Search Rank Fraud and Malware Detection in Google Play Application

Nisha.R[1], Dr.Subha.R[2], Balamurugan.V[3]
[1,2,3]*Department of CSE, Sri Eshwar College of Engineering, Coimbatore, India*

***Abstract*- The survey of Fair Play and a novel system discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud. Fair Play correlates review activities and uniquely combines detected review relations with linguistic and behavioral signals gleaned from Google Play app data in order to identify suspicious apps. Adversaries can have chances to launch attacks by gathering victim's information continuously. This survey describe that an adversary can successfully infer a victim's vertex identity and community identity by the knowledge of degrees within a time period. The survey also recommend to a new supervised clustering algorithm to find groups of data cluster. It directly incorporates the information of sample categories into the fraud clustering process.**

Index Terms- Graph Mining, Co-Review Mining, Clustering, Fair Play, Security, Clique detection

I. INTRODUCTION

The commercial success of Android app markets such as Google Play and the incentive model they offer to popular apps, make them appealing targets for fraudulent and malicious behaviors. Some fraudulent developers deceptively boost the search rank and popularity of their apps, 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.

Fraud and Malware Detection Approach is to detect fraud and malware, we propose and generate 28 relational, behavioral and linguistic features that we use to train supervised learning algorithms.

Formulate the notion of co-review graphs to model reviewing relations between users. Develop PCF, an efficient algorithm to identify temporally constrained, co-review pseudo-cliques—formed by reviewers with substantially overlapping co-review activities across short time windows. The main objectives of the FairPlay are

1. To automatically detect malicious and fraudulent apps.
2. To correlate review activities and uniquely combines detected review relations with linguistic and behavioral signals.
3. To discover and leverage traces left behind by fraudsters.
4. To detect both malware and apps subjected to search rank fraud.

The achieve the main goal, the specific objectives required are

- To create a The Co-Review Graph (CoReG) that identifies apps reviewed in a contiguous time window by groups of users with significantly overlapping review histories.
- To propose review feedbacks approach which exploits feedback left by genuine reviewers?
- To prepare clique from the Co-Review graph so that most related fraudulent users are found out.

II.RELATED WORKS

iker burguera, urko zurutuza [1] proposed a new framework to obtain and analyze smart phone application activity. In collaboration with the Android user's community, it will be capable of distinguishing between benign and malicious applications of the same name and version, detecting anomalous behavior of known applications. Furthermore, by deploying our plat- form on a number of test smart phones, we have created a proof of concept for this mechanism, as a means of analyzing emerging threats. We have indicated that monitoring system calls is a feasible way for detecting malware. This analysis technique has been widely used in the literature. According to the brief survey, we have seen that there're many different

approaches to detect malware. We considered that monitoring system calls is one of the most accurate techniques to determine the behavior of Android applications, since they provide detailed low level information. We do realize that API call analysis, information flow tracking or network monitoring techniques can contribute to a deeper analysis of the malware, providing more useful information about malware behavior and more accurate results. On the other hand, more monitoring capability will place a higher demand on the amount of resources consumed in the device.

Asaf shabtai, uri kanonov [2] presents Andromaly—a framework for detecting malware on Android mobile devices. The proposed framework realizes a Host-based Malware Detection System that continuously monitors various features and events obtained from the mobile device and then applies Machine Learning anomaly detectors to classify the collected data as normal (benign) or abnormal (malicious). Since no malicious applications are yet available for Android, we developed four malicious applications, and evaluated Andromaly's ability to detect new malware based on samples of known malware. We evaluated several combinations of anomaly detection algorithms, feature selection method and the number of top features in order to find the combination that yields the best performance in detecting new malware on Android. Empirical results suggest that the proposed framework is effective in detecting malware on mobile devices in general and on Android in particular.

In this paper we presented a malware detection framework for Android which employs Machine Learning and tested various feature selection methods and classification/anomaly detection algorithms. The detection approach and algorithms are light-weight and run on the device itself. There is however also an option to perform the detection at a centralized location, or at least report the analysis results, derived locally on each device, to such a centralized location. This can be useful in detection of malware propagation patterns across a community of mobile devices. As stated by Rich Canning's, 5 Google's Android Security Leader, the Android Market place was chosen to be the place for reporting security issues by users. Users can mark applications as harmful, thereby triggering a security team to launch an investigation. Andromaly can be used for reporting suspicious behavior of applications to the Android Market.

michael grace, yajin zhou [3] presents a proactive scheme to scalably and accurately sift through a large number of apps in existing Android markets to spot zero-day malware. Specifically, our scheme assesses the potential security risks from un- trusted apps by analyzing whether dangerous behaviors are exhibited by these apps (with two-order risk analysis). We have implemented a prototype of Risk Ranker and evaluate it using 118 , 318 apps from a variety of Android markets to demonstrate its effectiveness and accuracy: among the apps in the sample, our system successfully discovered 718 malware samples in 29 families, including 322 zero-day specimens from 11 distinct families.

hao peng, chris gates [4] introduce the notion of risk scoring and risk ranking for Android apps, to improve risk communication for Android apps, and identify three desiderata for an effective risk scoring scheme. We propose to use probabilistic generative models for risk scoring schemes, and identify several such models, ranging from the simple Naive Bayes, to advanced hierarchical mixture models. Experimental results conducted using real-world datasets show that probabilistic general models significantly outperform existing approaches, and that Naive Bayes models give a promising risk scoring approach.

We have discussed the importance of effectively communicating the risk of an application to users, and propose several methods to rate this risk. We test these methods on large real-world datasets to understand each method's ability to assign risk to applications. One particular valuable method is the PNB model which has several ad- vantages. It is monotonic, and can provide feedback as to why risk is high for a specific app and how a developer could reduce that risk. It performs well in identifying most current malware apps as high risk, close to the sophisticated HMNB model. And it can differentiate between critical permissions and less-critical ones, making it more difficult to evade when compared with the BNB model

suleiman y. Yerima, sakir sezer [5] investigate parallel classification approach to Android malware detection using inherently diverse machine learning algorithms. The proposed approach utilized a wide range of features which included API calls related,

commands related and permission features. The recent increase in Android malware and their growing ability for adept detection avoidance of existing signature - based approaches definitely calls for novel alternatives. The parallel classification approach proposed in this paper is a viable scheme that provides a complementary tool that not only potentially improves Android malware detection but al so allows the strengths of diverse classifiers to be leveraged. For example, the rule based classifiers can provide human - interpretable intermediate output that can be useful for driving further analysis stages. Furthermore, the proposed approach is ideal from performance point of view since it is cost effective in classifying a new application because: 1) static app features are employed and 2) the selected constituent classification models have low computational requirements during classification decision .

justin sahs, latifur khan [6] presented a novel machine learning-based malware detection system for the Android operating system. Our system has shown promising results in that it has a very low false negative rate, but also much room for improvement in its high false positive rate. There are a number of possible improvements that could be investigated. A. Features: Our system is limited to just the permissions (built-in and non-standard), and CFGs of the input applications. There are many other potential sources of information-rich features. For instance, there are other metadata entries in the manifest file that contains the requested permissions. There are also many potential sources of features in the program code itself, such as constant declarations and method names. Additionally, the current features could be improved. In particular, the way we extract CFGs abandons much of the information originally present in the code: we label nodes in the CFG based only on the last instruction of the block it rep-resents. We could instead label based on all of the instructions in the block. We also only have a small set of labels, which could be expanded to include more detailed information about the kinds of instructions present (e.g. arithmetic operations, memory access, etc.). Such distinctions would lead to a much more robust label set, which would possibly increase the power of the graph kernel, since it is based on the graph labels.

borja sanz, igor santos[7] describes, permissions are the most recognizable security feature in Android. User must accept them in order to install the application. In this paper we evaluate the capacity of permissions to detect malware using machine-learning techniques. In order to validate our method, we collected 239 malware samples of Android applications. Then, we extracted the aforementioned features for each application and trained the models, evaluating each configuration using the Area under ROC Curve (AUC). We obtained a 0.92 of AUC using the Random Forest classifier. Nevertheless, there are several considerations regarding the viability of our approach. Forensic experts are developing reverse engineering tools over Android applications, from which researchers could retrieve new features to enhance the data used to train the models. Furthermore, despite the high detection rate, the obtained result has an high false positive rate. Consequently, this method can be used as a first step before other more extensive analysis, such as a dynamic analysis.

Future work of this Android malware detection tool is oriented in two main directions. First, there are other features from the applications that could be used to improve the detection ratio that do not require executing the sample. Forensics tools for Android applications should be developed in order to obtain new features. Second, dynamic analysis provides additional information that could improve malware detection systems. Unfortunately, smart phones resources are limited and this kind of analysis usually consumes resources that these devices don't have.

junting ye, leman akoglu [8] proposed an unsupervised and scalable approach for spotting spammer groups in online review sites solely based on their network footprints. Our method consists of two main components: (1) NFS; a new measure that quantifies the statistical distortions of well-studied properties in the review network, and (2) Group-Strainer; a hierarchical clustering method that chips off colluding groups from a sub network induced on target products with high NFS values. We validated the effectiveness of our method on both synthetic and real-world datasets, where we detected various groups of users with suspicious colluding behavior.

III. METHODOLOGY

Commercial success of Android app markets like Google Play] and the incentive model they offer to popularize apps, make them appealing targets for malicious and fraudulent behaviors. Some fraudulent developers deceptively boost 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. The main problem is to detect malicious and fraudulent apps. Hence if a system that leverages the above observations to efficiently detect Google Play fraud and malware, then it will be helpful. So the project introduces FairPlay, a system to automatically detect malicious and fraudulent apps.

A) Overview

FairPlay organizes the feedbacks given by users and preprocesses the reviews. Then the CoReview Graph is being constructed. This CR Graph exploits the observation that fraudsters who control many accounts will re-use them across multiple jobs. Its goal is then to detect sub-sets of an app's reviewers that have performed significant common review activities in the past. In the following, we describe the co-review graph concept, formally present the weighted maximal clique enumeration problem, then introduce an efficient heuristic that leverages natural limitations in the behaviors of fraudsters.

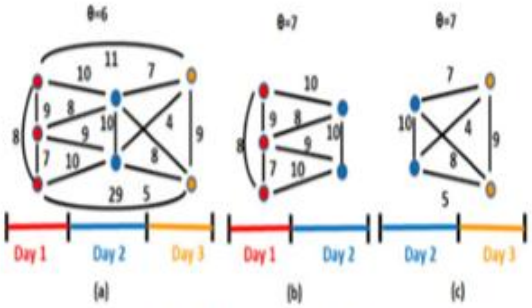

Fig 3.1 Cliques and PCF output

Nodes are users and edge weights denote the number of apps reviewed in common by the end users. Review timestamps have a 1-day granularity. (a) The entire co-review graph, detected as pseudo-clique by PCF when u is 6. When u is 7, PCF detects the sub graphs of (b) the first two days and (c) the last two days. When u=8, PCF detects only the clique formed by the first day reviews (the red nodes).

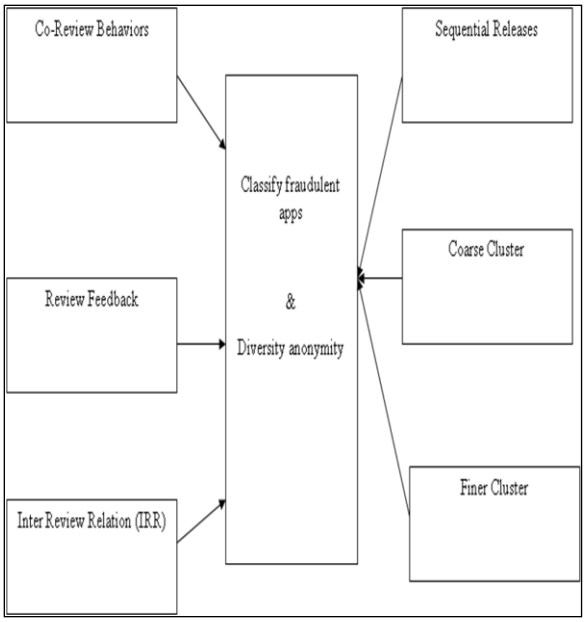

Fig 3.2 Architecture Diagram

Here coarse cluster is the generated main graph. Fine cluster is the graph with least connected nodes removed. If a node with all the edge weights below a given threshold, then the edges and that node are removed.

The following modules are present in the project.

- Tweets Collection for reviews.
- Co-Review Graph Construction.
- Finding Cliques to get fraud users.
- Remove nodes with edge weights below threshold so normal users are treated as non-fraud users.

Tweets Collection For Reviews

In this module,

- Using twitter package and search twitter function, the tweets are downloaded and preprocessed.
- Stop word removal, punctuation removal, unicode character removal are carried out.
- Key Terms are filtered such that first 50 more occurrence words are taken.
- Then unique users in the tweet are also found out.

Co-Review Graph Construction

In this phase,

- From unique users in the tweet are found out.

- Same Key word present in two topics of two different users are found, then two nodes and one edge is formed in the graph.
- Thus the full graph is constructed. During edge addition, co-occurrence count is also found out and set as edge weight.

Finding Cliques To Get Fraud Users
In this phase,

- From the full graph constructed, cliques are found out with minimum 5 nodes in them.
- These cliques denote the users who are densely connected.
- These users are treated as fraud users.

Remove nodes with edge weights below threshold so normal users are treated as non-fraud users
In this phase,

- One nodes, all edges are taken. If all the edge weights are below the given threshold values, it means the user is giving rating less times only.
- The user is treated as normal user.

IV. EXPERIMENTAL RESULTS

The following Table 4.1 describes experimental result for Clique and Coarse Cluster analysis. The table contains finding number of Google App usage for attacks in malware social environments are shown.

Table 6.1 Fig 6.1 Clique and Coarse Cluster Performance Analysis

S.NO	Clique Techniques	Coarse Cluster
1	0.16	0.19
2	0.19	0.22
3	0.24	0.29
4	0.31	0.34
5	0.38	0.43
6	0.43	0.49
7	0.50	0.54
8	0.59	0.62
9	0.67	0.69
10	0.72	0.74

The following Fig 6.1 describes experimental result for Clique and Coarse Cluster analysis. The figure contains finding number of Google App usage for attacks in social environments are shown.

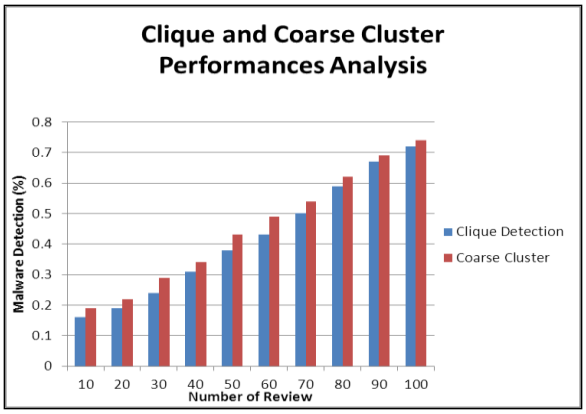

Fig 6.1 Clique and Coarse Cluster Performance Analysis

The following Table 6.2 describes experimental result for Clique and Coarse Cluster error rate analysis. The table contains Number application review and average percentages for CT and CC using malware detection are shown.

Table 6.2 Reduced Error Rate for Clique Detection and Coarse Cluster

Mobile Review	Clique Techniques (%)	Coarse Cluster (%)
10	72.54	78.62
20	76.13	78.11
30	82.42	83.13
40	86.66	84.67
50	88.13	89.78
60	80.44	82.66
70	78.33	80.21
80	87.22	89.76
90	79.22	80.65
100	91.22	92..62

The following Fig 6.2 describes experimental result for Clique and Coarse Cluster error rate analysis. The figure contains Number application review and average percentages for CT and CC using malware detection are shown.

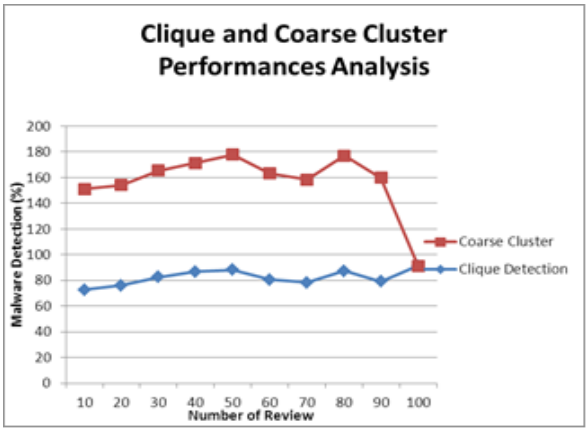

Fig 6.3 Reduced Error Rate for Clique and Coarse Cluster

V.RESULTS

- The statistical analysis of Malware app injection attacks data if prepared can be used for research development.
- N number of review can be found out easily where the injections are easy found out.
- The multimedia app attacks can also be detected
- The efficiency of the paper is further improved by improving coding efficiency
- In future, the time taken to complete the task is minimized
- Multitasking can also performed

VI CONCLUSION

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. This survey seeks to identify both malware and search rank fraud subjects in Google Play. This combination is not arbitrary: we posit that malicious developers resort to search rank fraud to boost the impact of their malware. Unlike existing solutions, this project builds this work on the observation that fraudulent and malicious behaviors leave behind telltale signs on app markets. The survey has introduced FairPlay, a system to detect both fraudulent and malware Google Play apps. The experiments on the twitter posts, have shown that a high percentage of fraud users are found. In addition, it recommendation for FairPlay's ability to discover non-fraud users also.

In the future, how to utilize inferred information and extend the framework for efficient and effective network monitoring and application design.

The new system become useful if the below enhancements are made in future. At present, number of posts/forum, average sentiment values/forums, positive % of posts/forum and negative % of posts/forums are taken as feature spaces for K-Means clustering. In future, neutral replies, multiple-languages based replies can also be taken as dimensions for clustering purpose. In addition, currently forums are taken for hot spot detection. Live Text streams such as chatting messages can be tracked and classification can be adopted. The new system is designed such that those enhancements can be integrated with current modules easily with less integration work and it becomes useful if the above enhancements are made in future.

REFERENCES

[1] Mahmudur Rahman, Mizanur Rahman, Bogdan Carbunar, and Duen Horng Chau, "Search Rank Fraud and Malware Detection in Google Play", IEEE Transactions On Knowledge And Data Engineering, Vol. 29, NO. 6, June 2017.

[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 Androiddevices,"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] 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.

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

[7] J. Sahs and L. Khan, "A machine learning approach to Android malware detection," in Proc. Eur. Intell. Secur. Inf. Conf., 2012, pp. 141–147.

[8] 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.

[9] 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.
[10] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine." Computer Networks and ISDN Systems, vol. 30, nos. 1-7, pp. 107-117, 1998.
[11] R. Cai, J.-M. Yang, W. Lai, Y. Wang, and L. Zhang, "iRobot: An Intelligent Crawler for Web Forums," Proc. 17th Int'l Conf. World Wide Web, pp. 447-456, 2008.
[12] A. Dasgupta, R. Kumar, and A. Sasturkar, "De-Duping URLs via Rewrite Rules," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 186-194, 2008.
[13] C. Gao, L. Wang, C.-Y. Lin, and Y.-I. Song, "Finding Question-Answer Pairs from Online Forums," Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 467-474, 2008.
[14] H.S. Koppula, K.P. Leela, A. Agarwal, K.P. Chitrapura, S. Garg, and A. Sasturkar, "Learning URL Patterns for Webpage De-Duplication," Proc. Third ACM Conf. Web Search and Data Mining, pp. 381-390, 2010.
[15] L. Zhang, B. Liu, S.H. Lim, and E. O'Brien-Strain, "Extracting and Ranking Product Features in Opinion Documents," Proc. 23rd Int'l Conf. Computational Linguistics, pp. 1462-1470, 2010.
[16] M.L.A. Vidal, A.S. Silva, E.S. Moura, and J.M.B. Cavalcanti, "Structure-Driven Crawler Generation by Example," Proc. 29thAnn. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 292-299, 2006.
[17] Y. Wang, J.-M. Yang, W. Lai, R. Cai, L. Zhang, and W.-Y. Ma, "Exploring Traversal Strategy for Web Forum Crawling," Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 459-466, 2008.
[18] J.-M. Yang, R. Cai, Y. Wang, J. Zhu, L. Zhang, and W.-Y. Ma, "Incorporating Site-Level Knowledge to Extract Structured Data from Web Forums," Proc. 18th Int'l Conf. World Wide Web, pp. 181-190, 2009.
[19] Y. Zhai and B. Liu, "Structured Data Extraction from the Web based on Partial Tree Alignment," IEEE Trans. Knowledge Data Eng., vol. 18, no. 12, pp. 1614-1628, Dec. 2006.