Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing

Riwa Mouawi
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon
ma162@aub.edu.lb

Mariette Awad
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon

Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon

Imad H. El Hajj
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon

Ayman Kayssi
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon

***Abstract*— In recent years, mobile advertising has gained popularity as a mean for publishers to monetize their free applications. One of the main concerns in the in-app advertising industry is the popular attack known as "click fraud", which is the act of clicking on an ad, not because of interest in this ad, but rather as a way to generate illegal revenues for the application publisher. Many studies evaluated click fraud attacks in the literature, and some proposed solutions to detect it. In this paper, we propose a click fraud detection model, hereafter CFC, to classify fraudulent clicks by adopting some features and then testing using KNN, ANN and SVM. In fact, based on our experimental results, the different featured classifiers reached an accuracy higher than 93%.**

Keywords*— *click fraud; in-app ads; KNN, SVM, ANN, mobile advertisement.

I. INTRODUCTION

Mobile advertising has gain popularity in recent years, as a mean for publishers to monetize their free applications, just by adopting few simple integration steps. In fact, researchers predicted a growth of $17 billion by 2018 for mobile advertising [1].

The mobile advertising industry also known as in-app advertising, consists of four main components: 1) The user that views the ad, 2) The advertiser that pays to have his/her ad shown in a set of applications, 3) The publisher (i.e. the application's developer) who is willing to display ads in his/her application in return for a certain revenue, and 4) The ad network that works as a relay between the advertiser and the publisher. There are currently many ad network companies whose main goal is to coordinate with both publishers and advertisers in return of a percentage of the publisher's profit.

There are many charging models adopted in this industry, such as the *cost-per-thousand-impression* model, the *cost-per-click* model, and the *cost-per-action* model. One of the main concerns in the in-app advertising industry is the popular attack known as "click fraud". Click fraud is the act of clicking on an ad, not because of interest in this ad, but rather as a way to generate revenue for the publisher of the application. Thus, it takes advantage of the CPC charging model to generate illegal ad click requests and is Advertiser's loss due to click fraud was estimated to be $1 billion in 2013 [2], and according to a CNBC report [3], "Businesses could lose $16.4 billion to online advertising fraud in 2017".

Many studies evaluated click fraud attacks in the literature, and some proposed solutions to detect it using, for example, random ads to detect if any automated ad clicking tool is used, or cryptographically authenticating users, etc. Because machine learning is being actively thought for network attacks, fraudulent activities [4], spam email [5] we propose in this paper to classify fraudulent clicks by adopting some custom-designed features and by testing using multiple pattern recognition concepts such as KNN, ANN and SVM. In addition, differently from existing literature that adopted decision trees, SVM, and Bayesian theory to classify fraudulent clicks, our work distinguishes itself by introducing a new party trusted by both advertisers and ad network that manages the click fraud detection by crowdsourcing ad requests.

The rest of this paper is organized as follows: we discuss the literature review in section II. In section III, we explain our approach, while the experimental procedure, results and analysis are detailed in section IV. Section V concludes the study with future work.

II. RELATED WORK

In the mobile advertising domain, pattern recognition approaches were used to classify fraudulent acts. Dave et al. [6] used the Bayes theorem to calculate the ratio of non-malicious users following a non-malicious clicking event. This Bayes probability value was used to estimate the click-fraud rate.

Crussell et al. [7] built a decision tree based on the categorical features extracted from their ground truth dataset to classify ad requests versus none ad request. Using Weka, Xu et al. [8] constructed a pruned decision tree to classify traffic as valid, casual or fraudulent. After extracting app-publisher features such as the distance of buttons to ad placements, Vasumati et al. [9] applied a decision tree and a bi-partite graph classifier to classify spam versus non-spam publishers.

Liu et al. [10] built a binary SVM classifier based on their collected dataset to determine if two UI controls (such

978-1-5386-6673-9/18/$31.00 ©2018 IEEE

as a button) most likely lead to the same state (such as a new in-app page). This classification was used by their automated Monkey tool that simulates user interaction in the context of ad clicking.

Vani et al. [11] proposed a node-tag propagation (N-TP) algorithm that returns a list of apps sorted according to their probability of being spam in the context of click fraud. Cho et al. [12] tested the robustness of multiple existing ad networks by sending a new falsified device identifier in each new ad request to simulate each click performed by a new mobile device.

Blizard et al. [13] evaluated a known website ads malware whose main goal is to increase the revenue of the publisher by tricking the user into clicking the ad a second time, for example. Alrwais et al. [14] investigated the well-known "Operational Ghost Click" ad related attack, where attackers controlled ad view session from innocent publishers. They performed a large-scale study of over 7,000 ad requests to detect the presence of this attack. Stone-Gross et al. [15] analyzed ad related traffic collected by an existing ad network and detected multiple ad fraud instances. Pearce et al. [16] analyzed the behavior of a large-scale botnet called ZeroAccess that performs click fraud. Miller et al. [17] studied two families of popular bots that performed click fraud.

Juels et al. [18] proposed a click-fraud detection system that cryptographically authenticates legitimate users based on their past behaviors. Liu et al. [19] proposed a new ad fetching system that securely fetches ad information and signs it with the user's signature. Haddadi et al. [20] proposed a new click spam detection method that creates random ads with random text. They assumed that users clicking on these arbitrary ads are malicious automated tools.

III. PROPOSED SOLUTION

The click fraud detection systems, in the literature, are either managed by ad networks (without collaborating with advertisers) or by advertisers (without collaborating with ad networks). However, these two parties have opposite motives in terms of determining whether an ad click is fraudulent or legitimate. While it's in the interest of the ad network to consider an ad click to be able to charge the advertiser for it, it's in the interest of the advertiser to flag malicious ad request to avoid paying the ad network.

Accordingly, we created a model, hereafter CFC that overcomes this problem by introducing a new party, trusted by both advertisers and ad network. It manages the click fraud detection by crowdsourcing ad requests from multiple combinations of ad networks-publishers-advertisers.

Our proposed system is composed of four components as shown in Fig. 1.

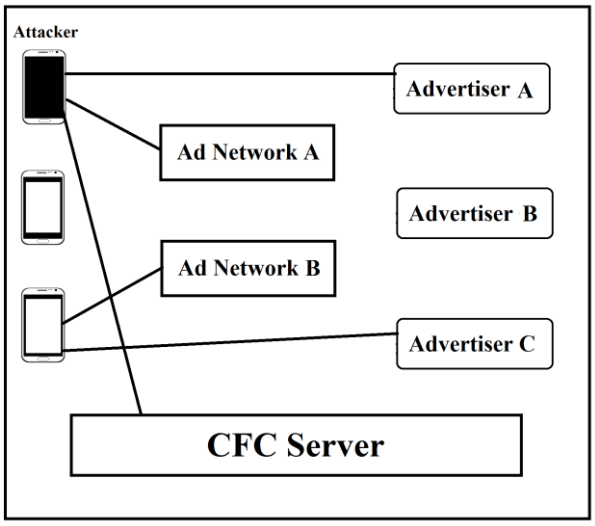

Figure 1: CFC Proposed Solution

A. *Phone Component*

On the client side, the publisher integrates in the app an ad component in the form of a single jar file, which is published by the ad network on its website after merging it with another jar file created by the CFC party in order to handle click fraud detection. This phone component fetches ads, displays them, and manages clicks by displaying the advertised websites on ad clicks and sending clicks information to the CFD party simultaneously.

B. *Ad Network Component:*

This component acts as a relay between different publishers and advertisers by: 1) selecting which ads to send to the publisher for display, 2) charging advertisers for each ad click and 3) paying the publisher a percentage of the charged money.

C. *Advertiser Component:*

After clicking on an ad featured by the phone component, the user will be redirected to the advertiser's website, which represents the advertiser component.

D. *CFC Server Component:*

After collecting a large number of clicks (to be determined by the CFC administrator) from different phone components of ad networks using the CFC service, a Crowdsource based approach is performed. This is because an ad network is able to monitor and assess many clicks from different apps, however, it is not able to monitor the user's activity for click fraud detection once s/he is redirected to the advertising site. Besides while an advertiser is able to monitor the user activity on his/her site (for example the duration spent on the site), it is prone to a high false positive rate, when the fraudulent judgment is only click based. Accordingly, if a user clicks on an ad, and after being redirected to the advertiser's website, s/he shows no interest in this website, s/he will be flagged as malicious for exiting the website after few seconds.

Our proposed solution addresses these two shortcomings by combining both the click information provided by the ad networks and the user activity information provided by the advertiser. In our CFC model, we extract the following features:

1) Percentage of duration-suspicious clicks per publisher

2) Total number of clicks per publisher: This includes suspicious and nonsuspicious ones.

3) Ratio of unique IPs in total number of clicks per publisher: CFC investigates the user IP in each ad request by performing an IP challenge to make sure that the IP is not spoofed. Therefore, this feature is used to detect if multiple ad requests are performed from the same mobile device, which is usually done by malicious automated click tools.

4) Ratio of app downloads per publisher: we considered this feature to take into consideration if a malicious publisher has a high number of registered users that perform fraudulent clicks.

5) Variance of the number of ad clicks in time: we measured the modification in the total number of clicks per publisher from one iteration to another. If the user suddenly presented a higher number of ad clicks compared to previous iterations (time), then this feature value will have a higher number.

IV. Experimental Setup, Results and Analysis

To overcome the difficulty of generating real ad traffic to test our model, we used Matlab to simulate ad requests. We created a total population of 500,000 ad requests, and 1,000 different publishers (10% of which are considered as malicious and generate a higher number of ad requests). To simulate real-life ad traffic, we created multiple time iterations, in which we chose (without replacement) random samples from the total population.

We generated multiple datasets and combined them to have 3,247 instances. We performed data preprocessing by deleting instances with missing key information such as the user IP. To fetch this information in the real world, we are relying on an API that does not provide an IP in some cases. In addition, we normalized the data for each of the 5 features. To simulate real-life traffic, we manually modified 1% of the total instances as outliers by changing their real classification to malicious (when in fact they are nonmalicious) and nonmalicious (when in fact they are malicious).

A. ANN Classification

We performed an ANN classification using Matlab. We tested five different ANN parameters:

- C1/Case1: 3 hidden layers with 50 neurons in each using 'crossentropy' for cost function and 'traingd' = 0.01.
- C2/Case2: 1 hidden layer with 50 neurons in each using 'crossentropy' for cost function and 'traingd' = 0.01.
- C3/Case3: 3 hidden layer with 150 neurons in each using 'crossentropy' for cost function and 'traingd' = 0.01.
- C4/Case4: 3 hidden layer with 50 neurons in each using 'mse' for cost function and 'traingd' = 0.01.
- C5/Case5: 3 hidden layer with 50 neurons in each using 'crossentropy' for cost function and 'traingd' = 0.01.

Fig. 2 displays the confusion matrices (average of all folds) obtained using our ANN classifier for the five different ANN parameters.

Table 1 displays the average accuracy of all folds () as well as theF1 score for the 5 ANN cases. All the five cases presented a high value of F1 score.

B. KNN Classification

We tested 3 different values for neighbors (K = 3, K = 5, K = 9). We considered 30% of the data as training data and

Table 1: ANN Classification Accuracy and F1 score

	C1	C2	C3	C4	C5
Ave. Acc.	93.64	93.11	96.67	89.68	93.84
F1 Score	0.8	0.77	0.9	0.57	0.96

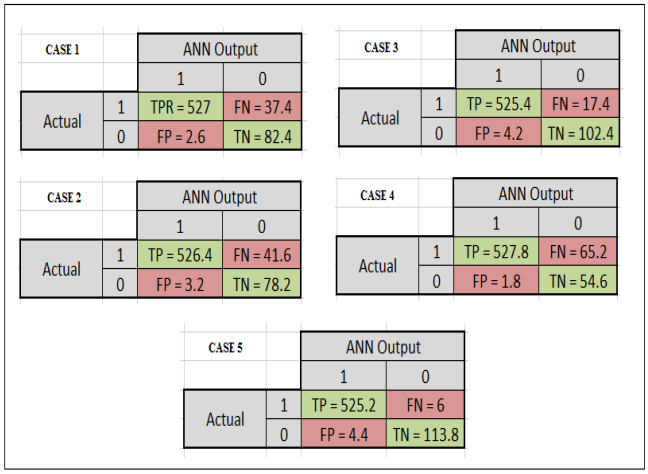

Figure 2: ANN Confusion Matrices

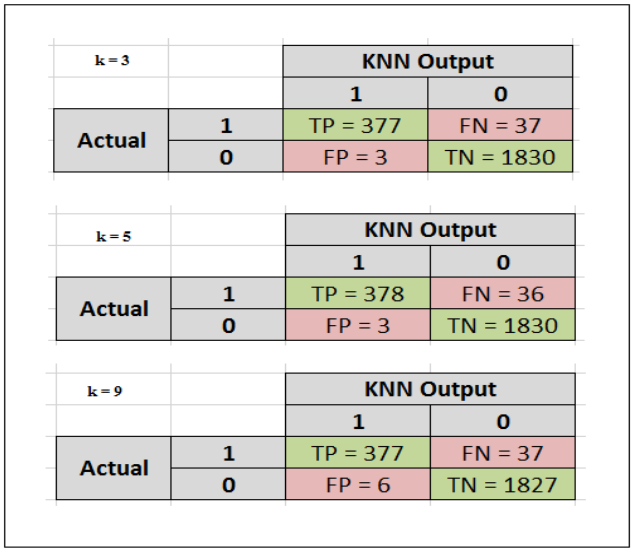

Figure 3: KNN Confusion Matrices

Table 2: KNN Classification Accuracy and F1 Score

	K=3	K=5	K=9
Ave. Acc.	98.22	98.26	98.08
F1 Score	0.99	0.99	0.99

CASE 1		SVM Output	
		1	0
Actual	1	TP = 370	FN = 44
	0	FP = 2	TN = 1831

CASE 3		SVM Output	
		1	0
Actual	1	TP = 393	FN = 21
	0	FP = 36	TN = 1797

CASE 2		SVM Output	
		1	0
Actual	1	TP = 330	FN = 84
	0	FP = 3	TN = 1830

CASE 4		SVM Output	
		1	0
Actual	1	TP = 322	FN = 92
	0	FP = 2	TN = 1831

Figure 4: SVM Confusion Matrices

Table 3: SVM Classification Accuracy and F1 Score

	C1	C2	C3	C4
Ave. Acc.	97.95	96.12	97.46	95.81
F1 Score	0.94	0.88	0.91	0.87

the rest were used as testing data. Fig. 3 displays the confusion matrices obtained using our KNN classifier for the 3 different KNN parameters. The different cases presented very similar results for a high number of TP and TN along with a very low number of FN and FP.

Table 2 displays the accuracy and F1 score for all 3 KNN settings. The accuracy of our KNN classifier is very high (>98%) for all the different cases and all 3 cases presented a high value of F1 score.

C. *SVM Classification*

We tested with four different SVM configurations:

- C1/Case1: kernel RBF
- C2/Case2: kernel RBF with scale = 2 and box constraints = 1
- C3/Case3: kernel RBF with scale = 0.1 and box constraints = 0.5
- C4/Case4: Polynomial function with polynomial order = 3 and box constraints = 0.1).

Fig. 4 displays the confusion matrices obtained using these different SVM classifiers.

Table 3 displays the accuracy for all the 4 SVM parameters as well as the F1 scores. Here also, the accuracy of all SVM classifiers is very high (>95%) for all the different cases. And all 4 cases showed a high value of F1 score.

V. Conclusion

In this paper, we proposed generating ad-related information and extracting features such as the ratio of unique IPs in the total number of clicks per publishers. To classify malicious publishers, we evaluated three different classifiers: KNN, SVM, and ANN. All three classifiers gave very promising results. KNN seems to be the best classifier for our data (almost 98%) followed by SVM (almost 96%) then ANN (almost 93%).

Low values of the FPR (false positive rate) are considered as an important achievement in our click fraud detection system. Future work includes the evaluation of deep learning with malicious publishers.

References

[1] Ash, cloud-iq, "In-App Advertising Growing in Popularity in UK", [Online]. Available: http://blog.cloud-iq.com/blog/in-appadvertising-growing-in-popularity-in-uk [Accessed: May 2016]

[2] dmnews, "Bots Mobilize", December 2015 [Online]. Available: http://www.dmnews.com/mobile-marketing/bots-mobilize/article/291566/ [Accessed: May 2016]

[3] L. Handley, "Businesses could lose $16.4 billion to online advertising fraud in 2017: Report", CNBC, 13 April 2017. Available: https://www.cnbc.com/2017/03/15/businesses-could-lose-164-billion-to-online-advert-fraud-in-2017.html.

[4] M. Awad, R. Khanna, " Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers", ISBN-13: 978-1430259893, 2015

[5] A. Saab*, N. Mitri*, M. Awad, "Ham or Spam? A comparative study for some Content-based Classification Algorithms for Email Filtering" 17th IEEE Mediterranean Electrotechnical Conference - Information & Communication Systems, Beirut, Lebanon, 13-16 April 2014

[6] V. Dave, S. Guha and Y. Zhang, 'Measuring and fingerprinting click-spam in ad networks', in ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, p. 175, 2012, Helsinki, Finland.

[7] J. Crussell, R. Stevens, H. Chen (2014, June). 'MAdFraud: Investigating ad fraud in android applications', in Proceedings of the 12th annual international conference on Mobile systems, applications, and services (pp. 123-134), Bretton Woods, NH, USA, ACM.

[8] H. Xu, D. Liu, A. Koehl, H. Wang, A. Stavrou (2014). 'Click Fraud Detection on the Advertiser Side', in Computer Security-ESORICS 2014 (pp. 419-438), Wroclaw, Poland, Springer International Publishing.

[9] D. Vasumati, M.S. Vani, R. Bhramaramba, O.Y. Babu. (April 2015). 'Data Mining Approach to Filter Click-spam in Mobile Ad Networks', in Int'l Conference on Computer Science, Data Mining & Mechanical Engineering (ICCDMME'2015) April 20-21, 2015, Bangkok.

[10] B. Liu, S. Nath, R. Govindan, J. Liu (2014, April). 'DECAF: detecting and characterizing ad fraud in mobile apps', in Proc. of NSDI, Seattle, WA, USA.

[11] M. Vani, R. Bhramaramba, D. Vasumati, O.Y. Babu. 'A Node-Tag Prediction Algorithm for Touch Spam Detection in Mobile Ad Networks', in International Journal of Computer Engineering and Applications, Volume VIII, Issue III, Part I, December 2014.

[12] G. Cho, J. Cho, Y. Song, H. Kim (2015, August). 'An empirical study of click fraud in mobile advertising networks',10th IEEE International Conference on Availability, Reliability and Security (ARES) (pp. 382-388), Toulouse, France.

[13] T. Blizard,, N. Livic, (2012, October). 'Click-fraud monetizing malware: A survey and case study', in Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on (pp. 67-72), Puerto Rico, USA, IEEE.

[14] S. Alrwais, S. A., A. Gerber, C.W. Dunn, O. Spatscheck, M. Gupta, E. Osterweil (2012, December). 'Dissecting ghost clicks: Ad fraud via misdirected human clicks', in Proceedings of the 28th Annual Computer Security Applications Conference (pp. 21-30), Florida, USA, ACM.

[15] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, G. Vigna (2011, November). 'Understanding fraudulent activities in online ad exchanges', in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (pp. 279-294), Berlin, Germany, ACM.

[16] P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, G.M. Voelker (2014, November). 'Characterizing large-scale click fraud in zeroaccess', in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 141-152), Arizona, USA, ACM.

[17] B. Miller, P. Pearce, C. Grier, C. Kreibich, V. Paxson (2011). 'What's clicking what? techniques and innovations of today's clickbots', in Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 164-183), Amsterdam, The Netherlands, Springer Berlin Heidelberg.

[18] A. Juels, S. Stamm, M. Jakobsson (2007, August). 'Combating Click Fraud via Premium Clicks', in USENIX Security (pp. 1-10), Santa Clara, CA, USA.

[19] W. Li, H. Li, H. Chen, Y. Xia (2015, May). 'AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone', in Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (pp. 75-88), Florence, Italy, ACM.

[20] H. Haddadi (2010). 'Fighting online click-fraud using bluff ads', in ACM SIGCOMM Computer Communication Review, 40(2), 21-25, New Delhi, India.