International Journal of Engineering Research and Modern Education (IJERME)
ISSN (Online): 2455 - 4200
(www.rdmodernresearch.com) Volume I, Issue I, 2016

DETECTING RANKING FRAUD FOR MOBILE APPS AND REMOVE THE FRAUD MOBILE APPS

A. Karthiram* & K. Adlin Suji**
* PG Scholar, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu
** Associate Professor, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

Abstract:

Ranking fraud in the mobile App market refers to fraudulent or deceptive activities which have a purpose of bumping up the Apps in the popularity list. Indeed, it becomes more and more frequent for App developers to use shady means, such as inflating their Apps' sales or posting phony App ratings, to commit ranking fraud. The first propose to accurately locate the ranking fraud by mining the active periods, namely leading sessions, of mobile Apps. Furthermore, we investigate three types of evidences, i.e., ranking based evidences, rating based evidences and review based evidences, by modeling Apps' ranking, rating and review behaviors through statistical mining based hypotheses tests. In addition, the propose an optimization based application to integrate all the evidences for fraud detection based on EIRQ (efficient information retrieval for ranked query) algorithm. Finally, evaluate the proposed system with real-world App data collected from the iOS App Store for a long time period. In the experiments, validate the effectiveness of the proposed system, and show the scalability of the detection algorithm as well as some regularity of ranking fraud activities.

Index Terms: Mobile App, iOS App & App Ratings

1. Introduction:

In recent years smart mobile devices have become pervasive. More than 50 percent of all mobile phones are now smartphones, 1 and this statistic does not account for other devices such as tablet computers that are running similar mobile operating systems. According to Google, more than 400 million Android devices were activated in 2012 alone.

Android devices have widespread adoption for both personal and business use. From children to the elderly, novices to expert, and in many different cultures around the world, there is a varied user base for mobile devices. The ubiquitous usage of these mobile devices poses new privacy and security threats. Our entire digital lives are often stored on the devices, which contain contact lists, email messages, passwords, and access to files stored locally and in the cloud. Possible access to this personal information by unauthorized parties puts users at risk, and this is not where the risks end. These devices include many sensors and are nearly always with us, providing deep insights into not only our digital lives but also our physical lives. The GPS unit can tell exactly where you are, while the microphone can record audio, and the camera can record images.

Additionally, mobile devices are often linked directly to some monetary risks, via SMS messages, phone calls, and data plans, which can impact a user's monthly bill, or increasingly, as a means to authenticate to a bank or directly link to a financial account through a 'digital wallet'. This access means that any application (or app) that is allowed to run on the devices potentially has the ability to tap into certain aspects of the information.

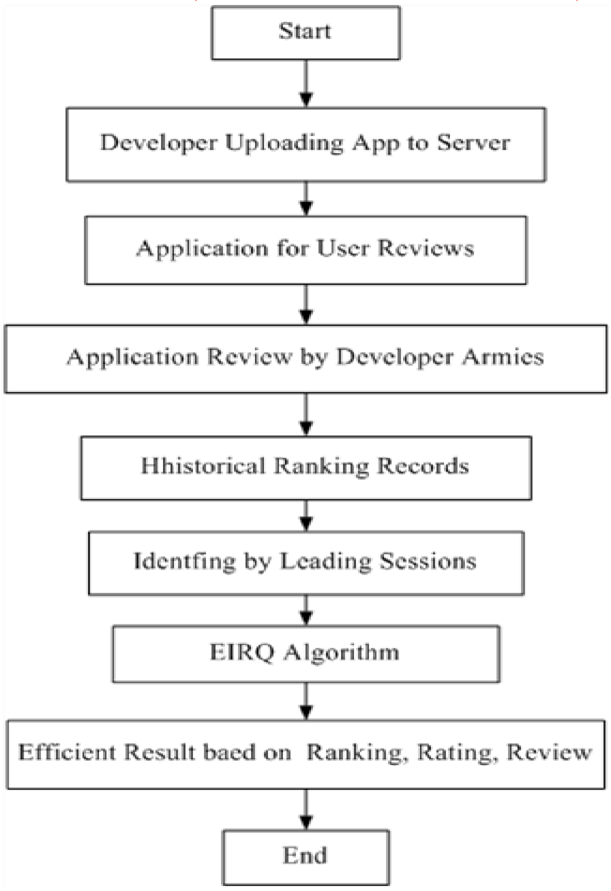

Start
Developer Uploading App to Server
Application for User Reviews
Application Review by Developer Armies
Hhistorical Ranking Records
Identfing by Leading Sessions
EIRQ Algorithm
Efficient Result baed on Ranking, Rating, Review
End

Figure 1: Data Flow Diagram

2. Related Works:

Choosing email as the spreading carrier of malware is not a new technique in the last decade. Early versions of email malware, such as Melissa and Love letter, work in a "naive" way. That is, a compromised user will send out malware emails only once, after which the user will not send out any further malware copies, even if she visits the malicious hyperlinks or attachments again. Take Melissa for example, the malware first checks a specific registry key in the Window OS and the malware will not do anything further when the value of this key suggests that the user has been infected before. In the following, we name this spreading mechanism as non reinjection. However, modern email malware is far more aggressive in spreading throughout email networks than before. Without checking if a computer has been infected before, modern email malware makes use of every chance to spread itself. We characterize its propagation with two kinds of new mechanisms, namely reinjection and self-start.way of extending the length range of the key shares to increase the attack cost substantially, and do some improvement Original Vanish system.

3. Proposed Work:

The instant download information of each mobile App is often not available for analysis. The App developers themselves are also reluctant to release their download information for various reasons. Therefore, in this paper, here mainly focus on ranking, rating and review records for ranking fraud detection using EIRQ (efficient information retrieval for ranked query) algorithm. Sometimes here need to detect such ranking fraud from Apps' current ranking observations. Actually, given the current ranking now of an App, here can detect ranking fraud for it in two different cases, such real-time ranking frauds also can be detected by the proposed approach of fraud detection system.

Advantages:

- ✓ Block the malware when the application downloaded.
- ✓ Download software based on risk score.
- ✓ Application uses safely.

Modules Description:

- ✓ Developer Uploading App to Server
- ✓ Application Review by Developer Armies
- ✓ Identifying the Mobile APPs mining Review
- ✓ Efficient result to predict the Fraud App based on EIRQ

Developer Uploading App to Server:

A server is both a running instance of some software capable of accepting requests from clients, and the computer such a server runs on. In today's technology lot of mobile application for messaging, browsing, editing etc. but this application may be created by the application developer and uploaded by the server. In server lot application is there based on the categories. It may be with original pattern rights or with duplicate malware application.

Application Review by Developer Armies:

First, the download information is an important signature for detecting ranking fraud, since ranking manipulation is to use so-called "bot farms" or "human armies" to inflate the App downloads and ratings in a very short time. However, the instant download information of each mobile App is often not available for analysis. Every application has some historical data due to the based on the reviews and response of the users. The review may be uploaded by the users or developer by fake ID. The App developers themselves are also reluctant to release their download information for various reasons for introduce the applications. Therefore, in this paper, here mainly focus on extracting evidences from Apps' historical ranking, rating and review records for reach the application to people usage priority increasing.

Identifying the Mobile Apps Mining Review:

There are two main steps for mining application review. First, here need to discover leading events from the App's historical ranking records of their own app by developer. Second, here need to merge adjacent leading events for constructing leading reviews of the other application. And approach is scalable for integrating other evidences if available, such as the evidences based on the download information and App developers' reputation. Specifically, here first extract individual leading feedback review event for the given App from the beginning time. Effective mining technique to identify the leading sessions of each app based on the historical ranking records.

4. Experimental Analysis and Results:

There are multiple storage services for a user to store data. Meanwhile, to avoid the problem produced by the centralized "trusted" third party, the responsibility of SeDas is to protect the user key and provide the function of self-destructing data. Fig. 4 shows the brief structure of the user application program realizing storage process. In this structure, the user application node contains two system clients: any third-party data storage system (TPDSS) and SeDas. The user application program interacts with the SeDas server through SeDas' client, getting data storage service.

Implementation is often used in the tech world to describe the interactions of elements in programming languages. In Java, where the word is frequently used, to implement is to recognize and use an element of code or a programming resource that is written into the program. One aspect of implementing an interface that can cause confusion is the requirement that to implement an interface, a class must implement all of the methods of that interface. This can lead to error messages due to insufficient implementation of methods. In general, the syntactical requirements of implementation and other tasks can be a burden for developers, and mastering this is part of becoming an in-depth user.

As mentioned, the difference of I/O process between SeDas and Native system (e.g. pNFS) is the additional Encryption/Decryption process which needs support from the computation resource of SeDas' client. Here compare two systems: i) a self-destructing data system based on active storage framework (SeDas for short), and ii) a conventional system without self-destructing data function (Native for short). Here evaluated the latency of upload and download with two schemes (SeDas and Native) under different file sizes. Also, here evaluated the overhead of encryption and decryption with two schemes under different file sizes. Figure shows the latency of the different schemes.

Implementation is the process of translating design specification in to source code. The primary goal of implementation is to write source code and internal implementation. So that conformance of code to its specification can be easily verified, So that debugging, testing and modification are eased. The source is developed with clarity, simplicity and elegance.

The coding is done in a modular fashion giving such importance even to the minute detail so, when hardware and storage procedures are changed or now data is added, rewriting of application programs is not necessary. To adapt or perfect use must determine new requirements, redesign generate code and test exiting software/hardware. Traditionally such task when they are applied to an existing program has been called maintenance.

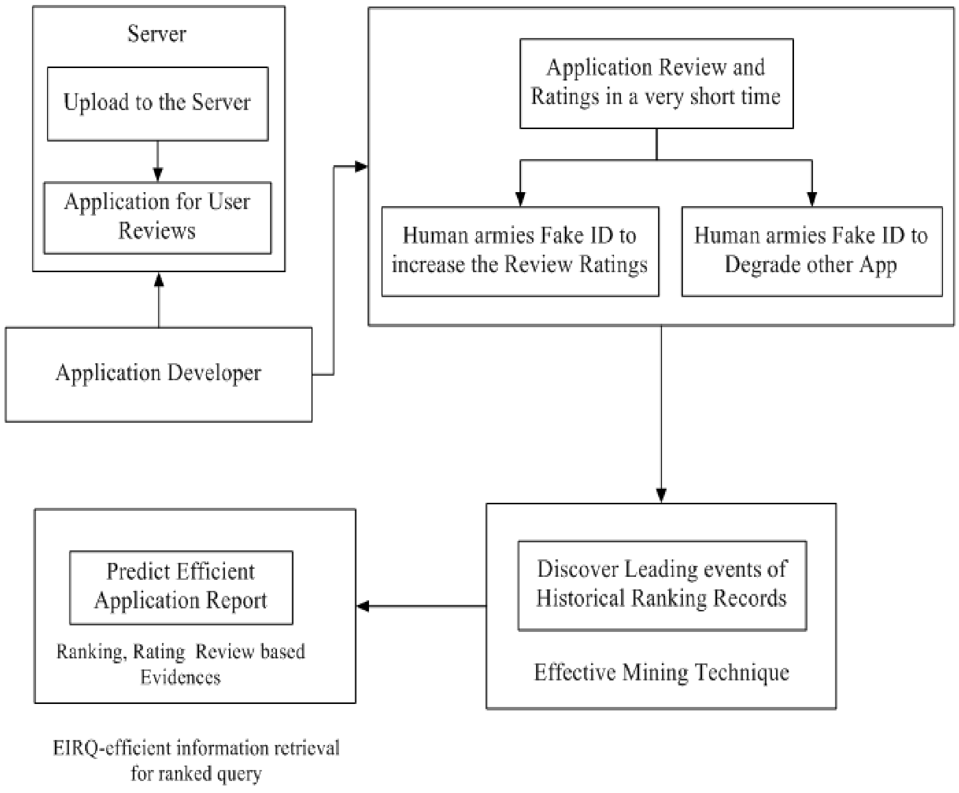
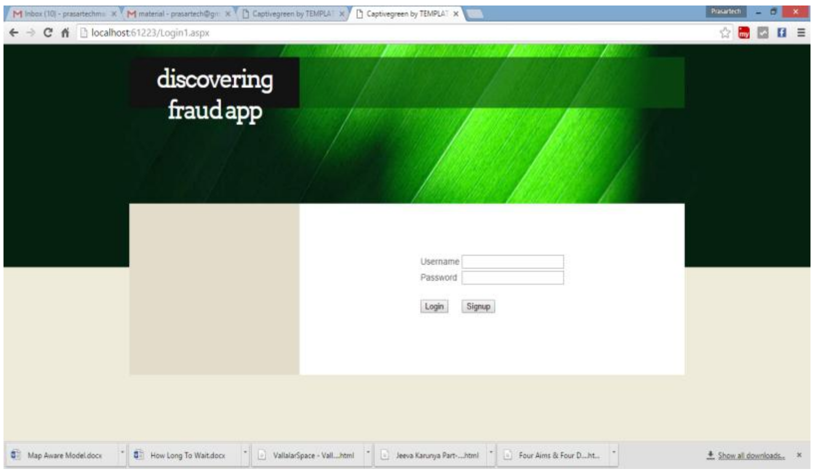
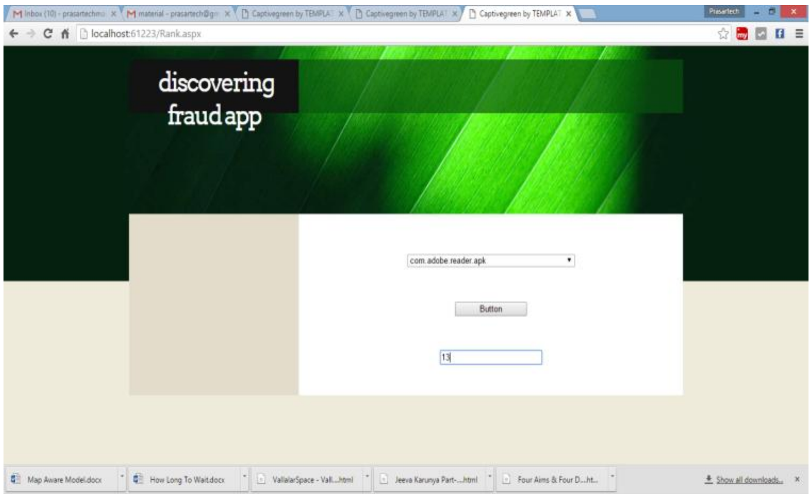

Server
Upload to the Server
Application for User Reviews
Application Developer
Application Review and Ratings in a very short time
Human armies Fake ID to increase the Review Ratings
Human armies Fake ID to Degrade other App
Discover Leading events of Historical Ranking Records
Effective Mining Technique
Predict Efficient Application Report
Ranking, Rating Review based Evidences
EIRQ-efficient information retrieval for ranked query

Figure 2: System Architecture

Login Page:



Ranking App:



5. Conclusion:

Here developed a ranking fraud detection system for mobile Apps. Specifically, here 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, here identified ranking based evidences, rating based evidences and review based evidences for detecting ranking fraud. Moreover, here proposed an optimization based aggregation method to integrate all the evidences for evaluating the credibility of leading sessions from mobile Apps. A unique perspective of this approach is that all the evidences can be modeled by statistical hypothesis tests, thus it is easy to be extended with other evidences from domain knowledge to detect ranking fraud. Finally, here validate the proposed system with extensive experiments on real-world App data collected from the App store. Experimental results showed the effectiveness of the proposed approach.

6. References:

1. K. Jin and E. L. Miller, "The effectiveness of De-duplication on virtual machine (VM) disk images," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ser. SYSTOR '09. Haifa, Israel: ACM, 2009, pp. 7:1–7:12.
2. K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An empirical analysis of similarity in virtual machine images," in Middle ware '11: Proceedings of the Middleware 2011 Industry Track Workshop. Lisbon, Portugal: ACM, 2011, pp. 6:1–6:6.

3. R. Schwarzkopf, M. Schmidt, M. R¨udiger, and B. Freisleben, “Efficient storage of virtual machine images,” in Science Cloud ’12: Proceedings of the 3rd Workshop on Scientific Cloud Computing Date. Delft, the Netherlands: ACM, 2012, pp. 51–60.
4. A. Muthitacharoen, B. Chen, and D. Mazi`eres, “A low-bandwidth network file system,” SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pp. 174–187, Oct. 2001.
5. M. Rabin, “Fingerprinting by random polynomials,” Center for Research in Computing Technology, Harvard University, Tech. Rep. TR-CSE-03-01, 1981.
6. B. Zhu, K. Li, and H. Patterson, “Avoiding the disk bottleneck in the data domain De-duplication file system,” in FAST’08: Proceedings of the6th USENIX Conference on File and Storage Technologies. San Jose,USA: USENIX Association, 2008, pp. 18:1–18:14.
7. C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak,J. Szczepkowski, C. Ungureanu, and M. Welnicki, “Hydrastor:a scalable secondary storage,” in FAST ’09: Proceedings of the 7thconference on File and storage technologies. San Francisco, USA: USENIX Association, 2009, pp. 197–210.
8. F. Guo and P. Efstathopoulos, “Building a high-performance De-duplication system,” in USENIXATC’11: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. Portland, USA: USENIX Association, 2011, pp. 25–39.
9. B. Nicolae, “Towards Scalable Checkpoint Restart: A Collective Inline Memory Contents Reduplications Proposal,” in IPDPS ’13: The 27thIEEE International Parallel and Distributed Processing Symposium, Boston, USA, 2013, pp. 19–28.