Fraud And Malware Detection Based On Reviews And Ratings

Shivani Nimse[1], Supriya Kolpe[2], Jagruti Sonar[3], Sayali Kekan[4], Prof. N. S. Patankar[5]

[1, 2, 3, and 4.] *BE Information Technology, Sanjivani COE, Kopargaon, Maharashtra, India*

[5] *Assistant Professor, Information Technology, Sanjivani COE, Kopargaon, Maharashtra, India*

ABSTRACT

To identify malware, previous work has focused on app executable and permission analysis. In this, we introduce FairPlay, a novel system that discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud. FairPlay correlates review activities and uniquely combines detected review relations with linguistic and behavioral signals gleaned from app data (87K apps, 2.9M reviews, and 2.4M reviewers, collected over half a year), in order to identify suspicious apps. FairPlay achieves over 95% accuracy in classifying user defined datasets of malware, fraudulent and legitimate apps. We show that 75% of the identified malware apps engage in search rank fraud. FairPlay discovers hundreds of fraudulent apps that currently evade Google Bouncer's detection technology. FairPlay also helped the discovery of more than 1,000 reviews, reported for 193 apps, that reveal a new type of "coercive" review campaign: users are harassed into writing positive reviews, and install and review other apps.

Keyword: - *android, review, ratings*

1. INTRODUCTION

Fraudulent developers frequently exploit crowd sourcing sites (e.g., Freelancer, Fever, and Best App Promotion) to hire teams of willing workers, to commit fraud collectively; emulating realistic, spontaneous activities from unrelated people (i.e., "crowdturfing"); we call this behavior as "search rank fraud". We build this work on the observation that fraudulent and malicious behaviors leave behind telltale signs on app markets. We uncover these nefarious acts by picking out such trails. For instance, the high cost of setting up valid Google Play accounts forces fraudsters to reuse their accounts across review writing jobs, making them likely to review more apps in common than regular users. Previous mobile malware detection work has focused on dynamic analysis of app executable as well as static analysis of code and permissions. However, recent Android malware analysis revealed that malware evolves quickly to bypass anti-virus tools. 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 executable (i.e., apks"), a set of required permissions, and a description. The app market publishes this information, along with the app's received reviews, ratings, 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. Each review consists of a star rating ranging between 1-5 stars, and some text. The text is optional and consists of a title and a description. Google Play limits the number of reviews displayed for an app to 4, 000. Figure 2 illustrates the participants in Google Play and their relations. Adversarial model. We consider not only malicious developers, who upload malware, but also rational fraudulent developers. Fraudulent developers attempt to tamper with the search rank of their apps, e.g., by recruiting fraud experts in crowd sourcing sites to write reviews, post ratings, and create bogus installs. While Google keeps secret the criteria used to rank apps, the reviews, ratings and install counts are known to play a fundamental part.

2. LITERATURE SURVEY

2.1 Android Leaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale

-by Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani

They applied framework which demonstrated by analyzing the data collected in the central server using two types of data sets:
1 .Artificial malware created for test purposes
2. Real malware found in the wild.

2.2 Andromaly: a Behavioural Malware Detection Framework for Android Devices.

- By Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss

They 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 and feature selection method to classify the collected data as normal (benign) or abnormal (malicious).

2.3 Risk ranker: Scalable and Accurate Zero-day Android Malware Detection.

-By Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang.

They propose a proactive scheme to spot zero-day Android malware. With- out relying on malware samples and their signatures, our scheme is motivated to assess potential security risks posed by these untreated apps.

2.4 Dissecting Android Malware: Characterization and Evolution.

-By Yajin Zhou and Xuxian Jiang.

In this paper, they focus on the Android platform and aim to systematize or characterize existing Android malware. The characterization and a subsequent evolution-based study of representative families reveal that they are evolving rapidly to circumvent the detection from existing mobile anti-virus software.

3. SYSTEM ARCHITECTURE

This system consists of four modules described as follows:

1. CoReg (coreview behaviour) Module
2. RF (Reviewer Feedback) Module
3. Inter-Review Relation Module
4. JH (Jekyll-Hyde) Module

3.1. CoReg (coreview behaviour) Module

This module 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. It identifies suspicious time-related coreview behaviour.

3.2. RF (Reviewer Feedback) Module

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. It detects suspicious behaviours reported by genuine reviews.

3.3. Inter-Review Relation Module

This module leverages temporal relations between reviews, as well as relations between the review, rating and install counts of apps, to identify suspicious behaviours. In order to compensate for a negative review, an attacker needs to post a significant number of Positive behaviour.

3.4. JH (Jekyll-Hyde) Module

This module identifies the permission ramps to pinpoint possible Jekyll-Hyde app Transactions. This module extracts following features: (i) the total number of permissions requested by the app, (ii) its number of dangerous permissions, (iii) the app's number of dangerous permission ramps, and (iv) its total number of Dangerous permissions added over all the ramps.

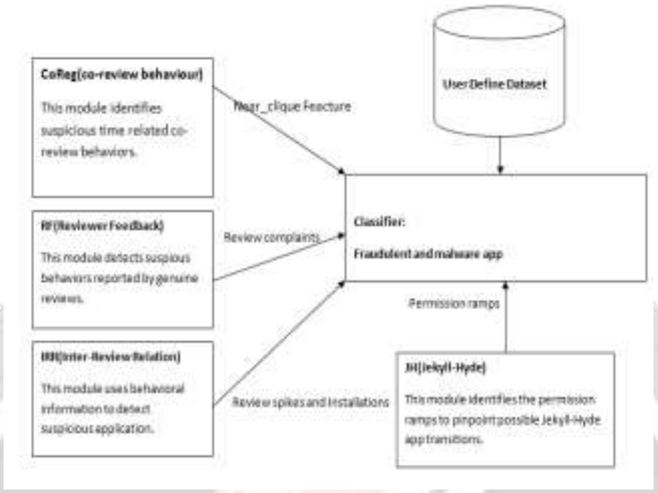
CoReg(co-review behaviour)
This module identifies suspicious time related co-review behaviors.
RF(Reviewer Feedback)
This module detects suspious behaviors reported by genuine reviews.
IRR(Inter-Review Relation)
This module uses behavioral information to detect suspiciout application.
Near_clique Feacture
Review complaints
Review spikes and installations
User Define Dataset
Classifier:
Fraudulent and malware app
Permission ramps
JH(Jekyll-Hyde)
This module identifies the permission ramps to pinpoint possible Jekyll-Hyde app transitions.

Fig -2: System Architecture

4. ALGORITHM

Algorithm: PCF algorithm pseudo-code.

Input: days, an array of daily reviews, and_, the weighted threshold density

Output: all Cliques, set of all detected pseudo-cliques

```
for d: =0 d < days.size (); d++
    Graph PC: = new Graph ();
    BestNearClique (PC, days[d]);
    c := 1; n := PC.size();
    for nd := d+1; d < days.size() & c = 1; d++
        bestNearClique(PC, days[nd]);
        c := (PC.size() > n); endfor
    if (PC.size() > 2)
        allCliques := allCliques.add(PC); fi endfor
return
function bestNearClique(Graph PC, Set revs)
    if (PC.size() = 0)
        for root := 0; root < revs.size(); root++
Graph candClique := new Graph ();
candClique.addNode (revs[root].getUser());
do candNode := getMaxDensityGain(revs);
if (density(candClique [ {candNode}) _ _))
candClique.addNode(candNode); fi
while (candNode != null);
if (candClique.density() > maxRho)
maxRho := candClique.density();
PC := candClique; fi endfor
else if (PC.size() > 0)
do candNode := getMaxDensityGain(revs);
if (density(candClique [ candNode) _ _))
PC.addNode(candNode); fi
while (candNode != null);
return
```

5. CONCLUSIONS

We have introduced FairPlay, a system to detect both fraudulent and malware Google Play apps. Our experiments on a newly contributed longitudinal app dataset have shown that a high percentage of malware is involved in search rank fraud; both are accurately identified by FairPlay. In addition, we showed FairPlay's ability to discover hundreds of apps that evade Google Play's detection technology, including new type of coercive attack.

6. REFERENCES

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

[2] .Ezra Siegel. Fake Reviews in Google Play and Apple App Store, Appentive, 2014.

[3].ENCK, W., GILBERT, P., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. N. Taint Droid: An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphone's. Tech. Rep. NAS-TR-0120-2010, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park,PA,USA,August2010

[4]. Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y. Zhao. Serf and Turf: Crowdturfing for Fun and Profit. In Proceedings of ACM WWW. ACM, 2012.

[5]. Jon Oberheide and Charlie Miller. Dissecting the Android Bouncer. SummerCon2012, New York, 2012.