PREDICTION OF SPAM APP PUBLISHERS IN MOBILE AD NETWORKS

M. Sree Vani[1], Dr.R.Bhramaramba[2], Dr.D.Vasumati[3], O.Yaswanth Babu[4]

[1]*Dept of CSE, MGIT, Gandipet, Hyderabad, (India)*
[2]*Dept of IT, GITAM University, Vizag, (India)*
[3]*Dept of CSE, JNTUH, Hyderabad, (India)*
[4]*IT Manager, TCS, Gachibowli, Hyderabad, (India)*

ABSTRACT

Smart phone Apps plays a vital role to attract mobile-Advertising. Popular apps can generate millions of dollars in profit and collect valuable personal user information. Spam, i.e., fraudulent or invalid tap or click on online ads, where the user has no actual interest in the advertiser's site, results in advertising revenue being misappropriated by spammers. It requires a user touch or click on control ads came from Smartphone-game Apps. It all need the user to tap the screen close to where the ad is displayed .While ad networks take active measures to block click-spam today, but not in mobile advertising. The presence of spam in mobile advertising is largely unknown. In this paper, we take the first systematic look at spam in mobile advertising. Then we design a Graph based label propagation algorithm on click-through data to identify spam Apps in Smartphone-game Apps. We validate our methodology using data from major ad networks. Our findings highlight the severity of the spam in mobile advertising.

Keywords: Spam, Mobile Apps, Click Spam.

I. INTRODUCTION

The Smartphone and tablet markets are growing in leaps and bounds, helped in no small part by the availability of specialized third-party applications ("apps"). Whether on the iPhone or Android platforms, apps often come in two flavors: a free version, with embedded advertising, and a pay version without. Both models have been successful in the marketplace. Mobile advertisements within the apps are only source of revenue for several mobile app publishers. Maximum of the apps in the major mobile app stores show ads [1]. To embed ads in an app, the app developer typically registers with a third-party mobile ad network such as AdMob [2], iAd [3], Microsoft Mobile Advertising [4] etc. The ad net- works supply the developer with an ad control (i.e. library with some visual elements embedded within). The developer includes this ad control in his app, and assigns it some screen real estate. When the app runs, the ad control is loaded, and it fetches ads from the ad network and displays it to the user. Different ad networks use different signals to serve relevant ads. One of the main signals that mobile ad networks use today is the app metadata [24]. As part of the registration process, most ad networks ask the developer to provide metadata information about the app (for e.g. category of the app, link to the app store description etc.). This allows the ad network to serve ads related to the app metadata. Ad networks also receive dynamic signals sent by the ad control every time it fetches a new ad. Depending on the privacy policies and the security architecture of the platform, these signals can include the location, user identity, etc. Note that unlike JavaScript embedded in the browsers, the ad controls are integral parts of the application, and have access to the all the APIs provided by the platform.

IJATES

1.1 Background on Mobile Advertising

A typical mobile advertising system has five participants: mobile clients, advertisers, ad servers, ad exchanges and ad networks as Figure 2 shows. A mobile application includes an ad control module (e.g., AdControl for Windows Phones, AdMob for Android) which notifies the associated ad server any time an ad slot becomes available on the client's device. The ad server decides how to monetize the ad slot by displaying an ad. Ads are collected from an ad exchange. Ad exchanges are neutral parties that aggregate ads from different third party ad networks and hold an auction every time a client's ad slot becomes available. The ad networks participating in the exchange estimate their expected revenue from showing an ad in such an ad slot and place a bid on behalf of their customers (i.e., the advertisers). An ad network attempts to maximize its revenue by choosing ads that are most appropriate given the context of the user, in order to maximize the possibility of the user clicking on the ads. The ad network receives information about the user such as his profile, context, and device type from the ad server, through the ad exchange. Ad exchange runs the auction and chooses the winner with the highest bid. Advertisers register with their ad networks by submitting an ad campaign. A campaign typically specifies an advertising budget and a target number of impressions/clicks within a certain deadline (e.g., 50,000 impressions delivered in 2 weeks).

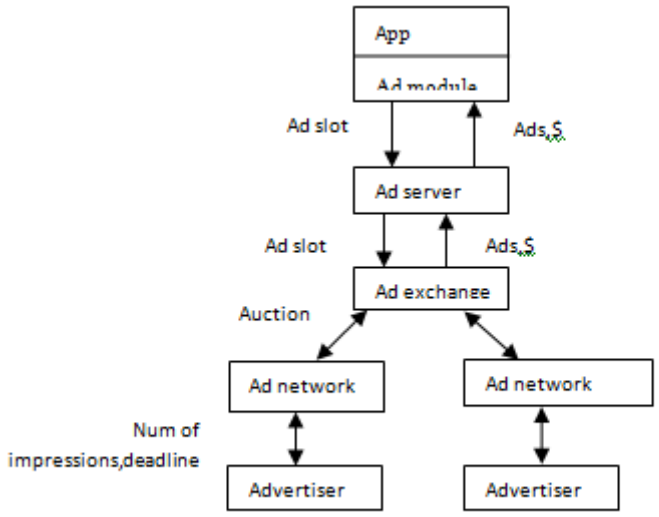

Fig 1: Architecture of a Typical Mobile Ad System

They can also specify a maximum cap on how many times a single client can see a specific ad and how to distribute ads over time (e.g., 150 impressions per hour). The ad server is responsible for tracking which ads are displayed and clicked, and thus determining how much money an advertiser owes. The revenue of an ad slot can be measured in several ways, most often by views (Cost Per Impression) or click-through (Cost Per Click), the former being most common in mobile systems. The ad server receives a premium on the sale of each ad slot, part of which is passed to the developer of the app where the ad was displayed.

1.2. Background and Motivation for Spam in Mobile Advertising

A mobile developer accidentally (or intentionally) places the in-app advertising control close to where the user must tap, or drag on usage of mobile. Given the tiny screen real-estate, the user is prone to mistapping. When he does so, the browser navigates to the ad-click URL. The user may realize his error and switch back to the game. The browser, which in the mean time has already begun fetching the ad landing-page, aborts the attempt. As a result, the user will appear to have spent very little time on the advertiser's page. We saw exactly this behavior on our mobile ads —95% of users spent less than a second as mentioned earlier.

The core issue here is the advertiser being charged despite the user not spending any time on the landing page. It is hard for an ad network to know how long the user spent on the advertiser's site. If it relied on the advertiser to get this information, the advertiser could easily lie to get a discount. Solving this without modifying the browser, and without hurting the user experience is a non-trivial problem. One mitigating approach would be to audit apps that trick users into mistapping on the ad. Doing so would likely spark an arms race for apps intentionally exploiting this loop-hole, but would at least protect advertisers from apps accidentally triggering this. Unfortunately, ad networks are making it harder for advertisers and independent third-parties to identify bad apps. The rest of the paper is organized as follows. In Section 2, we review approaches for click spam detection in previous work. In Section 3, we introduce our methodology for prediction of spam in Mobile Apps. Section 4 presents our novel Graph based label propagation algorithm. Section 5 describes our experiment setup and shows experimental results. Finally, our conclusions and future directions are presented in Section 6.

II. RELATED WORK

Existing works on ad fraud mainly focus on the click-spam behaviors, characterizing the features of click-spam, either targeting specific attacks [5, 6, 16, 18], or taking a broader view [7]. Some work has examined other elements of the click-spam ecosystem: the quality of purchased traffic [19, 20], and the spam profit model [12, 13]. Very little work exists in exploring clickspam in mobile apps. From the controlled experiment, authors in [7] observed that around one third of the mobile ad clicks may constitute click-spam. A contemporaneous paper [9] claimed that they are not aware of any mobile malware in the wild that performs advertising click fraud. DECAF focuses on detecting violations to ad network terms and conditions, and even before potentially fraudulent clicks have been generated. With regard to detection, most existing works focus on bot-driven click spam, either by analyzing search engine query logs to identify outliers in query distributions [52], characterizing networking traffic to infer coalitions made by a group of bot-driven fraudsters [14, 15], or authenticating normal user clicks to filter out bot-driven clicks [10, 11, 49]. A recent work, Viceroi [8], designed a more general framework that is possible to detect not only bot-driven spam, but also some non-bot driven ones (like search-hijacking).To the best of our knowledge, ours is the first work to detect touch spam in mobile apps.

III. METHODOLOGY

Based on these observations, we design a label propagation algorithm on click-through data. Firstly, a small number of seed game-apps are selected and labeled as spam or non-spam. Then their labels are propagated on the click-through bipartite graph and other possible spam/non-spam game-apps are identified. The input consists of a) a set of labeled game-apps (spam or non-spam), b) a set of unlabeled game-apps and c) a set of constraints between game-apps and the ad-controls in the log. The goal is to find spam game-apps which are misplaced ad-controls from the unlabeled data.

3.1. Advertiser Web Server Log Contains Click-Through Data C and Bipartite Graph G

The click log consists of triples <ad-c, g-app, fag>, where ad-c is a ad-control clicked by gamer or player, g-app is publisher website which serves an ad to user by fetching from ad-network and fag is the Number of times that ad-control ad is clicked when user is playing game g. Define AD = {ad-c | ad-c appears in C} and GA = {g-app | g-app appears in C}.

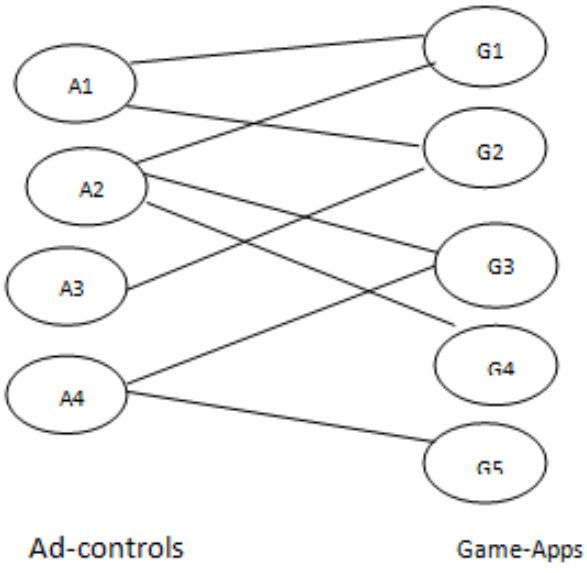

Fig 2: Bipartite Graph G

Click-through data C has an equivalent form – a click-through bipartite graph G = (AD, GA, E). There are two different types of nodes, ad-controls and GAs in G. For every record <ad-c, g-app, fag> in C, there is an edge (ad-c, g-app) ∈E with weight fag. Each ad-c/g-app is assigned with a probability pad-c/pg-app, which denotes how likely this ad-c/g-app is to be a spam game app which is misplaced ad-control in game or in other words, the spamicity of ad-c/g-app. Note that the click-through bipartite graph can be constructed either on page-level or site-level. In the latter form, u is replaced by its site but not the URL of itself. For example, <"Nokia", http://product.pcpop.com/Mobile/00283_1.html, 100> is replaced
by <"Nokia", http://product.pcpop.com/, 100>.

3.2. Labeled Seed GS set L

L contains all of the game-apps in C (G) that are manually labeled as spam or non-spam. More formally, L = {g-app | g-app is labeled as a spam game-app or non-spam game-app}. We will discuss the construction details of L in Section 5.2.

3.3. GA Result Set GU

Set GU contain all the <g-app, pg> and <ad-c, pa> pairs, respectively. After our algorithm ends, each GA g-app or ad-control ad-c in C (or G) will be assigned with a probability pg/pa, which denotes the probability that this GA or ad-control is a spam game-app which are misplaced ad-control. More formally,
GU = {<g-app, pg> | pg is the spamicity score for g-app}.
Given G = (AD, GA, E) and L∈GA, the goal of the spam game apps mining problem is to obtain the results set GU , which contain all of the possible spam game apps which are misplaced ad-control in G.

IV. A GRAPH BASED LABEL PROPAGATION ALGORITHM

4.1 Algorithm Design

In this paper, we propose label propagation (LP) algorithm to solve the problem that is defined in the previous section. More specifically, for every game-app g-app,ad-control ad-coquetry q, we could calculate the probability pg that g-app is a spam game-app by incorporating all of the label information of its neighbors. We describe this procedure more formally as follows.

For ad-c/g-app, we use la/lg to denote its label, which is S for spam and N for non-spam. Note that P(lg=N) = 1-P(lg=S). Thus every GA g-app in labeled set L would have P(lg=S)=1 or P(lg=S)=0 initially and every GA g-app in the set GA-L would have P(lg=S)=0.Then we have

$$p(l_g = S) = \sum_{a:(a,g)\in E} \omega_{ga} P(l_a = S) \quad (1)$$

where

$$\omega_{ga} = \frac{f_{ag}}{\sum_{a:(a,g)\in E} f_{ag}} \quad (2)$$

is the transition probability from GA g to ad-control a.

Similarly, for each ad-control AD a in GA\L, the probability $P(l_a = S)$ is computed as

$$p(l_a = S) = \sum_{g:(a,g)\in E} \omega_{ag} P(l_g = S) \quad (3)$$

where

$$\omega_{ag} = \frac{f_{ag}}{\sum_{g:(a,g)\in E} f_{ag}} \quad (4)$$

is the transition probability from ad-control a to GA g. Note that both ω_{ag} and ω_{ga} are not limited to the above form but arbitrary. The only requirement for them is they should have a probability interpretation, which means $\sum_{a:(a,g)\in E} \omega_{ag} = 1$ and $\sum_{a:(a,g)\in E} \omega_{ga} = 1$.Using Equation (1) and (3), we can obtain P(lg=S) and P(la=S) recursively for all of the GAs in the click-through bipartite graph. We can have a concise representation of this iterative process. Suppose that there are |AD| ad-controls: a1, a2…a|AD| and |GA| Game-apps: g1, g2,…g|GA|. Define vectors: PAD=(P(la1=S), P(la2=S)…P(l a|AD|=S))T, PGA=(P(lg1=S), P(lg2=S)…P(lg|GA| =S))T, and the transition probability matrixes: Mag= (ω_{ag}) |AD|x|GA|, and Mga= (ω_{ga}) |GA|x|AD .Then in the ith iteration, we have PiAD=Mag Pi-1 GA , Pi GA= MgaPi AD

It should be noted that in each round of iteration, all of the GAs in seed set L should be re-assigned their initial labels. In this way, the algorithm converges. We will prove the convergence

in section 4.4. The outline of the Grah based Label Propagation algorithm is shown in Figure 3.

IJATES

Graph Based Label Propagation Algorithm

Input : labeled seed set L,click-through data C(G)

Output : P(lg=S) and P(la=S) for all GAs and ad-controls in G

Begin

Do

for (g∈L, set P(lg=S)=1 or 0 according to their label by human assertors.)

for (all a∈AD) do

$$p(l_a = S) = \sum_{g:(a,g)\in E} \omega_{ag} P(l_g = S)$$

end for

for (all g∈GA\L)do

$$p(l_g = S) = \sum_{a:(a,g)\in E} \omega_{ga} P(l_a = S)$$

end for

until convergence

Output P(lg=S) for every GameApp g in GA and P(la=S) for every ad-control a in AD

End

Fig 3: The Graph Based Label Propagation Algorithm

4.2 Convergence Of The LP Algorithm

It is evident that Mag and Mga are right stochastic matrixes, each of whose rows consists of nonnegative real numbers, with each row summing to 1. Then consider Mgg=MgaMag. For each element mij in Mgg, we have $m_{ij} = \sum_k \omega_{ik}\omega'_{kj}$ in Mgg, where ω_{ik} and ω'_{kj} are elements Mga and Mag, respectively. Thus we have

$$\sum_j m_{ij} = \sum_j \sum_k \omega_{ik}\omega'_{kj}$$

$$= \sum_j \sum_k \omega_{ik}\omega'_{kj}$$

$$= \sum_k \omega_{ik} \sum_j \omega'_{kj}$$

$$= \sum_k \omega_{ik}$$

$$= 1$$

which means that Mgg is also a right stochastic matrix.Now, if we are only interested in PGA, the iteration process can be rewritten as PiGA= MggPi-1GA= MgaMag P i-1GA, where i denotes the iterations.Suppose

IJATES

that there are |L| seed GAs in L, |C| 1-degree nodes and thus r = |GA|-|L|-|C| remaining GAs in C. More specifically,let the probability vector PGA=（PT PL） where PT are the top |L|+|C| rows of PGA(the labeled data and the pseudo labeled data) and PL are the remaining r rows of PGA(the unlabeled data). We split Mgg after the (|L|+|C|)th row and the (|L|+|C|)th column into 4sub-matrixes

$$M_{gg} = \begin{bmatrix} M_{(|L|+|C|)(|L|+|C|)} & M_{(|L|+|C|)r} \\ M_{r(|L|+|C|)} & M_{rr} \end{bmatrix}$$

.Note that PT never really changes. It can be shown that in our algorithm,

$\mathbf{P}_L = \boldsymbol{M}_{rr}\mathbf{P}_L + M_{r(|L|+|C|)}\,\mathbf{P}_T$ which lead to $\mathbf{P}_L = \lim_{n\to\infty}(\boldsymbol{M}_{rr})^n\mathbf{P}_L^0 + [\sum_{i=1}^{n}\boldsymbol{M}_{rr}^{i-1}]M_{r(|L|+|C|)}\,\mathbf{P}_T$. Zhu and Ghahramani [27] proved that PL converges to (I-Mrr)-1$M_{r(|L|+|C|)}\,\mathbf{P}_T$.□ if Mgg is a right stochastic matrix. Thus the initial value of PL is inconsequential. Using the same approach, we could prove that PAD also converges.

V. EXPERIMENTS

The goal of the experiments is to evaluate how effective our algorithm is in detecting spam Game Apps. Given a seed set L, the LP algorithm returns a list of game apps that are sorted according to their probability of being spam. Seed game apps are not included in the list. We also obtain a list of ad-controls that are sorted according to their probability of being used as a spam-oriented ad-control.

5.1 Datasets

We signed-up as an advertiser with a few major ad networks which creates our datasets. We log all web requests made to our server. The logs used in this study are standard Apache web server logs that include the user's IP address, date and time of access, URL accessed (of a page on our webserver) along with any GET parameters, the HTTP Referer value and User-Agent value sent for that request, and a cookie value we set the first time we see a user to identify repeat visits from the same user. Our datasets also consist of all selected game apps crawled from the Apple iOS App Store in 2012. We collected 13,267 top free game apps from App Store.

5.2 Bipartite Graph Construction

We pruned the entire ad -game apps pairs with just one click on any day in the log since they may contain noise and possible privacy information. After that, this click-through log consisted of 24,435 unique ad-controls, with 34,708 unique game apps in 1,055 sites. Altogether, 50,660 ad-game apps pairs were collected and they were used in constructing the bipartite graph. The maximal component of the graph contains 25.0% unique ad-controls, 29.0% game apps and 44.2% ad-game apps pairs.

From our datasets, we computed metadata for apps, developers, and users who post reviews .We divide the whole data sets into two parts as training dataset and test dataset. We then went for manual labeling of training datasets. To label apps as spam or non spam, we invited some volunteers who had experience with mobile game playing to participate. After labeling process, we have 81 spam posts (17%) and 401 non spam posts (83 %.).

5.3 Results

We would like to detect as many spam apps as possible while avoiding misclassifying non-spam ones. We conduct various experiments with our dataset using our Label propagation algorithm. We compared our algorithm against with pagerank and Trust algorithm. The experimental results shown in the Figure 4.

IJATES

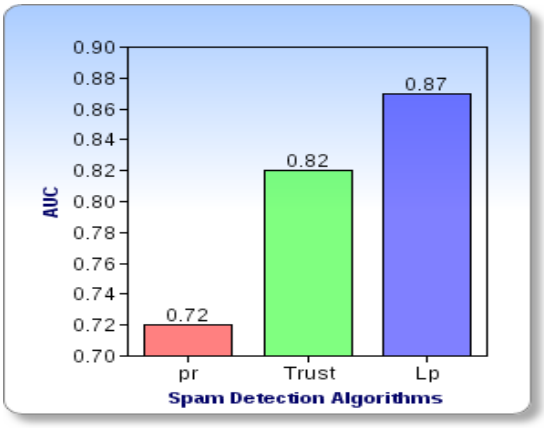

Fig 4: AUC Values for Different Spam Detection Algorithms.

From the Figures, we can see that all of the AUC values of the five algorithms are greater than 0.78, which suggests that they are effective in detecting Web spam. It is not surprising that pagerank performs the worst because spam sites can boost their pagerank scores using tricks such as the link-farm. TRUST works better than PR, which is consistent with previous research (10). The AUC of LP is 0.870, which is much better than both PR. And TRUST. This illustrates our algorithm gives better performance in detecting web spam sites. To test how robust our algorithm, we conducted experiment on seed selection .We randomly slit our spam sites into 21 subsets (each with 100 seed sites and then add them gradually into seed set. The experiment results are summarized in Figure 5. It can be seen that all of our algorithms are very robust. They can achieve a relatively high AUC value after only 400 sites are added into the seed sets. We also notice that LP performs consistently better than PR and TRUST.

Fig 5: Algorithm Performance with Different Seed Sets

VI. CONCLUSION

In this paper we have proposed a Graph based Labeled propagation algorithm to prediction spam in Mobile game apps. This Algorithm constructed bipartite graph G from Advertiser web server log data and generates Labeled Seed set L. Then finally extracts the resultant set GU, which contains all of the possible spam game apps which are misplaced ad-control in G. Experiment results show that our algorithm is both efficient and effective in predicting spam in mobile game Apps. For future work, we plan to combine our algorithm with some current anti-spam techniques results in a much better performance.

REFERENCES

[1] S. Ganov, C. Killmar, S. Khurshid, and D. Perry. Event listener analysis and symbolic execution for testing gui applications. In ICFEM, 2009.

[2] Google admob. http://www.google.com/ads/admob/.

[3] iad app network. http://developer.apple.com/support/appstore/iad-app-network/.

[4] Microsoft advertising. http://advertising.microsoft.com/en-us/splitter.

[5] S. Alrwais, A. Gerber, C. Dunn, O. Spatscheck,M. Gupta, and E. Osterweil. Dissecting ghost clicks: Ad fraud via misdirected human clicks. In ACSAC, 2012.

[6] T. Blizard and N. Livic. Click-fraud monetizing malware: A survey and case study. In MALWARE,2012.

[7] P. Chia, Y. Yamamoto, and N. Asokan. Is this app safe? a large scale study on application permissions and risk signals. In WWW, 2012.

[8] V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In ACM SIGCOMM, 2012.

[9] C. Cadar D. Dunbar and D. Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In USENIX OSDI, 2008.

[10] P. Gilbert, B. Chun, L. Cox, and J. Jung. Vision:automated security validation of mobile apps at app markets. In MCS, 2011.

[11] H. Haddadi. Fighting online click-fraud using bluff ads. ACM Computer Communication Review, 40(2):21–25, 2010.14

[12] C. Hu and I. Neamtiu. Automating gui testing for android applications. In AST, 2011.

[13] A. MacHiry, R. Tahiliani, and M. Naik. Dynodroid: An input generation system for android apps. In FSE, 2013.

[14] A. Mesbah and A. van Deursen. Invariant-based automatic testing of ajax user interfaces. In ICSE, 2009.

[15] Ali Mesbah, Arie van Deursen, and Stefan Lenselink. Crawling ajax-based web applications through dynamic analysis of user interface state changes. ACM Transactions on the Web, 6(1):1–30, 2012.

[16] A. Metwally, D. Agrawal, and A. El Abbadi. Detectives:Detecting coalition hit inflation attacks in advertising networks streams. In WWW, 2007.

[17] A. Metwally, F. Emekci, D. Agrawal, and A. El Abbadi.Sleuth: Single-publisher attack detection using correlation hunting. In PVLDB, 2008.

[18] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson. What's clicking what? techniques and innovations of today's clickbots. In DIMVA, 2011.

[19] L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajan,I. Obermiller, and S. Shayandeh. Appinsight: mobile app performance monitoring in the wild. In USENIX OSDI, 2012.

[20] W. Yang, M. Prasad, and T. Xie. A grey-box approach for automated gui-model generation of mobile applications. In FASE, 2013.

[21] M. Najork. Web spam detection. In L. Liu and M. T. □Ozsu, editors, Encyclopedia of Database Systems, pages 3520{3523. Springer US, 2009.

[22] Nick Bilton. Disruptions: So Many Apologies, So Much Data Mining. http://bits.blogs.nytimes.com/2012/02/12/disruptions-so-many-apologies-so-much-data-mining, 2012.

IJATES

[23] Peter Gilbert, Byung-Gon Chun, Landon P Cox, and Jaeyeon Jung. Vision: automated security validation of mobile apps at app markets. In Proceedings of the second international workshop on Mobile cloud computing and services - MCS '11, page 21, New York, New York, USA, 2011. ACM Press.

[24] Google admob: What's the difference between estimated and finalized earnings? http://support.google.com/adsense/answer/168408/.

[25] Microsoft advertising: Build your business. http: //advertising.microsoft.com/en-us/splitter.

[26] iad app network. http://developer.apple.com/support/appstore/iad-app-network/.

[27] Admob publisher guidelines and policies. http://support.google.com/admob/answer/1307237?hl=en&ref topic=1307235.

[28] Microsoft pubcenter publisher terms and conditions.http://pubcenter.microsoft.com/StaticHTML/TC/TC en.html.

[29] L. Breiman. Bagging predictors. Machine Learning,24(2):123{140, 1996.

[30] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In European Conference on Computational Learning Theory, pages 23{37, 1995.

[31] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco,CA, USA, 1993.

IJATES