between the review, rating and introduce checks of applications, to distinguish suspicious practices. Transient relations, with a specific end goal to make up for a negative review, an aggressor needs to post a critical number of positive reviews.

Fig. 3 plots the lower bound on the quantity of fake reviews that should be presented on drop a 1-star review, versus the application's present rating. It demonstrates that the quantity of reviews expected to help the rating of an application isn't consistent. Rather, as a review battle supports the rating of the subject application, the quantity of fake reviews expected to proceed with the procedure, additionally increments. For example, a 4 star application needs to get 3, 5-star reviews to adjust for a solitary 1 star review, while a 4.2 star application needs to get 4 such reviews. Along these lines, fraudsters who need to expand the rating of an application, i.e., counterbalance beforehand got negative reviews, should post an expanding, critical number of positive reviews. Such a "compensatory" conduct is probably going to prompt suspiciously high quantities of positive reviews.

We recognize such practices by distinguishing exceptions in the quantity of day by day positive audits gotten by an application. Demonstrates the courses of events and suspicious spikes of positive review for 2 applications from the deceitful application dataset. We distinguish days with spikes of positive reviews as those whose number of positive reviews surpasses the upper external fence of the box and-hair plot worked over the application's quantities of day by day positive reviews.

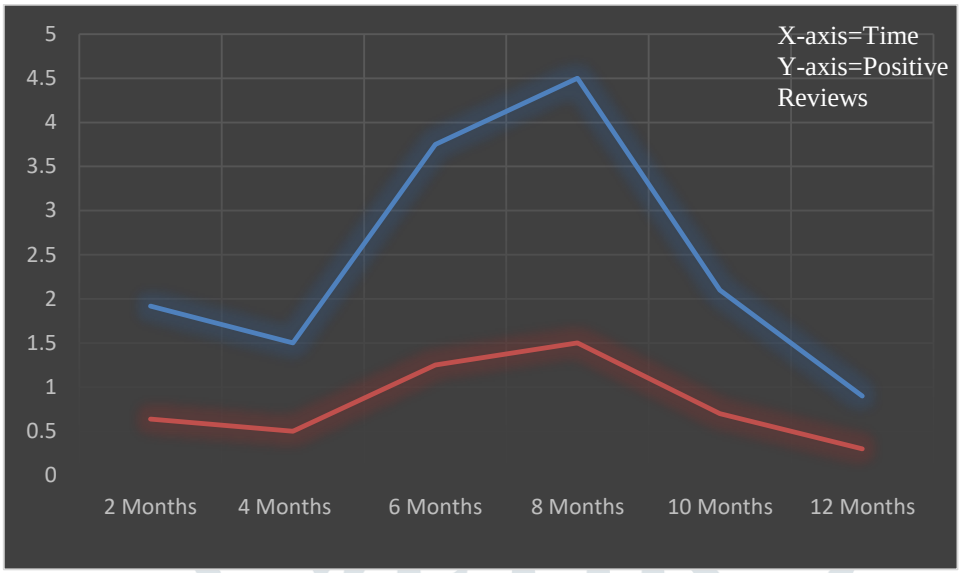

Figure 3: timelines of positive and negative reviews.

3.5 Jekyll-Hyde App Detection (JH) Module:

Fig. 4 demonstrates the dispersion of the aggregate number of consents asked for by malware, false and honest to goodness applications. Shockingly, malware and fake applications as well as genuine applications ask for huge quantities of authorizations.

Also, Android's API level 22 marks 47 authorizations as "risky". Fig. 4 looks at the appropriations of the quantity of hazardous authorizations asked for by the best quality level malware, deceitful and favorable applications. The most prevalent risky authorizations among these applications are "change or erase the substance of the USB stockpiling", "read telephone status and character", "discover accounts on the gadget", and "access exact area". Just 8 percent of the true blue applications ask for in excess of 5 perilous authorizations, while 16.5 percent of the malware applications and 17 percent of the deceitful applications ask for in excess of 5 consents. Maybe shockingly, most real (69 percent), malware (76 percent) and fake applications (61 percent) ask for somewhere in the range of 1 and 5 risky authorizations.

After an ongoing Google Play arrangement change, Google Play sorts out application authorizations into gatherings of related consents. Applications can ask for a gathering of authorizations and increase certain entrance additionally to perilous consents. Upon manual assessment of a few applications, we recognized another kind of malevolent expectation perhaps executed by misleading application designers: applications that look to pull in clients with insignificant authorizations, yet later demand perilous consents. The client might be unwilling to uninstall the application "just" to dismiss a couple of new consents. We call these Jekyll-Hyde applications.

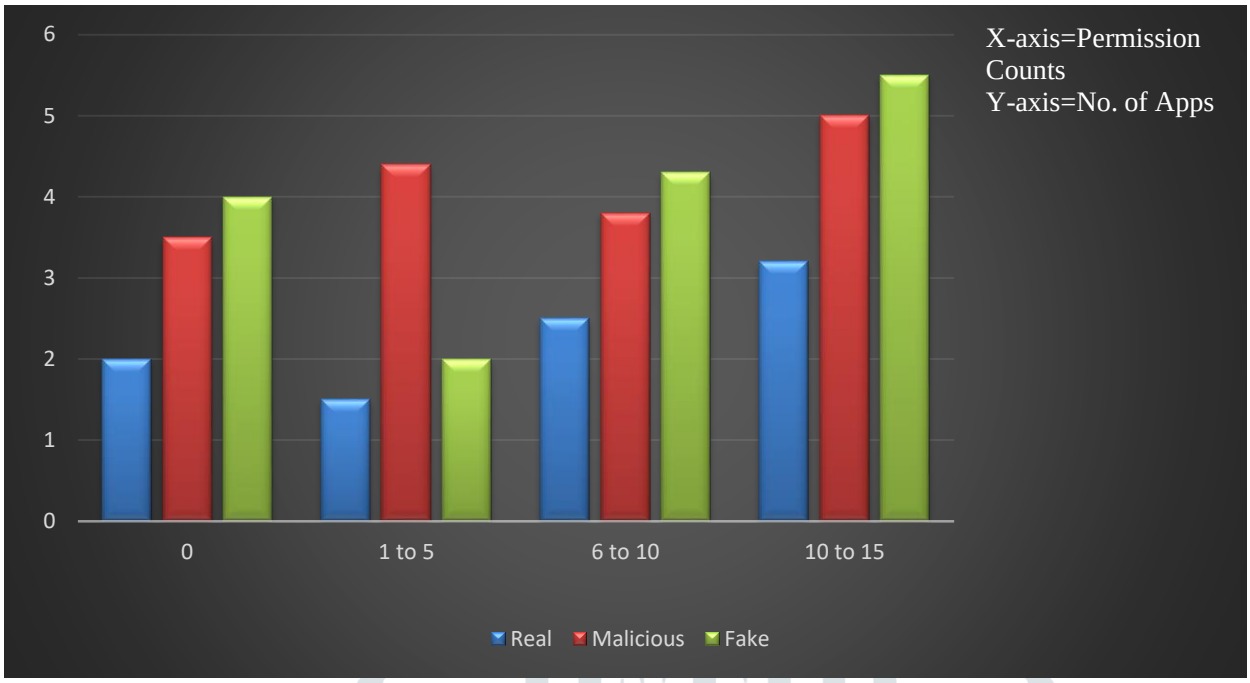

Figure 4: Appropriation of aggregate number of permissions requested by malicious, fake and real applications

IV. RESULTS

4.1 Audit Classification:

To assess the exactness of FairPlay's fake audit location segment (RF module), we utilized the best quality level datasets of deceitful and real surveys. We utilized GPCrawler to gather the information of the authors of these audits, including the 203 commentators of the 406 false surveys (21,972 audits for 2,284 applications) and the 315 analysts of the honest to goodness surveys (9,468 surveys for 7,116 applications). We watch that the clients who post bonafide audits compose less surveys altogether than the individuals who post false surveys; in any case, by and large, those clients survey more applications altogether. We have additionally gathered data about each of these gathered applications, e.g., the identifiers of the application engineer.

4.2 App Classification:

To assess FairPlay, we have gathered all the 97,071 audits of the 613 highest quality level malware, fake and kindhearted applications, composed by 75,949 clients, and additionally the 890,139 applications evaluated by these clients.

In the accompanying, we assess the capacity of different directed learning calculations to accurately arrange applications as either benign, false or malware. In particular, in the primary test we prepare just on deceitful and considerate application information, and test the capacity to precisely arrange an application as either fake or considerate. In the second trial, we prepare and test just on malware and benign applications. In the third trial, we prepare a classifier on fake and favorable applications, at that point test its exactness to group applications as either malware or considerate. At last, we ponder the most impactful highlights while ordering fake versus generous and malware versus benevolent applications.

Is Malware Involved in Fraud? We guessed that the above outcome is expected to some degree to malware applications being associated with search rank misrepresentation. To confirm this, we have prepared FairPlay on the best quality level favorable and deceitful application datasets, at that point we have tried it on the best quality level malware dataset. MLP is the most preservationist calculation, finding 60.85 percent of malware as misrepresentation members. Arbitrary Forest finds 72.15 percent, and Decision Tree banners 75.94 percent of the malware as deceitful. This outcome affirms our guess and demonstrates that inquiry rank extortion location can be a critical expansion to versatile malware discovery endeavors.

Additional astonishing are the highlights that don't show up in the best, for either classifier. Most prominently, the Jekyll-Hyde highlights that measure the inclines in the quantity of hazardous consents. One clarification is that the 212 malware applications in our best quality level dataset don't have adequate hazardous authorization inclines. Likewise, we take note of that our guess that

fraudster endeavors to modify the hunt rank of an application won't have the capacity to safeguard a characteristic adjust of the highlights that effect it (see IRR module) is just mostly approved: exclusively the I1/Rv1 include has an influence in separating malware from amiable applications.

Moreover, we have zoomed in into the appropriations of the sizes and densities of the biggest pseudo-coteries, for the highest quality level false and malware applications.

4.3 Coercive Review Campaigns:

Upon close assessment of applications hailed as deceitful by FairPlay, we identified applications executing another assault compose: badger the client to it is possible that (I) compose a positive review for the application, or (ii) introduce and compose a positive audit for different applications (regularly of a similar designer). We call these practices coercive audit battles and the subsequent reviews, as pressured reviews. Illustration constrained surveys incorporate, "I just evaluated it since I didn't need it to fly up while I am playing", or "Couldn't play one level before I needed to rate it [...] they really are instructing me to rate the application 5 stars".

Keeping in mind the end goal to discover confirmation of efficient coercive audit crusades, we have parsed the 2.9 million surveys of our dataset to distinguish those whose content contains one of the root words ["make", "ask", "power"] and "rate". Upon manual examination of the outcomes, we have discovered 1,024 pressured surveys. The surveys uncover that applications associated with coercive audit crusades either have bugs (e.g., they request that the client rate 5 stars even after the client has appraised them), or reward the client by evacuating promotions, giving more highlights, opening the following diversion level, boosting the client's amusement level or granting diversion focuses.

We have watched a few copies among the constrained audits. We recognize two conceivable clarifications. To begin with, as we already specified, some applications don't monitor the client having explored them, along these lines more than once constrain ensuing surveys from a similar client. A second clarification is that apparently constrained surveys can likewise be posted as a feature of a negative pursuit rank extortion crusade. In any case, the two situations depict applications liable to have been subjected to false practices.

V. CONCLUSIONS

We have presented FairPlay, a framework to distinguish both fake and malware Google Play applications. Our analyses on a recently contributed longitudinal application dataset, have demonstrated that a high level of malware is associated with search rank misrepresentation; both are precisely distinguished by FairPlay. Furthermore, we demonstrated FairPlay's capacity to find many applications that dodge Google Play's location innovation, including another sort of coercive extortion assault.

REFERENCES

1. Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution,” in Proc. IEEE Symp. Secur. Privacy, 2012, pp. 95–109.
2. I.Burguera, U. Zurutuza, and S. Nadjm-Tehrani, “Crowdroid: Behavior-based Malware detection system for Android,” in Proc. ACM SPSM, 2011, pp. 15–26.
3. A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “Andromaly: A behavioral malware detection framework for Android devices,” Intell. Inform.Syst., vol. 38, no. 1, pp.161–190, 2012.
4. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, “RiskRanker: Scalable and accurate zero-day Android malware detection,” in Proc. ACM MobiSys, 2012, pp. 281–294.
5. B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android Permissions: A Perspective Combining Risks and Benefits,” in Proc. 17th ACM Symp. Access Control Models Technol., 2012, pp. 13–22.
6. H. Peng, et al., “Using probabilistic generative models for ranking risks of Android Apps,” in Proc. ACM Conf. Comput. Commun.Secur. 2012, pp. 241–252.
7. S. Yerima, S. Sezer, and I. Muttik, “Android Malware detection using parallel machine learning classifiers,” in Proc. NGMAST, Sep. 2014, pp. 37–42.
8. J. Sahs and L. Khan, “A machine learning approach to Android malware detection,” in Proc. Eur. Intell. Secur. Inf. Conf., 2012, pp. 141–147.

9. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Alvarez, "Puma: Permission usage to detect malware in android," in Proc. Int. Joint Conf. CISIS12-ICEUTE' 12-SOCO' Special Sessions, 2013, pp. 289–298.
10. J. Oberheide and C. Miller, "Dissecting the Android Bouncer," presented at the SummerCon2012, New York, NY, USA, 2012.
11. J. Ye and L. Akoglu, "Discovering opinion spammer groups by network footprints," in Machine Learning and Knowledge Discovery in Databases. Berlin, Germany: Springer, 2015, pp. 267–282.
12. L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion Fraud Detection in Online Reviews by Network Effects," in Proc. 7th Int. AAAI Conf. Weblogs Soc. Media, 2013, pp. 2–11.
13. S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly, 2009.
14. Google Play. [Online]. Available: https://play.google.com/
15. E. Siegel, "Fake reviews in Google Play and Apple App Store," Appentive, Seattle, WA, USA, 2014.
16. Z. Miners. (2014, Feb. 19). "Report: Malware-infected Android apps spike in the Google Play store," PC World. Available: http:// www.pcworld.com/article/2099421/report-malwareinfectedandroid-apps-spike-in-the-google-play-store.html
17. S. Mlot. (2014, Apr. 8). "Top Android App a Scam, Pulled From Google Play," PCMag. Available: http://www.pcmag.com/article2/0,2817,2456165,00.asp
18. D. Roberts. (2015, Jul. 8). "How to spot fake apps on the Google Play store," Fortune. Available: http://fortune.com/2015/07/08/ google-play-fake-app/
19. A. Greenberg (2012, May 23). "Researchers say they snuck malware app past Google's 'Bouncer' Android market scanner," Forbes Security, [Online]. Available: http://www.forbes.com/ sites/andygreenberg/2012/05/23/researchers-say-they-snuckmalware-app-past-googles-bouncer-android-market-scanner/#52c8818d1041
20. Freelancer. [Online]. Available: http://www.freelancer.com