ADSHERLOCK: EFFICIENT AND DEPLOYABLE CLICK FRAUD DETECTION FOR MOBILE APPLICATIONS

[1]Mulkala.Pradeep, [2]Dr. DINESH KUMAR
[1]mtech, Department of Computer Science, VAAGESWARI COLLEGE OF ENGINEERING Thimmapur, Karimnagar, Telangana, INDIA, H.no:20S41D5827, pradeepmulkala27@gmail.com
[2]Professor,Department of Computer Science, VAAGESWARI COLLEGE OF ENGINEERING Thimmapur, Karimnagar, Telangana, INDIA, sahni.dinesh@live.com

ABSTRACT: Mobile advertising plays a vital role in the mobile app ecosystem. A major threat to the sustainability of this ecosystem is click fraud, i.e., ad clicks performed by malicious code or automatic bot problems. Existing click fraud detection approaches focus on analyzing the ad requests at the server side. However, such approaches may suffer from high false negatives since the detection can be easily circumvented, e.g., when the clicks are behind proxies or globally distributed. In this paper, we present AdSherlock, an efficient and deployable click fraud detection approach at the client side (inside the application) for mobile apps. AdSherlock splits the computation-intensive operations of click request identification into an offline procedure and an online procedure. In the offline procedure, AdSherlock generates both exact patterns and probabilistic patterns based on URL (Uniform Resource Locator) tokenization. These patterns are used in the online procedure for click request identification and further used for click fraud detection together with an ad request tree model. We implement a prototype of AdSherlock and evaluate its performance using real apps. The online detector is injected into the app executable archive through binary instrumentation. Results show that AdSherlock achieves higher click fraud detection accuracy compared with state of the art, with negligible runtime overhead.

Keywords *– Click fraud detection, mobile advertising, ad requests identification.*

1. INTRODUCTION

Mobile advertising plays a vital role in the mobile app ecosystem. A recent report shows that mobile advertising expenditure worldwide is projected to reach $247.4 billion in 2020 [1]. To embed ads in an app, the app developer typically includes ad libraries provided by a third-party mobile ad provider such as AdMob [2]. When a mobile user is using the app, the embedded ad library fetches ad content from the network and displays ads to the user. The most common charging model is PPC (Pay-Per-Click) [3], where the developer and the ad provider get paid from the advertiser when a user clicks on the ad. A major threat to the sustainability of this ecosystem is click fraud [4], i.e., clicks (i.e., touch events on mobile devices) on ads which are usually performed by malicious code programmatically or by automatic bot problems. There are many different click fraud tactics which can typically be characterized into two types: in-app frauds insert malicious code into the app to generate forged ad clicks; bots-driven frauds employ bot programs (e.g., a fraudulent application) to click on advertisements automatically. To quantify the inapp ad fraud in real apps, a recent work MAdFraud [5] conducts a large scale measurement about ad fraud in realworld apps. In a dataset including about 130K Android apps, MAdFraud reports that about 30 percent of apps make ad requests while running in the background.

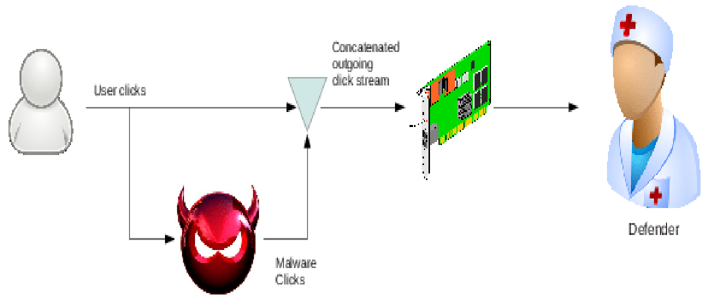


Fig.1: Example figure

Focusing on bots-driven click fraud, another recent work uses an automated click generation tool ClickDroid [4] to empirically evaluate eight popular advertising networks by performing real click fraud attacks on them. Results [4] show that six advertising networks out of eight are vulnerable to these attacks. Aiming at detecting click frauds in mobile apps, a straightforward approach is a threshold-based detection at the server-side. If an ad server is receiving a high number of clicks with the same device identifier (e.g., IP address) in a short period, these clicks can be considered as fraud. This straightforward approach, however, may suffer from high false negatives since the detection can be easily circumvented when the clicks are behind proxies or globally distributed. In the literature, there are also more sophisticated approaches [6], [7] focusing on detecting click frauds at the server-side. The precisions of these server-side approaches, however, are not sufficient enough for the click fraud problem. For example, in a recent mobile ad fraud competition [6], the best three approaches achieve only a precision of 46.15 to 51.55 percent using various machine learning techniques.

2. LITERATURE REVIEW

An empirical study of click fraud in mobile advertising networks:

Smartphone advertisement is increasingly used among many applications and allows developers to obtain revenue through in-app advertising. Our study aims at identifying potential security risks of a type of mobile advertisement where advertisers are charged for their advertisements only when a user clicks (or touches) on the advertisements in their applications. In the Android platform, we design an automated click generation attack and empirically evaluate eight popular advertising networks by performing real attacks on them. Our experimental results show that six advertising networks (75%) out of eight (Millennial Media, App Lovin, Ad Fit, Mdot M, Rev Mob and Cauly Ads) are vulnerable to our attacks. We also discuss how to develop effective defense mechanisms to mitigate such automated click fraud attacks.

MAdFraud: Investigating ad fraud in Android applications:

Many Android applications are distributed for free but are supported by advertisements. Ad libraries embedded in the app fetch content from the ad provider and display it on the app's user interface. The ad provider pays the developer for the ads displayed to the user and ads clicked by the user. A major threat to this ecosystem is ad fraud, where a miscreant's code fetches ads without displaying them to the user or "clicks" on ads automatically. Ad fraud has been extensively studied in the context of web advertising but has gone largely unstudied in the context of mobile advertising. We take the first step to study mobile ad fraud perpetrated by Android apps. We identify two fraudulent ad behaviors in apps: 1) requesting ads while the app is in the background, and 2) clicking on ads without user interaction. Based on these observations, we developed an analysis tool, MAdFraud, which automatically runs many apps simultaneously in emulators to trigger and expose ad fraud. Since the formats of ad impressions and clicks vary widely between different ad providers, we develop a novel approach for automatically identifying ad impressions and clicks in three steps: building HTTP request trees, identifying ad request pages using machine learning, and detecting clicks in HTTP request trees using heuristics. We apply our methodology and tool to two datasets: 1) 130,339 apps crawled from 19 Android markets including Play and many third-party markets, and 2) 35,087 apps that likely contain malware provided by a security company. From analyzing these datasets, we find that about 30% of apps with ads make ad requests while in running in the background. In addition, we find 27 apps which generate clicks without user interaction. We find that the click fraud apps attempt to remain stealthy when fabricating ad traffic by only periodically sending clicks and changing which ad provider is being targeted between installations.

Detecting click fraud in online advertising: A data mining approach:
Click fraud-the deliberate clicking on advertisements with no real interest on the product or service offered-is one of the most daunting problems in online advertising. Building an effective fraud detection method is thus pivotal for online advertising businesses. We organized a Fraud Detection in Mobile Advertising (FDMA) 2012 Competition, opening the opportunity for participants to work on real-world fraud data from BuzzCity Pte. Ltd., a global mobile advertising company based in Singapore. In particular, the task is to identify fraudulent publishers who generate illegitimate clicks, and distinguish them from normal publishers. The competition was held from September 1 to September 30, 2012, attracting 127 teams from more than 15 countries. The mobile advertising data are unique and complex, involving heterogeneous information, noisy patterns with missing values, and highly imbalanced class distribution. The competition results provide a comprehensive study on the usability of data mining-based fraud detection approaches in practical setting. Our principal findings are that features derived from fine-grained time-series analysis are crucial for accurate fraud detection, and that ensemble methods offer promising solutions to highly-imbalanced nonlinear classification tasks with mixed variable types and noisy/missing patterns. The competition data remain available for further studies

On hit inflation techniques and detection in streams of web advertising networks:
Click fraud is jeopardizing the industry of Internet advertising. Internet advertising is crucial for the thriving of the entire Internet, since it allows producers to advertise their products, and hence contributes to the well being of e-commerce. Moreover, advertising supports the intellectual value of the Internet by covering the running expenses of the content publishers' sites. Some publishers are dishonest, and use automation to generate traffic to defraud the advertisers. Similarly, some advertisers automate clicks on the advertisements of their competitors to deplete their competitors ' advertising budgets. In this paper, we describe the advertising network model, and discuss the issue of fraud that is an integral problem in such setting. We propose using online algorithms on aggregate data to accurately and proactively detect automated traffic, preserve surfers' privacy, while not altering the industry model. We provide a complete classification of the hit inflation techniques; and devise stream analysis techniques that detect a variety of fraud attacks. We abstract detecting the fraud attacks of some classes as theoretical stream analysis problems that we bring to the data management research community as open problems. A framework is outlined for deploying the proposed detection algorithms on a generic architecture. We conclude by some successful preliminary findings of our attempt to detect fraud on a real network.

5. Detecting click fraud in pay-per-click streams of online advertising networks
With the rapid growth of the Internet, online advertisement plays a more and more important role in the advertising market. One of the current and widely used revenue models for online advertising involves charging for each click based on the popularity of keywords and the number of competing advertisers. This pay-per-click model leaves room for individuals or rival companies to generate false clicks (i.e., click fraud), which pose serious problems to the development of healthy online advertising market. To detect click fraud, an important issue is to detect duplicate clicks over decaying window models, such as

jumping windows and sliding windows. Decaying window models can be very helpful in defining and determining click fraud. However, although there are available algorithms to detect duplicates, there is still a lack of practical and effective solutions to detect click fraud in pay-per-click streams over decaying window models. In this paper, we address the problem of detecting duplicate clicks in pay-per-click streams over jumping windows and sliding windows, and are the first that propose two innovative algorithms that make only one pass over click streams and require significantly less memory space and operations. GBF algorithm is built on group Bloom filters which can process click streams over jumping windows with small number of sub-windows, while TBF algorithm is based on a new data structure called timing Bloom filter that detects click fraud over sliding windows and jumping windows with large number of sub-windows. Both GBF algorithm and TBF algorithm have zero false negative. Furthermore, both theoretical analysis and experimental results show that our algorithms can achieve low false positive rate when detecting duplicate clicks in pay-per-click streams over jumping windows and sliding windows.

3. METHODOLOGY

Portable promoting assumes an indispensable part in the versatile application biological system. A significant danger to the supportability of this biological system is click misrepresentation, i.e., promotion clicks performed by malevolent code or programmed bot issues. Existing snap extortion recognition approaches center around breaking down the advertisement demands at the worker side. Nonetheless, such methodologies may experience the ill effects of high bogus negatives since the identification can be effortlessly evaded, e.g., when the snaps are behind intermediaries or universally dispersed.

Disadvantages:

1.Existing click fraud detection approaches focus on analyzing the ad requests at the server side.
2. However, such approaches may suffer from high false negatives since the detection can be easily circumvented, e.g., when the clicks are behind proxies or globally distributed.

We present AdSherlock, a productive and deployable snap misrepresentation identification approach at the customer side (inside the application) for versatile applications. AdSherlock parts the calculation serious tasks of snap demand distinguishing proof into a disconnected system and an online method. In the disconnected strategy, AdSherlock produces both careful examples and probabilistic examples dependent on URL (Uniform Resource Locator) tokenization. These examples are utilized in the online system for click demand recognizable proof and further utilized for click misrepresentation identification along with an advertisement demand tree model.

Advantages:

1. AdSherlock generates both exact patterns and probabilistic patterns based on URL (Uniform Resource Locator) tokenization.
2. Finally, AdSherlock instruments the online fraud detector into the app binaries which are then released by the app store.

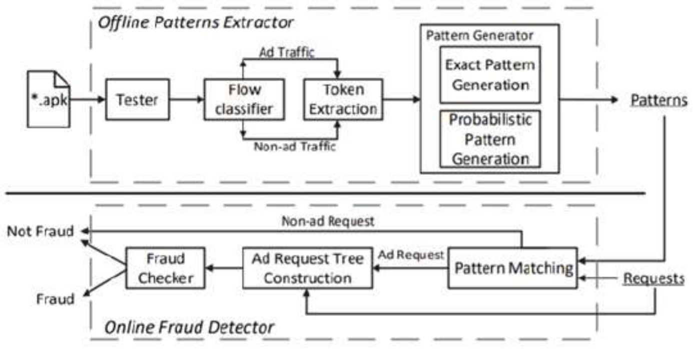


Fig.2: System architecture

Recent years, several works are fighting on ad frauds in mobile advertising. MadFraud [5] studies the in-app fraud by executing apps in Android emulators to observe deviated behavior to detect ad frauds. DECAF [14] analyzes the UI of apps to discover display fraud such as small ads, hidden ads, intrusive ads, etc. However, both of them are investigated in a controlled environment and are hard to detect bots-driven click frauds. Different from them, AdSherlock is deployable in a production environment and performs click fraud detection in an online manner. Another recent work aims at bots-driven click frauds is ClickDroid [15]. It develops an automated click generation tool to simulate attacker and detects frauds by distinguishing human-generated touch events from program-generated touch events. It needs the Android kernel to be modified to filter out program-generated touch events. AdSherlock does not assume any modification on the Android kernel and is a more general approach that proactively targets both inapp click frauds and bots-driven click frauds. There also exists a hardware-assisted solution [16] for fraud detection in mobile advertising. AdAttester [16] provides proofs of unforgeable clicks and verifiable display based on ARM's TruseZone. Different from AdAttester, AdSherlock does not need any hardware support.

4. IMPLEMENTATION

MODULE DESIGN:

1. ADMIN

In this module admin will login into web application. Admin will view users,view owners and view frauds.

2. OWNER

In this module user will register,login,add post and view ads.

3. USER

In this module user will register,login,view ads and profile.

Fraud Checker:

When an ad click is identified, AdSherlock invokes the fraud checker. The checker exploits the information of finegrained user input events to deviate human and nonhuman clicks. To obtain the input events, we investigate the event flow generated both by human clicks and non-human clicks. We further divide the non-human clicks into bots-driven fraudulent clicks and in-app fraudulent clicks according to the fraud tactics.

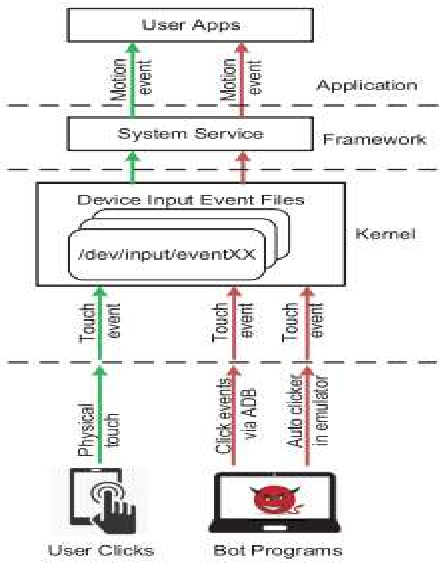


Fig.3: The event flow of input events generated by human and nonhuman clicks.

Since in-app fraudulent clicks do not generate any motion events and are easy to be identified, we focus on bots-driven fraudulent clicks. When the user clicks on the screen, touch events are generated and delivered to apps as motion events. However, such input events can be faked by bot programs. As illustrated in Fig , green lines show the event flow of human clicks while red lines show that of faked clicks. Bot programs fake human clicks by programmatically generating touch events in two ways: emulator-based fraudulence and ADB-based fraudulence. In emulator-based fraudulence, a bot program can automatically click on apps in emulators via auto clickers on PC. In ADB-based fraudulence, bot programs can send sendevent commands to the Android app via Android Debug Bridge (ADB). Since the ADB-based fraudulence requires root permission and is hard to perform in practice, AdSherlock focuses on detecting emulator-based fraudulence which is a widely used tactic. At the application level, both human clicks and fraudulent clicks are transferred to the app as motion events. We observe that these motion events contain information that can help us deviate human clicks and non-human clicks inside an app. In this section, we carefully analyze the useful properties contained in the motion event. We then indicate how to distinguish human clicks and non-human clicks using these properties.

5. EXPERIMENTAL RESULTS

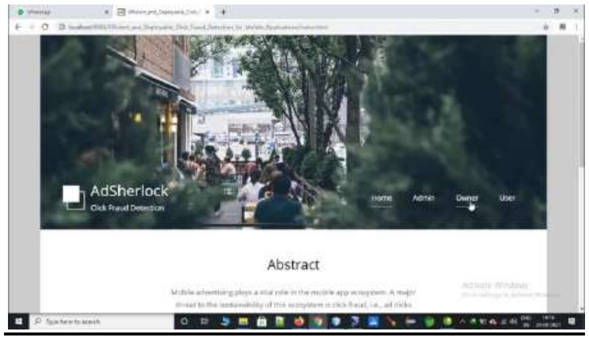


Fig.4: Home screen

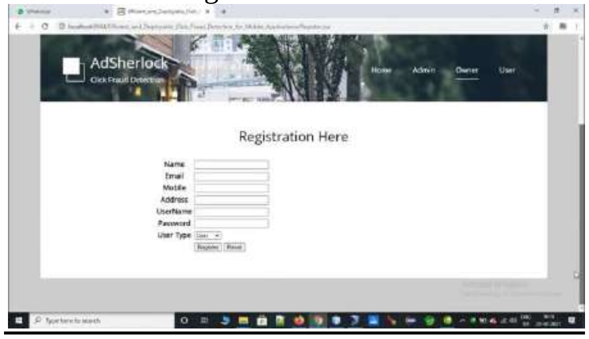


Fig.5: Regsitration

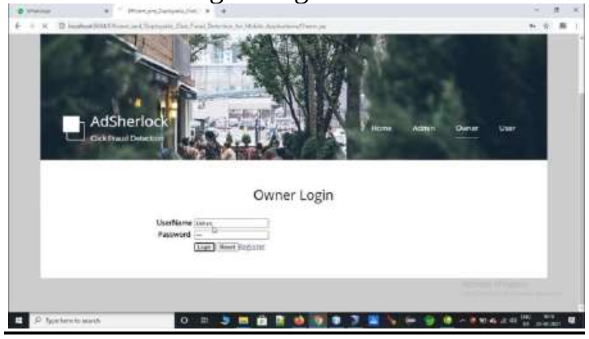


Fig.6: owner login

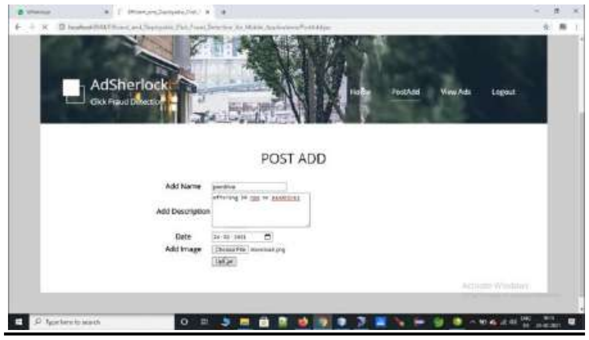


Fig.7: Post add

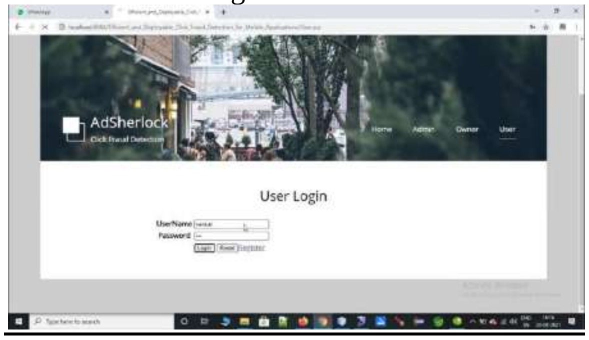
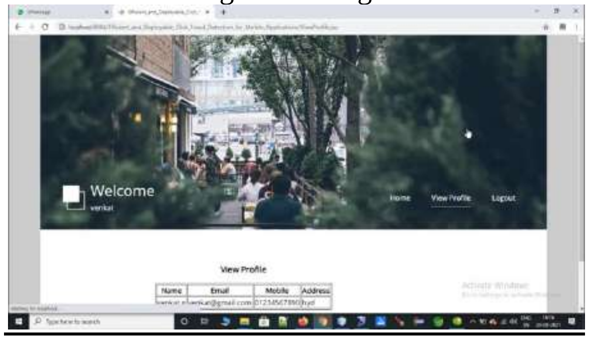


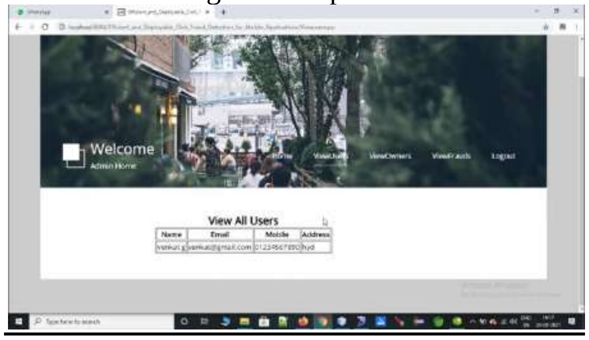
Fig.8: User login



View Profile

Name	Email	Mobile	Address
venkat g	venkat@gmail.com	01234567890	hyd

Fig.9: View profile



View All Users

Name	Email	Mobile	Address
venkat g	venkat@gmail.com	01234567890	hyd

Fig.10: View all users

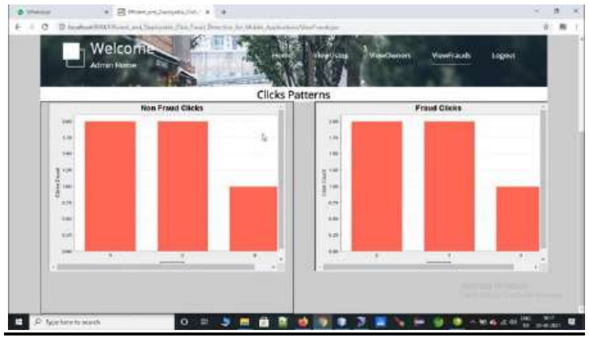


Fig.11: Graph

6. CONCLUSION

AdSherlock is an efficient and deployable click fraud detection approach for mobile apps at the client side. As a client-side approach, AdSherlock is orthogonal to existing server-side approaches. It splits the computation intensive operations of click request identification into an offline process and an online process. In the offline process, AdSherlock generates both exact patterns and probabilistic patterns based on url tokenization. These patterns are used in the online process for click request identification, and further used for click fraud detection together with an ad request tree model. Evaluation shows that AdSherlock achieves high click fraud detection accuracy with a negligible runtime overhead.

Future work

In the future, we plan to combine static analysis with the traffic analysis to improve the accuracy of ad request identification and explore attacks designed to evade AdSherlock.

REFERENCES

[1]. "Mobile advertising spending worldwide." [Online]. Available: https://www.statista.com/statistics/280640/mobile-advertisingspending-worldwide/
[2]. 2."Google admob." [Online]. Available: https://apps.admob.com/
[3]. M. Mahdian and K. Tomak, "Pay-per-action model for online advertising," in Proc. of ACM ADKDD, 2007.
[4]. 4.G. Cho, J. Cho, Y. Song, and H. Kim, "An empirical study of click fraud in mobile advertising networks," in Proc. of ACM ARES, 2015.
[5]. J. Crussell, R. Stevens, and H. Chen, "Madfraud: Investigating ad fraud in android applications," in Proc. of ACM MobySys, 2014.
[6]. 6.R. Oentaryo, E.-P. Lim, M. Finegold, D. Lo, F. Zhu, C. Phua, E.-Y. Cheu, G.-E. Yap, K. Sim, M. N. Nguyen, K. Perera, B. Neupane, M. Faisal, Z. Aung, W. L. Woon, W. Chen, D. Patel, and D. Berrar, "Detecting click fraud in online advertising: A data mining approach," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 99–140, 2014.
[7]. 7.B. Kitts, Y. J. Zhang, G. Wu, W. Brandi, J. Beasley, K. Morrill, J. Ettedgui, S. Siddhartha, H. Yuan, F. Gao, P. Azo, and R. Mahato, Click Fraud Detection: Adversarial Pattern Recognition over 5 Years at Microsoft. Cham: Springer International Publishing, 2015, pp. 181–201.
[8]. A. Metwally, D. Agrawal, and A. El Abbadi, "Detectives: detecting coalition hit inflation attacks in advertising networks streams," in Proc. of ACM WWW, 2007.
[9]. A. Metwally, D. Agrawal, A. El Abbad, and Q. Zheng, "On hit inflation techniques and detection in streams of web advertising networks," in Proc. of IEEE ICDCS, 2007.
[10]. F. Yu, Y. Xie, and Q. Ke, "Sbotminer: large scale search bot detection," in Proc. of ACM WSDM, 2010.

[11]. L. Zhang and Y. Guan, "Detecting click fraud in pay-per-click streams of online advertising networks," in Proc. of IEEE ICDCS, 2008.
[12]. A. Metwally, D. Agrawal, and A. El Abbadi, "Duplicate detection in click streams," in Proc. of ACM WWW, 2005.
[13]. M. S. Iqbal, M. Zulkernine, F. Jaafar, and Y. Gu, "Fcfraud: Fighting click-fraud from the user side," in Proc. of IEEE HASE, 2016.
[14]. B. Liu, S. Nath, R. Govindan, and J. Liu, "Decaf: detecting and characterizing ad fraud in mobile apps," in Proc. of USENIX NSDI, 2014.
[15]. G. Cho, J. Cho, Y. Song, D. Choi, and H. Kim, "Combating online fraud attacks in mobile-based advertising," EURASIP Journal on Information Security, vol. 2016, no. 1, p. 1, 2016.