ANALYSIS ON DETECTING FRAUD RANKING FOR MOBILE APPLICATIONS

[1]Parvathi Modugu, [2]J.V Krishna

[1]Research Scholar, [2]Associate Professor & HOD
Department of Computer Science and Engineering
Sree Vahini Institute of Science and Technology, Tiruvuru. A.P.,India.

***ABSTRACT*: Cloud computing has recently emerged as a promising hosting platform that allows multiple cloud users called tenants to share a common physical computing infrastructure. With rapid adoption of the concepts of Software as a Service (SaaS) and Service Oriented Architecture (SOA), the Internet has evolved into an important service delivery infrastructure instead of merely providing host connectivity. In this project, we present IntTest, attestation scheme that can dynamically verify the integrity of data processing results in the cloud infrastructure and pinpoint malicious service providers when inconsistent results are detected. We validate service integrity by analyzing result consistency information with graph analysis. We proposed a new runtime service integrity attestation scheme that employs a novel attestation graph model to capture attestation results among different cloud nodes. We design attestation graph analysis algorithm to pinpoint malicious service providers and recognize colluding attack patterns. Our scheme can achieve runtime integrity attestation for cloud dataflow processing services using a small number of attestation data. Thus, our approach does not require trusted hardware or secure kernel co-existed with third-party service providers in the cloud. Int Test can achieve better scalability and higher detection accuracy than the state-of-the-art schemes. We extend our work in SAAS system to recommend links with improved trust results and with large number of service functions and services with consistency graph.**

***Keywords*: Cloud computing, Secured Data Processing.**

I.INTRODUCTION

The number of mobile Apps has grown at a breathtaking rate over the past few years. For example, as of the end of April 2013, there are more than 1.6 million Apps at Apple's App store and Google Play. To stimulate the development of mobile Apps, many App stores launched daily App leader boards, which demonstrate the chart rankings of most popular Apps. Indeed, the App leader board is one of the most important ways for promoting mobile Apps. A higher rank on the leader board usually leads to a huge number of downloads and million dollars in revenue. Therefore, App developers tend to explore various ways such as advertising campaigns to promote their Apps in order to have their Apps ranked as high as possible in such App leader boards. However, as a recent trend, instead of relying on traditional marketing solutions, shady App developers resort to some fraudulent means to deliberately boost their Apps and eventually manipulate the chart rankings on an App store. This is usually implemented by using so-called "both farms" or "human water armies" to inflate the App downloads and ratings in a very short time. For example, an article from Venture Beat reported that, when an App was promoted with the help of ranking manipulation, it could be propelled from number 1,800 to the top 25 in Apple's top free leader board and more than 50,000- 100,000 new users could be acquired within a couple of days. In fact, such ranking fraud raises great concerns to the mobile App industry. For example, Apple has warned of cracking down on App developers who commit ranking fraud in the Apple's App store. In the literature, while there are some related works, such as web ranking spam detection online review spam detection and mobile App recommendation the problem of detecting ranking fraud for mobile Apps is still under-explored. To fill this crucial void, in this paper, we propose to develop a ranking fraud detection system for mobile Apps. Along this line, we identify several important challenges. First, ranking fraud does not always happen in the whole life cycle of an App, so we need to detect the time when fraud happens. Second, due to the huge number of mobile Apps, it is difficult to manually label ranking fraud for each App, so it is important to have a way to automatically detect ranking fraud without using any benchmark information. Finally, due to the dynamic nature of chart rankings, it is not easy to identify and confirm the evidences linked to ranking fraud. Indeed, our careful observation reveals that fraudulent Apps do not always be ranked high in the leaderboard, but only in some leading events, which form different leading sessions. Note that we will introduce both leading events and leading sessions in detail later. In other words, ranking fraud usually happens in these leading sessions. Therefore, detecting ranking fraud of mobile Apps is actually to detect ranking fraud within leading sessions of mobile Apps. Specifically, we first propose a simple yet effective algorithm To identify the leading sessions of each App based on its historical ranking records. Then, with the analysis of Apps' ranking behaviors, we find that the fraudulent Apps often have different ranking patterns in each leading session compared with normal Apps. Thus, we characterize some fraud evidences from Apps' historical ranking records, and develop three functions to extract such ranking based fraud evidences. Nonetheless, the ranking based evidences can be affected by some legitimate marketing campaigns, such as "limited-time discount". As a result, it is not sufficient to only use ranking based evidences. Therefore, we further propose two functions to discover rating based evidences,

which reflect some anomaly patterns from Apps' historical rating records. In addition, we develop an unsupervised evidence aggregation method to integrate these two types of evidences for evaluating the credibility of leading sessions from mobile Apps. Figure 1 shows the framework of our ranking fraud detection system for mobile Apps. It is worth noting that all the evidences are extracted by modeling Apps' ranking and rating behaviors through statistical hypotheses tests. The proposed framework is scalable and can be extended with other domain-generated evidences for ranking fraud detection. Finally, we evaluate the proposed system with real-world App data collected from the Apple's App store for a long time period. Experimental results show the effectiveness of the proposed system, the scalability of the detection algorithm as well as some regularity of ranking fraud activities.

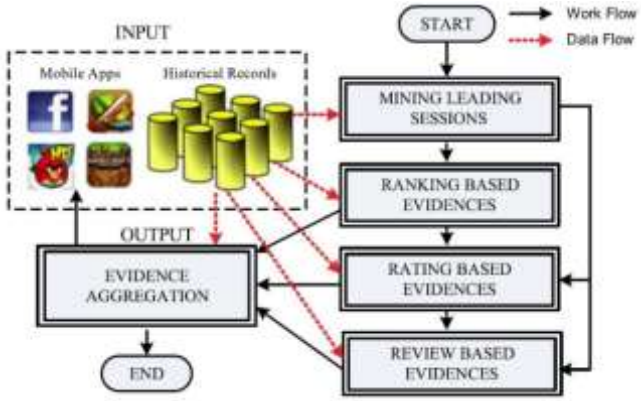

Fig. 1. The framework of our ranking fraud detection system for mobile Apps

II. RELATED WORK

The related works of this study is grouped into three categories. The first category is about Web ranking spam detection. Specifically, the Web ranking spam refers to any deliberate actions which bring to selected Web pages an unjustifiable favorable relevance or importance. In this, the problem of unsupervised web spam detection is studied. They introduce the concept of spam city to measure how likely a page is spam. Spam city is more flexible and user controllable measure than the traditional supervised classification methods. They propose efficient online link spam and term spam detection methods using spam city. These methods do not need training and also cost effective. A real data set is used to evaluate the effectiveness and the efficiency [1]. For example, Ntoulas et al. [2] have studied various aspects of content-based spam on the Web and presented a number of heuristic methods for detecting content based spam.

In this paper, they continue investigations of "web spam": the injection of artificially-created pages into the web in order to influence the results from search engines, to drive traffic to certain pages for fun or profit. This paper considers some previously- undescribed techniques for automatically detecting spam pages, examines the effectiveness of these techniques in isolation and when using classification algorithms aggregated. Zhou et al [1] have studied the problem of unsupervised Web ranking spam detection. Specifically, they proposed an efficient online link spam and term spam detection methods using spam city. Recently, Spirin et al. [3] have reported a survey on Web spam detection, which comprehensively introduces the principles and algorithms in the literature. Indeed, the work of Web ranking spam detection is mainly based on the analysis of ranking principles of search engines, such as Page Rank and query term frequency. This is different from ranking fraud detection for mobile Apps. They categorize all existing algorithms into three categories based on the type of information they use: content-based methods, link-based methods, and methods based on non-traditional data such as user behavior, clicks, HTTP sessions. In turn, there is a sub categorization of link-based category into five groups based on ideas and principles used: labels propagation, link pruning and reweighting, labels refinement, graph regularization, and feature based. The second category is focused on detecting online review spam. For example, Lim et al. [4] have identified several representative behaviors of review spammers and model these behaviors to detect the spammers. This paper aims to detect users generating spam reviews or review spammers. They identify several characteristic behaviors of review spammers and model these behaviors so as to detect the spammers. In particular, authors seek to model the following behaviors. First, spammers may target specific products or product groups in order to maximize their impact. Second, they tend to deviate from the other reviewers in their ratings of products. They propose scoring methods to measure the degree of spam for each reviewer and apply them on an Amazon review dataset. Authors then select a subset of highly suspicious reviewers for further scrutiny by user evaluators with the help of a web based spammer evaluation software specially developed for user evaluation experiments. Wu et al. [5] have studied the problem of detecting hybrid shilling attacks on rating data. The proposed approach is based on the semi-supervised learning and can be used for trustworthy product recommendation. This paper presents a Hybrid Shilling Attack Detector or HySAD for short, to tackle these problems. In particular, HySAD introduces MCRelief to select effective detection metrics, and Semi supervised Naive Bayes (SNBλ) to precisely separate Random-Filler model attackers and Average-Filler model attackers from normal users. Xie et al. [6] have studied the problem of singleton review spam detection. Specifically, they solved this problem by detecting the co-anomaly patterns in multiple review based time series. Although some of above approaches can be used for anomaly detection from historical rating and review records, they are not able to extract fraud evidences for a given time period (i.e., leading session). Finally, the third category includes the studies on mobile App recommendation. For example, Yan et al. [7] developed a mobile App recommender system, named Appjoy, which is based on user's App usage records to build a preference matrix instead of using explicit user ratings. Also, to solve the sparsity problem of App usage records, Shi et al. [8] studied several recommendation models and proposed a content based collaborative filtering model, named Eigenapp, for recommending Apps in their Web site Getjar. In addition, some researchers studied the problem of exploiting enriched contextual information for mobile App recommendation.

III. PROPOSED SYSTEM

In proposed system we overcome the drawbacks of Mining leading session algorithm which is based on ranking, review & rating. First, the download information is an important

signature for detecting ranking fraud, since ranking manipulation is to use so-called "bot farms" or "human water armies" to inflate the App download and ratings in a very short time. However, the instant download information of each mob. App is often not available for analysis. In fact, Apple and Google do not provide accurate download information on any App. Furthermore, the App developers themselves are also reluctant to release their download information for various reasons. Therefore, in this paper, the focus is on extracting evidences from Apps' historical ranking, rating and review records for ranking fraud detection. However, our approach is scalable for integrating other evidences if available, such evidences based on the download information and App developers' reputation. Second, the proposed approach can detect ranking fraud happened in A,' historical Ranking fraud detection in mobile apps is actually to detect ranking fraud within leading session of mobile apps. Specifically we identified first leading sessions based on Apps historical ranking records. Then with the analysis of Apps‘ranking behaviors we characterized some fraud evidences from historical records. The ranking based evidences can be affected some Apps‘developer reputation and some legitimate marketing campaigns, such as ―limited-time discount‖. This method is not enough to detect fraudulent Apps‘so we propose two new methods of fraud evidences based on Apps‘ historical rating and review records. Additionally, we developed an unsupervised evidence-aggregation method to integrate these types of evidences.

IV. EVIDENCE AGGREGATION ALGORITHM

1. Analyze the historical records of mobile apps.
2. Differentiate the evidences as Ranking based, Rating based, Review based.
3. Aggregate these evidences by using optimal aggregation algorithm.
4. Design Android application framework

Step1: Analyzing of historical records is nothing but obtaining the app related information from Google play store and apple store. Historical records consist Rank of the applications in leader board; Rating given by users to apps, different reviews of different types of users, no of downloads off the apps.

V. RANKING BASED EVIDENCES

According to the definitions introduced in Section 2, a leading session is composed of several leading events. Therefore, we should first analyze the basic characteristics of leading events for extracting fraud evidences. By analyzing the Apps‘ historical ranking records, we observe that Apps‘ ranking behaviors in a leading event always satisfy a specific ranking pattern, which consists of three different ranking phases, namely, rising phase, maintaining phase and recession phase. Specifically, in each leading event, an App‘s ranking first increases to a peak position in position for a period (i.e., maintaining phase), and finally decreases till the end of the event (i.e., recession phase) the leader board (i.e., rising phase).

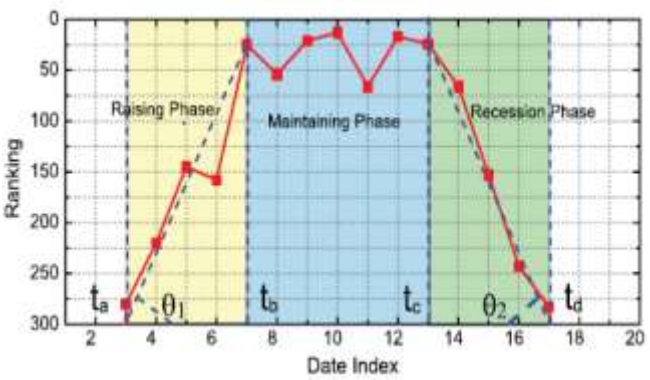

Fig2: Leader board (phases)

VI. RATING BASED EVIDENCES

User rating is one of the most important features of App advertisement. An App which has higher rating may attract more users to download and can also be ranked higher in the leader board. Thus, rating manipulation is also an important perspective of ranking fraud. Intuitively, if an App has ranking fraud in a leading session s, the ratings during the time period of s may have anomaly patterns compared with its historical ratings, which can be used for constructing rating based evidences. an App with rating manipulation might have surprisingly high ratings in the fraudulent leading sessions with respect to its historical ratings.

VII. REVIEW BASED EVIDENCES

Besides ratings, most of the App stores also allow users to write some textual comments as App reviews. Such reviews can reflect the personal perceptions and usage experiences of existing users for particular mobile Apps. Indeed, review manipulation is one of the most important perspective of App ranking fraud. Specifically, before downloading or purchasing a new mobile App, users often first read its historical reviews to ease their decision making, and a mobile App contains more positive reviews may attract more users to download. Therefore, imposters often post fake reviews in the leading sessions of a specific App in order to inflate the App download, and thus propel the App‘s ranking In this section, They discuss FA (Fagin‘s Algorithm) [Fag99]. This algorithm is implemented in Garlic [CHS+95], an experimental IBM middleware system; see [WHRB99] for interesting details about the implementation and performance in practice. Chaudhuri and Gravano [CG96]consider ways to simulate FA by using __filter conditions‘‘, which might say, for example, that the color score is at least 0.2.6 FA works as follows. 1. Do sorted access in parallel to each of the m sorted lists Li: (By __in parallel‘‘, we mean that we access the top member of each of the lists under sorted access, then we access the second member of eachof the lists, and so on.)7 Wait until there are at least k __matches‘‘, that is, wait5QBIC is a trademark of IBM Corporation.6Chaudhuri and Gravano originally saw an early version of the conference paper (in the 1996 ACM Symposium on Principles of Database Systems) that expanded into the journal version [Fag99]. It is not actually important that the lists be accessed __in lockstep‘‘. In practice, it may be convenient to allow the sorted lists to be accessed at different rates, in batches, etc. Each of the algorithms in this paper where there is __sorted access in parallel‘‘ remain correct even when sorted access is not in lockstep. Furthermore, all of our instance

optimality results continue to hold even when sorted access is not in lockstep, as long as the rates of sorted access of the lists are within constant multiples of each other. R. Fagin et al. / Journal of Computer and System Sciences 66 (2003) 614–656 619until there is a set of at least k objects such that each of these objects has been seen in each of the m lists. 2. For each object R that has been seen, do random access as needed to each of the lists Li to find the ith field xi of R: 3. Compute the grade tðRÞ ¼ tðx1;y; xmÞ for each object R that has been seen. Let Y be a set containing the k objects that have been seen with the highest grades (ties are broken arbitrarily). The output is then the graded set fðR; tðRÞÞ j RAYg:It is fairly easy to show [Fag99]that this algorithm is correct for monotone aggregation functions t (that is, that the algorithm successfully finds the top k answers). If there are N objects in the database, and if the orderings in the sorted lists are probabilistically independent, then the middleware cost of FA is OðNðm 1Þ=mk1=mÞ; with arbitrarily high probability [Fag99]. An aggregation function t is strict [Fag99]if tðx1;y; xmÞ ¼ 1 holds precisely when xi ¼ 1 for every i: Thus, an aggregation function is strict if it takes on the maximal value of 1 precisely when each argument takes on this maximal value. We would certainly expect an aggregation function representing the conjunction to be strict (see the discussion in [Fag99]). In fact, it is reasonable to think of strictness as being a key characterizing feature of the conjunction Fagin shows that his algorithm is optimal with high probability in the worst case if the aggregation function is strict (so that, intuitively, we are dealing with a notion of conjunction),and if the orderings in the sorted lists are probabilistically independent. In fact, the access pattern of FA is oblivious to the choice of aggregation function, and so for each fixed database, the middleware cost of FA is exactly the same no matter what the aggregation function is. This is true even for a constant aggregation function; in this case, of course, there is a trivial algorithm that gives us the top k answers (any k objects will do) with Oð1Þ middleware cost. So FA is not optimal in any sense for some monotone aggregation functions t: As a more interesting example, when the aggregation function is max (which is not strict), it is shown in [Fag99]that there is a simple algorithm that makes at most mk sorted accesses and no random accesses that finds the top k answers. By contrast, as we shall see, the algorithm TA is instance optimal for every monotone aggregation function, under very weak assumptions.

VIII. MATHEMATICAL MODEL

Let S is the Whole System Consist of
S= {I, P, O}
I = Input.
I = {U,Q,MA}
U = User
U = {u1,u2….un}
Q = Query Entered by user.
Q = {q1, q2, q3…qn}
MA = Mobile Apps
MA = {ma1, ma2, ma3…..man}
P = Process.

A. Mining Leading Sessions:

There are two main steps for mining leading sessions
1. We need to discover leading events from the App‘s historical ranking records.
2. We need to merge adjacent leading events for constructing leading sessions.

B. Ranking Based Evidences:

We should first analyze the basic characteristics of leading events for extracting fraud evidences. Therefore, we should first analyze the basic characteristics of leading events for extracting fraud evidences. 1. By analyzing the Apps‘ historical ranking records, we observe that Apps‘ ranking behaviors in a leading eventual ways satisfy a specific ranking pattern, which consists of three different ranking phases, namely

- Rising phase:
- Maintaining phase:
- Recession phase:

2. In each leading event, an App‘s ranking first increases to a peak position in the leader board (i.e., rising phase), then keeps such peak Position for a period (i.e., maintaining phase), and finally decreases till the end of the event.

C. Rating Based Evidences:

The ranking based evidences are useful for ranking fraud detection.

D. Review Based Evidences:

Most of the App stores also allow users to write some textual comments as App reviews. Such review scan reflects the personal perceptions and usage experiences of existing users for particular mobile Apps.

E. Evidence Aggregation:

After extracting three types of fraud evidences, the next challenge is how to combine them for ranking fraud detection.

- We propose an unsupervised approach based on fraud similarity to combine these evidences.
- We define the final evidence score ψ*(s) as a linear combination of all the existing evidences as Equation.
- We propose to use the linear combination because it has been proven to be effective and is widely used in relevant domains, such as ranking aggregation.

F. Android Framework

Android is a most powerful mobile platform and it powers hundreds of millions of mobile devices in more than 190 countries of the world. Android is a fully power packed operating system that provides strong base to the world supporting lakhs of applications and games for android users as well as an open marketplace supporting Android App Development. It gives you a single and a unique application model which enables you to deploy your apps broadly for Application development and App Development to hundreds of millions of users across a wide range of devices that is from phones to tablets and beyond. Android has undertaken15 powerful, open source and cross platform frameworks. These frameworks enhance Android App Development and Mobile App Development.

IX. CONCLUSION

In this paper, we developed a ranking fraud detection system for mobile Apps. Specifically, we first showed that ranking fraud happened in leading sessions and provided a method for mining leading sessions for each App from its historical ranking records. Then, we identified ranking based evidences, rating based evidences and review based evidences for detecting ranking fraud. Moreover, we proposed an mining Leading session algorithm for obtain

mining leading session and aggregation method. In the future, we plan to study more effective fraud evidences and analyze the latent relationship among rating, review and rankings. Moreover, we will extend our ranking fraud detection approach with other mobile App related services, such as mobile Apps recommendation, for enhancing user experience.

REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, ―Latent Dirichlet allocation,‖ J. Mach. Learn. Res., pp. 993–1022, 2003.
[2] Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou, ―A taxi driving fraud detection system,‖ in Proc. IEEE 11th Int. Conf. Data Mining, 2011, pp. 181–190.
[3] D. F. Gleich and L.-h. Lim, ―Rank aggregation via nuclear norm minimization,‖ in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 60–68.
[4] A. Klementiev, D. Roth, K. Small, and I. Titov, ―Unsupervised rank aggregation with domain-specific expertise,‖ in Proc. 21st Int. Joint Conf. Artif. Intell., 2009, pp. 1101–1106.
[5] A. Klementiev, D. Roth, and K. Small, ―Unsupervised rank aggregation with distance-based models,‖ in Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 472–479
[6] Hengshu Zhu, HuiXiong, Senior Member, IEEE, Yong Ge, and Enhong Chen, Senior Member, IEEE, ―Discovery of Ranking Fraud for Mobile Apps", vol.13,n0.1,Jan 2015
[7] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, ―Detecting spam web pages through content analysis,‖ in Proc. 15th Int. Conf. World Wide Web, 2006, pp. 83–92.
[8] N. Spirin and J. Han, ―Survey on web spam detection: Principles and algorithms,‖ SIGKDD Explor. Newslett., vol. 13, no. 2, pp. 50– 64, May 2012.
[9] B. Zhou, J. Pei, and Z. Tang, ―A spamicity approach to web spam detection,‖ in Proc. SIAM Int. Conf. Data Mining, 2008, pp. 277–288.
[10] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, ―Detecting product review spammers using rating behaviors,‖ in Proc.

M.PARVATHI M-Tech Dept. of CSE SreeVahini Institute of Science and Technology Tiruvuru Andhra Pradesh.

J.V Krishna AssocProfessor M-Tech Dept. of CSE SreeVahini Institute of Science and Technology Tiruvuru Andhra Pradesh.