International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 6 | ISSN : 2456-3307

Discovering Fraudulent Behaviors in Google App Stories

G. Sasikala*[1], N. Jayanthi[2]

[1]Assistant Professor, Department of Computer Science and Applications, Adhiparasakthi College of Arts and Science (Autonomous), G.B.Nagar, Kalavai, Vellore, Tamil Nadu, India

[2]M.Phil (CS) Research Scholar, Department of Computer Science and Applications, Adhiparasakthi College of Arts and Science (Autonomous), G.B.Nagar, Kalavai, Vellore, Tamil Nadu, India

ABSTRACT

Google play store first releases its applications in 2008. Since that, it distributes applications to all the Android users. In Google Play Store, an extensive number of those applications are created by a small number of developers; it provides benefits that user can find the specific application, purchase those applications and install it on their mobile devices. Since Android is open source environment, all the details about the application users can be easily accessed by the application developers through Google play. In Google play 1.8 Million mobile applications are accessible and over 25 billion users download that across the world. This prompts to greater chance of installing malware to the applications that could affect user's mobile devices. FairPlay is formed as a system to find and use traces left behind by fraudulent developers to identify both malware and apps subjected to search rank fraud based on review, rating and ranking. FairPlay also links review activities and uniquely combines detected review, rating and ranking relations with linguistic and behavioural signals collected from Google Play app data.

Keywords: Ranking, Review, Rating, Android market, search rank fraud, malware detection

I. INTRODUCTION

To make fraud search in Apps is by searching the high ranked applications up to 30-40, which may be ranked high in some days, or the applications that are in those high ranked lists should be verified but this is not applied when we work for thousands of applications added per day. So, we go for broad view by applying some technique to every application to judge its ranking. In this paper of our project disclosure of ranking fraud for mobile applications, we develop a need to make a flawless and fraud less result that shows corrected application according to provided ranking; where we can get the result by searching fraud of applications. They make ranking of fraud App as high by using techniques such as human water armies and botfarms; where they make fraud by downloading applications through different devices and providing fake ratings and reviews. Therefore, as we said above here we have to mine crucial data relating particular application such as review, which we said, comments and so many other information we have to mine and place algorithm to detect fakeness in application rank.

Some fraudulent developers deceptively boost the search rank and popularity of their apps (e.g., through fake reviews and bogus installation counts) , while malicious developers use app markets as a launch pad for their malware. The motivation for such behaviors is impact: app popularity surges translate into financial benefits and expedited malware proliferation. Fraudulent developers frequently exploit crowd sourcing sites (e.g., Freelancer, Fiverr, BestApp Promotion) to hire teams of willing workers to commit fraud collectively,

emulating realistic, spontaneous activities from unrelated people (i.e., "crowd turfing" [10]), see Fig. 1 for an example. We call this behavior "search rank fraud". In addition, the efforts of Android markets to identify and remove malware are not always successful.

In this paper, we also look to identify both malware and search rank fraud subjects in Google Play. This combination is not arbitrary: we place that malicious developers turn to search rank fraud to increase the effect of their malware. Unlike existing solutions, we build this work on the observation that fraudulent and malicious behaviors leave behind telltale signs on app markets. Resource constraints can compel fraudsters to post reviews within short time intervals. Legitimate users affected by malware may report unpleasant experiences in their reviews. Malware (JekyllHyde) transition can be identified by sudden increases in the number of requested permissions from one version to the next version, which we will call "permission ramps".

Fraudulent developers use search-ranking algorithm to promote their apps to the top while searching. After downloading mobile applications from Google play users are asked to give the ratings and reviews about that particular downloaded applications. However, fraudulent developers give fake ratings and reviews about their application promote their application to the top. There are two typical approaches used for detecting malware in Google Play. Thus are Static and Dynamic. The dynamic approach needs apps to be run in a secure environment to detect its benign. The static approach is not used as the need to give all types of attack in early stage itself but that is impossible as everyday attackers find the new way to inject malware on applications.

The rest of the paper is organized as follows: Section II, presents the literature survey over the related work. In section III, proposed system is presented. In section IV, implementation for each module is presented. Finally, the section V concludes paper.

II. LITERATURE SURVEY

2.1 Android Malware Detection Using Parallel Machine Learning Classifiers (2014)

This paper proposes and investigates a parallel machine learning based classification approach for early detection of Android malware. A composite classification model is produced from parallel combination of heterogeneous classifiers by using real malware samples and generous applications. The empirical evaluation of the model under different combination schemes demonstrates its efficacy and potential to improve detection accuracy. More specifically, their strengths can be harnessed not only for enhanced Android malware detection by utilizing several classifiers with diverse characteristics but also for performing quicker white box analysis using the more interpret-able constituent classifiers.

Algorithm Inherently diverse machine learning algorithms

Advantage Provides a complementary tool .

Disadvantage Hard to handle app

Conclusion A parallel classification approach to Android malware detection using inherently diverse machine learning algorithms was investigated.

2.2 Opinion Fraud Detection in Online Reviews by Network Effects (2013)

This paper proposes a fast and effective framework, FRAUD EAGLE, for spotting fraudulent developers and fake reviews in online review datasets. This technique has multiple advantages as follows: (1) it exploits the network effect among reviewers and products, unlike majority of existing techniques that focus on review text or behavioral analysis, (2) it includes two complementary steps; scoring users and reviews for fraud detection, and grouping for

visualization and sense making, (3) it operates in a completely unsupervised fashion without need of labeled data, while still incorporating side information if available, and (4) it is scalable to large datasets as its run time grows linearly with network size.

Algorithm Fraud Eagle

Advantage It consists of two complementary steps; Scoring users and reviews for fraud detection, and grouping for visualization and sense making. **Disadvantage** Little lazy start because of heavy dataset

Conclusion They propose a novel framework called Fraud Eagle that exploits the network effects to automatically detect fraudulent users and fake reviews in online review network.

2.3 PUMA: Permission Usage to detect Malware in Android (2013)

In this Paper, they present PUMA, a new technique for detecting malicious Android applications by analyzing the extracted permissions from the application itself through machine-learning techniques.

Algorithm Naive Bayes. **Advantage** The high detection rate. **Disadvantage** Time consuming.

Conclusion Improve the detection ratio that does not require executing the sample.

2.4 A Machine Learning Approach to Android Malware Detection (2012)

In paper, they present a machine learning based system for the detection of malware on Android devices. It provides a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner to leverage the higher computing power of a server or cluster of servers.

Algorithm Weisfeiler-Lehman relabeling. **Advantage** these system extracts a maximum number of features. **Disadvantage** Require higher computing power

Conclusion This system extracts a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner.

2.5 RiskRanker: Scalable and Accurate Zero-day Android Malware Detection Twitter (2012)

In paper, they propose a proactive scheme to spot zero day Android malware. This technique is motivated to assess potential security risks posed by these un-trusted apps without relying on malware samples and their signatures. Specifically, they have developed an automated system called Risk Ranker to scalable analyzes whether a particular app exhibits dangerous behavior (e.g. launching a root exploit or sending background SMS messages.

Algorithm Android app analysis. **Advantage** Demonstrate effectiveness and accuracy.

Disadvantage Time consuming. **Conclusion** They present a proactive scheme to scalable and accurately sift through a large number of apps in existing Android markets to spot zero-day malware.

III. SYSTEM MODEL

A. System model:

We focus on the Android app market ecosystem of Google Play. The participants, consisting of users and developers, have Google accounts. Developers create and upload apps that consist of executables (i.e., apks"), a set of required permissions, and a description. The app market publishes this information, along with the app's received reviews (1-5 stars rating & text), ratings (1-5 stars, no text), aggregate rating (over both reviews and ratings), install count range (predefined buckets, e.g., 50-100, 100-500), size, version number, price, time of last update, and a list of "similar" apps.

B. Adversarial model:

The system considers both malicious developers, who upload malware, and rational fraudulent developers. Fraudulent developers attempt to tamper with the search rank of their fraudulent apps. While Google keeps secret the criteria used to rank apps, the reviews, ratings and install counts are known to play

a fundamental part (see e.g., Fraudulent developers often rely on crowd sourcing sites to hire teams of workers to commit fraud collectively. To review or rate an app, a user needs to have a Google account, register a mobile device with that account, and install the app on the device. This process complicates the job of fraudsters, who are thus more likely to reuse accounts across review writing jobs.

IV.EXISTING SYSTEM

1. Within the literature, whereas there are a unit some
   connected work, like net ranking spam detection, on-line review spam detection and mobile App recommendation, the matter of detective work ranking fraud for mobile Apps remains under-explored.
2. Typically speaking, the connected works of this paper are often sorted into 3 classes.
3. The primary class is regarding net ranking spam detection.
   4. The second class is targeted on detective work on-line review spam.
4. Finally, the third class includes the studies on mobile App recommendation.

V. PROPOSED SYSTEM

FairPlay organizes the analysis of longitudinal app data into the following four modules, illustrated in Figure 1. The Co-Review, Rating& Recommendation Graph (CoReRRG) module identifies apps reviewed and Rated in a contiguous time window by groups of users with significantly overlapping review and Rating histories. The **Review & Rating & Recommendation Feedback (RRRF)** module exploits feedback left by genuine reviewers and users, while the **Inter Review Rating Recommendation Relation (IRRRR)** module leverages relations between reviews, ratings and install Rating Recommendation counts. **The Jekyll-Hyde (JH) module** is utilized to monitor app permissions for identifying apps that converted into malware from benign. Each module produces several features that are used to train an app classifier. FairPlay also uses general features such as the app's average rating, total number of reviews, and total number of Recommendations.

A. The Co-Review, Rating& Recommendation Graph (CoReRRG) Module:

The CoReRRG module utilizes the observation that fraudsters who control many accounts will re-use them across multiple purposes, and detects sub-sets of an app's reviewers that have performed significant common review, Rating and recommendations activities in the past. In the following, we describe the co-review rating and recommendation graph concept, formally present the weighted maximal clique enumeration problem, then introduce an efficient heuristic that leverages natural limitations in the behaviors of fraudsters. CoReRR graphs. Let the CoReRR graph of an app, where nodes correspond to user accounts who reviewed the app, and undirected edges have a weight that indicates the number of apps reviewed rated and recommended in common by the edge's endpoint users.

Pseudo Clique Finder (PCF):

The problem of finding dense structures in a given graph is quite basic in informatics including data mining and data engineering. Clique is a popular model to represent dense structures, and widely used because of its simplicity and ease in handling. Pseudo cliques are formed as natural extension of cliques which are sub graphs formed by removing small number of edges from cliques. We here define a pseudo clique by a sub graph such that the ratio of the number of its edges compared to that of the clique with the same number of vertices is no less than a given threshold value. In this paper, we address the problem of enumerating all pseudo cliques for a given graph and a threshold value. We first show that it seems to be difficult to obtain

polynomial time algorithms using straightforward divide and conquer approaches. Then, we propose a polynomial time, polynomial delay in precise, algorithm based on reverse search. The time complexity for each pseudo clique is $O(\Delta \log |V|+\min \{\Delta^2,|V|+|E|\})$. Computational experiments show the efficiency of our algorithm for both randomly generated graphs and practical graphs.

Algorithm 1. PCF Algorithm

Input: days, an array of daily reviews, an array of daily ratings, an array of daily recommendations, and θ , the weighted threshold density

Output: allCliques, set of all detected pseudo-cliques

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() & c = 1; d++
6. bestNearClique (PC, days[nd]);
7. c := (PC.size() > n); **end for**
8. **if** (PC.size() > 2)
9. AllCliques: = allCliques.add (PC); fi **end for**
10. return
11. **function** bestNearClique (Graph PC, Set revs, Set rats, Set recom)
12. **if** (PC.size () = 0)
13. **for** root: = 0; root < revs.size (); root++
14. Graph RatcandClique: = new Graph (); Graph RevcandClique: = new Graph ();Graph ReccandClique: = new Graph ();
15. RevcandClique.addNode (revs [root].getUser ()); RatandClique.addNode (rats [root].getUser ()); ReccandClique.addNode (recom [root].getUser ());
16. **do** RevcandClique:= getMaxDensityGain(revs);**do** RatandClique:= getMaxDensityGain(rats); d**o** ReccandClique:= getMaxDensityGain(recom);
17. if (density(RevcandClique {candNode})>= θ)) for all
18. RevcandClique.addNode(candNode); fi
19. while (candNode != null);
20. if (RevcandClique.density() > maxRho)
21. maxRho := RevcandClique.density();
22. PC: = RevcandClique; fi **end for**
23. **else if** (PC.size() > 0)
24. **do** candNode:= getMaxDensityGain(revs);
25. **if** (density(RevcandClique candNode) θ))
26. PC.addNode(candNode); fi
27. **while** (candNode != null);
28. return

B. Review & Rating & Recommendation Feedback (RRRF) Module:

The RRRF module exploits this observation through a two step approach:

(i) detect and filter out fraudulent reviews, rating and recommendations then (ii) identify malware and fraud indicative feedback from the remaining reviews, rating and recommendations.

Step 1: Fraudulent Filter:

- Reviewer based feature: The expertise of U for app A, defined as the number of reviews U wrote for apps that are "similar" to A, as listed by Google Play
- Text based features. To implement this feature we used the NLTK library and the Naive Bayes classifier, where these are trained on two sentences extracted from 700 positive and 700 negative IMDB movie reviews.

Step 2: feedback extraction:

We conjecture that (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.

C. Inter Review Rating Recommendation Relation (IRRRR) Module:

This module leverages temporal relations between reviews, as well as relations between the review, rating, recommendations and installs counts of apps, to identify suspicious behaviors.

• **Temporal Relations:** In order to compensate for a negative review, an attacker needs to post a significant number of positive reviews.

• **Reviews, Ratings, Recommendations and Install Counts:** We used the Pearson's x2 test to investigate relationships between the install count and the rating count, as well as between the install count and the average app rating of the 87 K new apps, at the end of the collection interval. We grouped the rating count in buckets of the same size as Google Play's install count buckets

D. Jekyll-Hyde App Detection (JH) Module:

The module ensures distribution of the total number of permissions requested by malware, fraudulent and legitimate apps, where the module also detects legitimate apps requesting large numbers of permissions.

VI.RESULT

A. SCREENSHOT

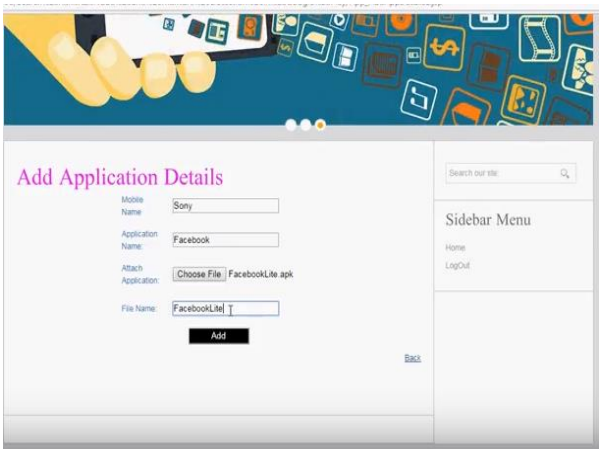

Figure 1. Adding application details

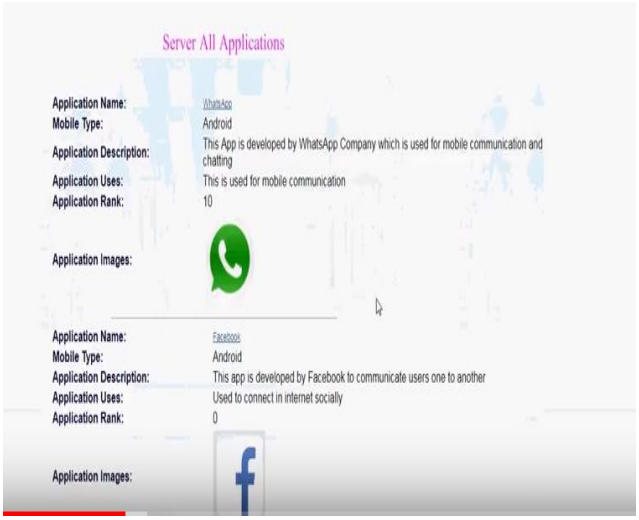

Figure 2. List of applications

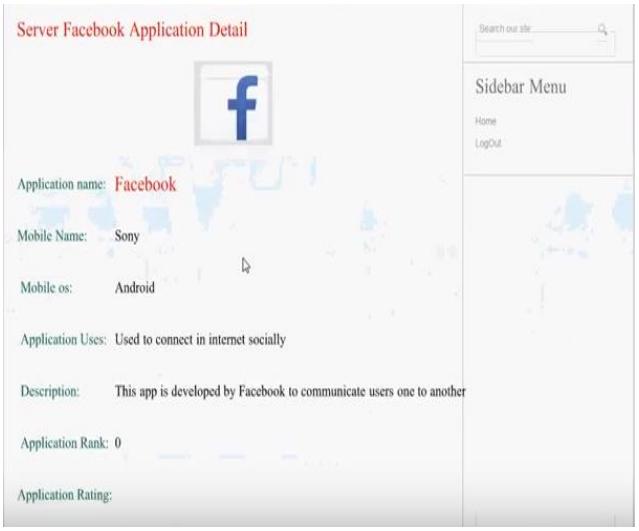

Figure 3. Selected application details

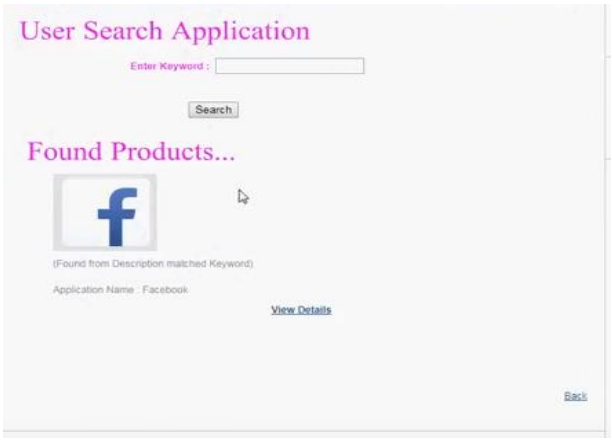

Figure 4. Searching application based on keyword

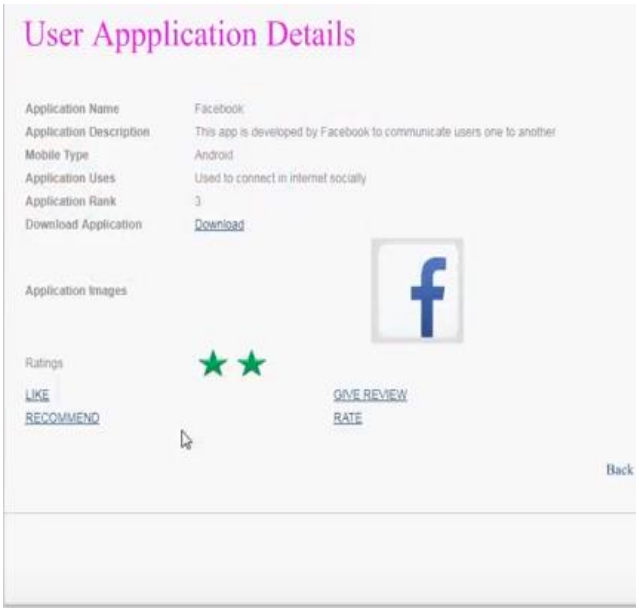

Figure 5: Searched application details

Figure 6. Downloading searched application

Figure 7. Reviewing downloaded application

Figure 8. Review list of selected application

Figure 9. User recommendation list for selected application

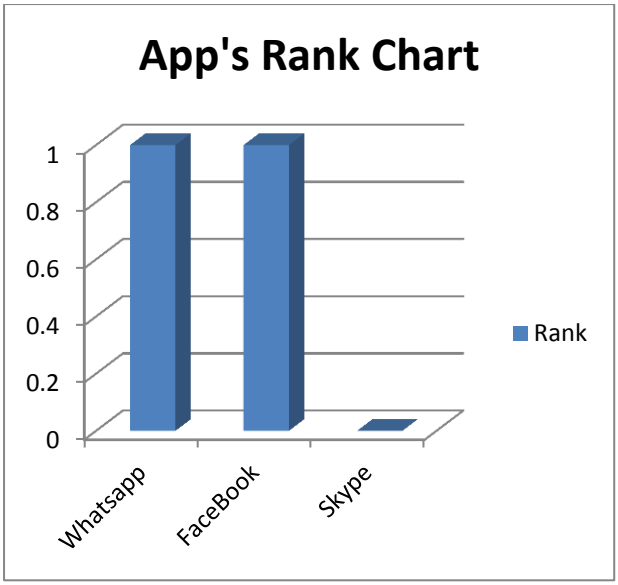

Figure 10. App's Rank Chart

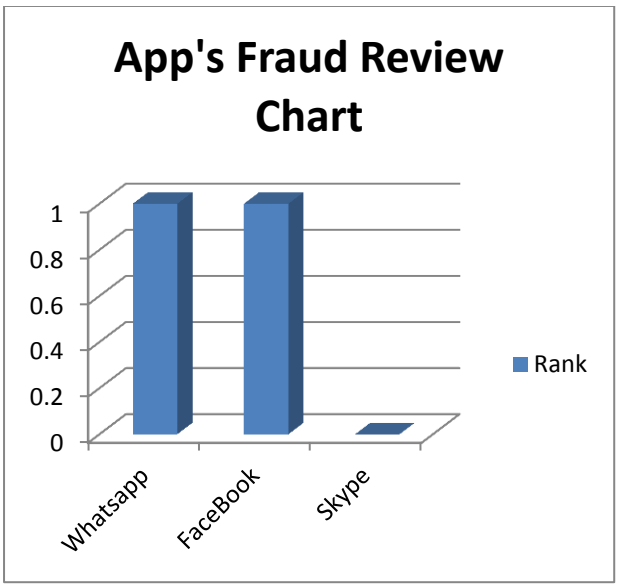

Figure 11. App's Review Fraud Chart

VII.CONCLUSION AND FUTURE ENHANCEMENT

A. CONCLUSION

We proposed FairPlay as a system to identify both fraudulent and malware apps in Google Play. FairPlay accurately identifies that high percentage of malware is involved in search rank fraud through our experiments on an inventive contributed longitudinal app dataset. FairPlay can also identify hundreds of apps that evade Google Play's detection technology as well as a new type of coercive fraud attack.

B. FUTUE ENHANCEMENT

In the future, we plan to study more effective fraud evidence and analyse the latent relationship among rating, review, and rankings. In addition, we can extend our ranking fraud detection methods with other mobile App related services, such as mobile Apps recommendation, for enhancing user experience.

IV. REFERENCES

[1]. Google Play. Online]. Available: https://play.google.com/

[2]. E. Siegel, "Fake reviews in Google Play and Apple App Store," Appentive, Seattle, WA, USA, 2014.

[3]. 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

[4]. 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

[5]. 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/

[6]. 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/ andy Green berg / 2012/05/23/researchers-say-they snuckmalware-app-past-googles-bouncer-android-market-scanner/ #52c8818d1041

[7]. Freelancer. Online]. Available: http://www.free lancer. com

[8]. Fiverr. Online]. Available: https://www.fiverr. com/

[9]. BestAppPromotion. Online]. Available: www.bestreviewapp. com/

[10]. G. Wang, et al., "Serf and turf: Crowdturfing for fun and profit," in Proc. ACM WWW, 2012. Online]. Available: http://doi.acm. org/10.1145/ 2187836.2187928

[11]. J. Oberheide and C. Miller, "Dissecting the Android Bouncer," presented at the SummerCon2012, New York, NY, USA, 2012.

[12]. VirusTotal - free online virus, Malware and URL scanner. Online]. Available https://www. virustotal.com/, Last accessed on: May 2015.

[13]. I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani , "Crowdroid: Behavior-based Malware detection system for Android," in Proc. ACM SPSM, 2011, pp. 15–26.

[14]. 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.

[15]. 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.

[16]. 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.

[17]. 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.

[18]. S. Yerima, S. Sezer, and I. Muttik, "Android Malware detection using parallel machine learning classifiers," in Proc. NGMAST, Sep. 2014, pp. 37–42.

[19]. Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in Proc. IEEE Symp. Secur. Privacy, 2012, pp. 95–109.

[20]. Fraud detection in social networks, Online]. Available: https:// users.cs.fiu.edu/ carbunar /caspr.lab /socialfraud.html

[21]. Google I/O 2013 - getting discovered on Google Play, 2013. Online]. Available: www.youtube.com/ watch?v=5Od2SuL2igA