ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

Detection of Malware and Fraud Ranking Using Fairplay System

P.Keerthi[1], M.Kiran Kumar[2], T.Kalpana[3], P.Karthisha[4]. Sk.Afrin[5]

U.G. Student, Department of CSE, Sree Venkateswara College of Engineering, Nellore, Andhra Pradesh, India[1,3,4,5]

Associate Professor, Department of CSE, Sree Venkateswara College of Engineering, Nellore, Andhra Pradesh, India[2]

ABSTRACT: Android has the major share in the mobile application market. Android mobile applications become an easy target for the attackers because of its open source environment. To identify fake and malware applications, all the previous methods focused on getting permission from the user and executing that particular mobile application. Here, we present FairPlay, A malware detection framework that discovers and break traces left behind by fraudulent developers, to detect search rank fraud as well as malware in Google Play. FairPlay, a system that efficiently detect Google Play fraud and malware. To detect fake reviews and rating in Google Play.

KEYWORDS: Android market, search rank fraud, malware detection.

I. INTRODUCTION

Google play first releases its app in 2008.Since that it distributes applications to all the Android users. In Google Play Store, it provides services that user can discover the particular application, purchase those applications and install it on their mobile devices. Since Android is open source environment all the detail about the application users can be easily accessed by the application developers through Google play. In Google play 1.8 Million mobile applications are available and that is downloaded by over 25 billion users across the world. This leads to greater chance of installing malware to the applications that could affect users mobile devices. Google play store uses its own security system known as Bouncer system to remove the malicious apps from its store. However, this method is not effective as testing some apps using virus tools many apps are found as malicious which are not detected by Bouncer system. 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.

Discovery of Ranking Fraud for Mobile Apps, Ranking fraud detection system for mobile Apps. Particularly, showing rank fraud will happen in primary sessions and provided a method for mining foremost sessions for each App from its historical ranking records. Then, they identified ranking based evidences, rating based evidences and review based evidences for detecting ranking fraud.

FairPlay: Fraud and Malware Detection in Google Play, FairPlay, it is a system to identify both false and malware Google Play apps. We discussed on a newly contributed longitudinal app data set, in which they had shown a high percentage of malware is involved in search rank fraud, both are accurately identified by FairPlay.

Mob Safe: Forensic Analysis for Android Applications and Detection of Fraud Apps Using CloudStack and Data Mining, Play store provides number of applications but few of those applications are fake. Such applications might damages the phone and also may be data hacked. Hence those kind of applications must be noticed, so that the apps will be identifiable for play store users. So we are proposing a web application which will process the information, comments and thee reviews of the application with natural language processing to give results in the form of graph. So it will be easier to decide which application is fraud or not. Multiple applications can be processed at a time with the

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

web application. Also User cannot always get correct or true reviews about the product on internet. So we can check for more than 2 sites, for reviews of same product. Hence we can get higher probability of getting real reviews.

Survey on Fraud Ranking in Mobile Apps, the fraud is happen any time during the whole life cycle of app, so the identification of the exact time of fraud is needed. Because of large number of android Apps, it becomes more tough to manually tag ranking fraud for each App, so it is important to automatically detect fraud without using any basic information. Android mobile Apps are not always ranked high in the leader board, but only in some leading events fraud usually happens in leading sessions. The main goal is to detect ranking fraud of Android mobile Apps with in leading sessions.

II. RELATED WORK

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:*

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. 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., [17]. Fraudulent developers often rely on crowdsourcing sites [16, 18, 19] 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.

III. FAIRPLAY SYSTEM MODEL

We now introduce FairPlay, a system to automatically detect malicious and fraudulent apps.

FairPlay System architecture, FairPlay organizes the analysis of longitudinal app data into the following 4 modules, illustrated in Figure 2. The Co-Review Graph (CoReG) module identifies apps reviewed in a contiguous time window by groups of users with significantly overlapping review histories. The Review Feedback (RF) module exploits feedback left by genuine reviewers. The Inter Review Relation (IRR) module leverages relations between reviews, ratings and install counts. The Jekyll-Hyde (JH) module monitors app permissions, with a focus on dangerous ones, to identify apps that convert from benign to malware. The CoReG module identifies suspicious, time related co-review behaviors. The RF module uses linguistic tools to detect suspicious behaviors reported by genuine reviews. The IRR module uses behavioral information to detect suspicious apps. The JH module identifies permission ramps to pinpoint possible Jekyll-Hyde app transitions.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

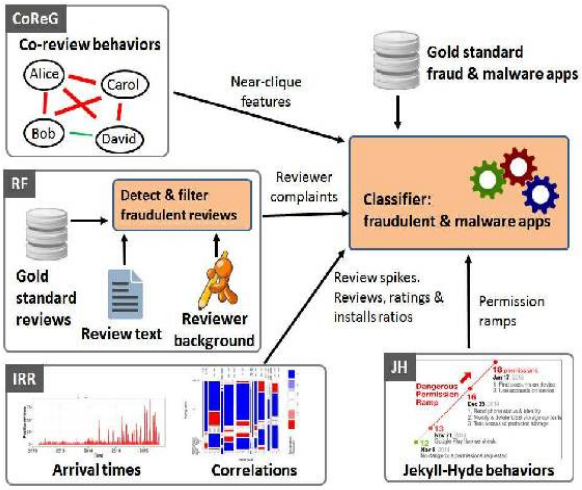

Figure.1: FairPlay system architecture.

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, ratings and installs, for a total of 28 features.

Notation	Definition
CoReG Module	
$nCliques$	number of pseudo cliques with $\rho \geq \theta$
stats(ρ)	clique density: max, median, SD
stats($cliqueSize$)	pseudo cliques size: max, median, SD
$inCliqueSize$	% of nodes involved in pseudo cliques
RF Module	
$malW$	% of reviews with malware indicators
$fraudW$, $goodW$	% of reviews with fraud/benign words
FRI	fraud review impact on app rating
IRR Module	
stats($spikes$)	days with spikes & spike amplitude
I_1/Rt_1, I_2/Rt_2	install to rating ratios
I_1/Rt_1, I_2/Rt_2	install to review ratios
JH Module	
$permCt$, $dangerCt$	# of dangerous and total permissions
$rampCt$	# of dangerous permission ramps
$dangerRamp$	# of dangerous permissions added

Table.1:FairPlay's Most Important Features, Organized by Their Extracting Module

A. *The Co-Review Graph (CoReG) 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. In the following, we describe the co-review graph concept, formally present the weighted maximal clique enumer- ation problem, then introduce an efficient heuristic that leverages natural limitations in the behaviors of fraudsters. Co-Review Graphs. Let the co-review graph of an app, see figure.2. be a graph where nodes correspond to user accounts who reviewed the app, and undirected edges have a weight that indicates the number of

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

apps reviewed in common by the edge's endpoint users. The co- review graph concept naturally identifies user accounts with significant past review activities.

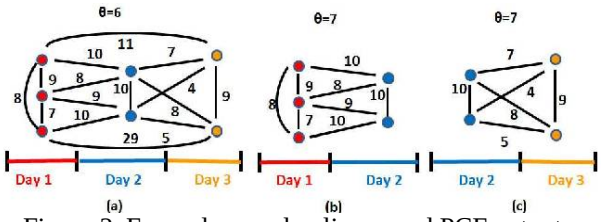

Figure.2. Example pseudo-cliques and PCF output.

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 subgraphs 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).

a) The Pseudo Clique Finder (PCF) algorithm:

This algorithm takes input as the set of the reviews of an app, organized by days, and a threshold value θ. PCF outputs a set of identified pseudo cliques with $\rho \geq \theta$, that were formed during contiguous time frames. we discuss the choice of θ. For each day when the app has received a review (line 1), PCF finds the day's most promising pseudo clique (lines 3 and 12−22): start with each review, then greedily add other reviews to a candidate pseudo clique; keep the pseudo clique (of the day) with the highest density. With that "work-in-progress" pseudo clique, move on to the next day (line 5): greedily add other reviews while the weighted density of the new pseudo clique equals or exceeds θ (lines 6 and 23 − 27). When no new nodes have been added to the work-in-progress pseudo clique (line 8), we add the pseudo clique to the output (line 9), then move to the next day (line 1). The greedy choice (getMaxDensityGain, not depicted in Algorithm 1) picks the review not yet in the work- in progress pseudo clique, whose writer has written the most apps in common with reviewers already in the pseudo clique. The output of PCF for several θ values. If d is the number of days over which A has received reviews and r is the maximum number of reviews received in a day, PCF's complexity is O(dr2(r + d)).

Algorithm 1. PCF Algorithm Pseudo-Code

```
Input: days, an array of daily reviews, 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); endfor
8.  if (PC.size() > 2)
9.    allCliques := allCliques.add(PC); fi endfor
10. return
11.function bestNearClique(Graph PC, Set revs)
12.    if (PC.size() = 0)
13.       for root := 0; root < revs.size(); root++
14.          Graph candClique := new Graph ();
15.candClique.addNode (revs[root].getUser());
16. do candNode := getMaxDensityGain(revs);
17.  if (density(candClique{candNode})>= θ))
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

```
18.    candClique.addNode(candNode); fi
19.      while (candNode != null);
20.      if (candClique.density() > maxRho)
21.           maxRho := candClique.density();
22.           PC := candClique; fi endfor
23. else if (PC.size() > 0)
24.  do candNode:= getMaxDensityGain(revs);
25.    if (density(candClique candNode) θ))
26.        PC.addNode(candNode); fi
27.    while (candNode != null);
28. return
```

Co-Review Graphs:

Let the co-review graph of an app, a graph where nodes correspond to user accounts who reviewed the app, and undirected edges have a weight that indicates the number of apps reviewed in common by the edge's endpoint users. The co-review clique of one of the seed fraud apps. The coreview graph concept naturally identifies user accounts with significant past review activities.

CoReG Features:

CoReG extracts the following features from the output of PCF (see Table 1) (i) the number of cliques whose density equals or exceeds u, (ii) the maximum, median and standard deviation of the densities of identified pseudo-cliques, (iii) the maximum, median and standard deviation of the node count of identified pseudo-cliques, normalized by n (the app's review count), and (iv) the total number of nodes of the co-review graph that belong to at least one pseudo-clique, normalized by n.

B. *Reviewer Feedback (RF) 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.

Step 1: Fraudulent Review 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.*

We used the NLTK library and the Naive Bayes classifier, trained on two sentences extracted from 700 positive and 700 negative IMDB movie reviews.

Step 2: Reviewer 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 fraud- ulent reviews, we extract feedback from the remain-ing reviews.

C. *Inter-Review Relation (IRR) Module:*

This module leverages temporal relations between reviews, as well as relations between the review, rating and install 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 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 buck- ets. Fig. 3. shows

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

the mosaic plot of the relationships between rating and install counts. p=0.0008924, thus we conclude dependence between the rating and install counts. The standardized residuals identify the cells (rectangles) that contribute the most to the x2 test. The most significant rating:install ratio is 1:100.

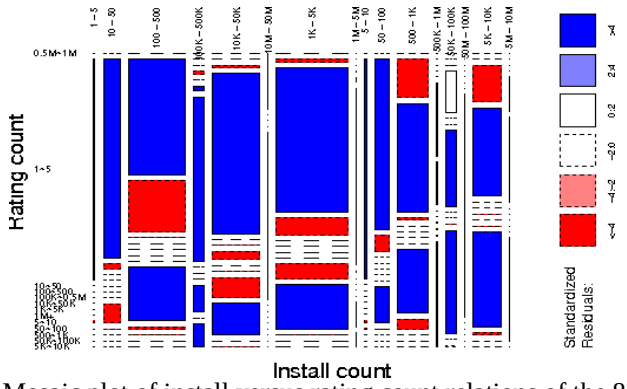

Figure.3. Mosaic plot of install versus rating count relations of the 87K apps.

Larger cells (rectangles) signify that more apps have the corresponding rating and install count range; dotted lines mean no apps in a certain install/rating category. The standardized residuals identify the cells that contribute the most to the x2 test. The most significant rating: install ratio is 1:100.

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

The distribution of the total number of permissions requested by malware, fraudulent and legitimate apps. Surprisingly , not only malware and fraudulent apps but also legitimate apps request large numbers of permissions.

IV. RESULTS

FairPlay has high accuracy and real-world impact

A. High Accuracy:

FairPlay get over 97 percent correctness in classifying falsified and gentle apps, and over 95 percent accuracy in classifying malware and gentle apps. In addition, we showed that malware regularly engages in search rank fraud as well as when trained on fraudulent and benign apps.

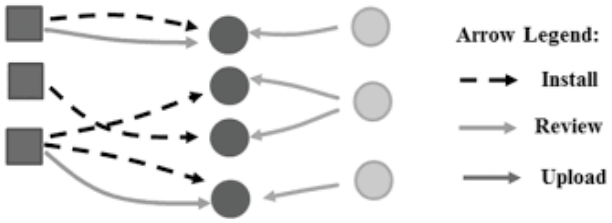

Fig.4. Google Play components and relations.

Google Play's functionality centers on apps, shown as red disks. Developers, shown as orange disks upload apps. A developer may upload multiple apps. Users, shown as blue squares, can install and review apps. A user can only review an app that he previously installed.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

B. Real-World Impact:

Uncover Fraud & Attacks. FairPlay discovers hundreds of fraudulent apps. We show that these apps are indeed suspicious: the reviewers of 93.3 percent of them form at least 1 pseudo-clique, 55 percent of these apps have at least 33 percent of their reviewers involved in a pseudo-clique, and the reviews of around 75 percent of these apps contain at least 20 words indicative of fraud. FairPlay also enabled us to discover a novel, coercive review campaign attack type, where app users are harassed into writing a positive review for the app, and install and review other apps.

V. EVALUATION

A. *Experiment Setup:*

We have implemented FairPlay using Python to extract data from parsed pages and compute the features, and the R tool to classify reviews and apps. We have set the threshold density value θ to 3, to detect even the smaller pseudo cliques. We have used the Weka data mining suite to perform the experiments, with default settings. We experimented with multiple supervised learning algorithms. Due to space constraints, we report results for the best performers: Multi Layer Perceptron (MLP), Decision Trees (DT) (C4.5) and Random Forest (RF), using 10-fold cross-validation. We use the term "positive" to denote a fraudulent review, fraudulent or malware app; FPR means false positive rate. Similarly, "negative" denotes a genuine review or benign app; FNR means false negative rate.

B. *Review Classification:*

To evaluate the accuracy of FairPlay's fraudulent review detection component (RF module), we used the gold standard datasets of fraudulent and genuine reviews of § 3.2. We used GPCrawler to collect the data of the writers of these re- views, including the 203 reviewers of the 406 fraudulent reviews (21,972 reviews for 2,284 apps) and the 315 re- viewers of the genuine reviews (9,468 reviews for 7,116 apps).

Strategy	FPR%	FNR%	Accuracy%
FairPlay/DT	3.01	3.01	96.98
FairPlay/MLP	1.51	3.01	97.74
FairPlay/RF	1.01	3.52	97.74

Table 2: FairPlay classification results (10-fold cross validation) of gold standard fraudulent (positive) and benign apps.

C. *App Classification:*

To evaluate FairPlay, we have collected all the 97,071 reviews of the 613 gold standard malware, fraudulent and benign apps, written by 75,949 users, as well as the 890,139 apps rated or played by these users.

D. *Fraud Detection Accuracy:*

Table 2. shows 10-fold cross validation results of FairPlay on the gold standard fraudulent and benign apps (see § 3.2). All classifiers achieve accuracies of around 97%. Random Forest is the best, having the highest accuracy of 97.74% and the lowest FPR of 1.01%. shows the co-review subgraph for one of the seed fraud apps identified by FairPlay's PCF. We observe that the app's reviewers form a tightly con- nected clique, with any two reviewers having reviewed at least 115 and at most 164 apps in common.

E. *Malware Detection Accuracy:*

We have used Sarma et al. solution as a baseline to evaluate the ability of FairPlay to accurately detect malware. We computed Sarma et al. RCP and RPCP indicators using the longitudinal app dataset. We used the SVM based variant of Sarma et al, which performs best. 10-cross validation results over the malware and benign gold standard sets. FairPlay significantly outperforms Sarma et al. solution, with an accuracy that consistently exceeds 95%. Random Forest has the smallest FPR of 1.51% and the highest accuracy of 96.11%. This is surprising: most FairPlay features are meant to identify search rank fraud, yet they also accurately identify malware. Is Malware Involved in Fraud? We conjectured that the above result is due in part to malware apps being involved in search rank fraud. To

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 2, February 2018

verify this, we have trained FairPlay on the gold standard benign and fraudulent app datasets, then we have tested it on the gold standard malware dataset. MLP is the most conservative algorithm, discovering 60.85% of malware as fraud participants. Random Forest discovers 72.15%, and Decision Tree flags 75.94% of the malware as fraudulent. This result confirms our conjecture and shows that search rank fraud detection can be an important addition to mobile malware detection efforts.

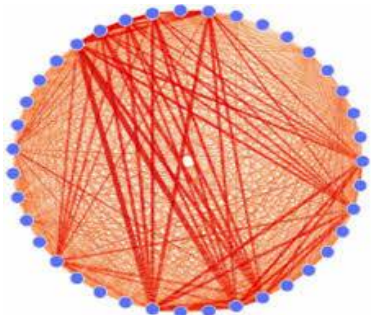

Figure 5: (a) Clique flagged by PCF for "Tiempo - Clima gratis", one of the 201 seed fraud apps (see § 3.2), involving 37 reviewers (names hidden for privacy); edge weights proportional to numbers of apps reviewed in common (ranging from 115 to 164 apps).

VI. CONCLUSION

Fair play has been introduced to detect both fraudulent and malware goggle play apps. We has been done our experiments on a newly contributed longitudinal app data set, have shown that high percentage of malware is involved in search rank fraud. Both are accurately identified by fair play and also fair plays ability to discover hundreds of apps that evade Google play detection technologies including a new type of coercive fraud attack.

REFERENCES

[1] Google Play. https://play.google.com/.
[2] Ezra Siegel. Fake Reviews in Google Play and Apple App Store. Appentive, 2014.
[3] Zach Miners. Report: Malware-infected Android apps spike in the Google Play store. PCWorld, 2014.
[4] Stephanie Mlot. Top Android App a Scam, Pulled From Google Play. PCMag, 2014.
[5] Daniel Roberts. How to spot fake apps on the Google Play store. Fortune, 2015.
[6] Andy Greenberg. Malware Apps Spoof Android Market To Infect Phones. Forbes Security, 2014.
[7] Jon Oberheide and Charlie Miller. Dissecting the Android Bouncer. SummerCon2012, New York, 2012.
[8] VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/, Last accessed on May 2015.
[9] Iker Burguera, Urko Zurutuza, and Simin Nadjm- Tehrani. Crowdroid: Behavior-Based Malware De- tection System for Android. In Proceedings of ACM SPSM, pages 15–26. ACM, 2011.
[10] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. Andromaly: a Behavioral Malware Detection Framework for Android Devices. Intelligent Information Systems, 38(1):161–190, 2012.
[11] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. Riskranker: Scalable and Accurate Zero-day Android Malware Detection. In Proceedings of ACM MobiSys, 2012.
[12] Bhaskar Pratim Sarma, Ninghui Li, Chris Gates, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. An- droid Permissions: a Perspective Combining Risks and Benefits. In Proceedings of ACM SACMAT, 2012.
[13] Hao Peng, Chris Gates, Bhaskar Sarma, Ninghui Li, Yuan Qi, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In Proceedings of ACM CCS, 2012.
[14] S.Y. Yerima, S. Sezer, and I. Muttik. Android Malware Detection Using Parallel Machine Learning Classifiers. In Proceedings of NGMAST, Sept 2014.
[15] Yajin Zhou and Xuxian Jiang. Dissecting Android Malware: Characterization and Evolution. In Proceedings of the IEEE S&P, pages 95–109. IEEE, 2012.