International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Ad Sherlock Efficient Deployable Click Fraud Detection for Mobile Applications

[1]Bollenkula Sai Praveen, [2]Dr. Dinesh Kumar

[1,2] Vaageswari College of Engineering, Beside LMD Police Station, Karimnagar – 505 527, India

A B S T R A C T

Without mobile advertising, there would be no mobile app ecosystem. It's clear that click fraud, where ads are clicked on by malicious code or bots, poses a serious threat to the sustainability of this ecosystem. Click fraud can now be detected by server-side analysis of advertising requests. Due to the simplicity with which the detection can be avoided, for example when the clicks are disguised behind proxies or are geographically separated, such methods may produce a large number of false negatives. In this paper, we provide AdSherlock, an efficient and deployable client-side (within-app) solution to click fraud detection in mobile apps. AdSherlock divides the computationally intensive phases of recognising click requests into an offline and an online procedure. AdSherlock uses a probabilistic pattern-creation approach based on URL (Uniform Resource Locator) tokenization that operates in an offline mode. These patterns, in conjunction with an ad request tree model, are used to identify click requests in real time, thereby detecting click fraud. AdSherlock was put through its paces by creating a prototype and testing it with real-world applications. The online detector is built into the executable bundle of the programme through binary instrumentation. When compared to other methods for detecting click fraud, AdSherlock performs better while practically never affecting system performance.

KEYWOEDS: Click fraud detection, mobile advertising, adrequests identification.

1. INTRODUCTION

A mobile app ecosystem would not exist without mobile advertising. It has been estimated that by 2020, the global market for mobile advertising would be worth $247.4 billion.

[1]. Third-party mobile ad providers like AdMob

[2] provide ad libraries that app developers incorporate into their apps in order to integrate advertisements. The embedded ad library retrieves ad content from the network and presents it to the user when the user is on a mobile device running the app. PPC (Pay-Per-Click)

[3] is the most popular style of monetization, in which the developer and the ad supplier are paid by the advertiser when a user clicks on the ad. Click fraud.

[4] is a significant challenge for the long-term health of this ecosystem since it involves fraudulent clicks (or touch events on mobile devices) on advertisements. These clicks are typically generated by malicious code or automated bots. Generally speaking, the various click fraud techniques can be broken down into two categories: in-app frauds, which involve inserting malicious code into the app to generate forged ad clicks, and bots-driven frauds, which involve using bot programmes (such as a fraudulent app) to automatically click on advertisements. Recent work MAdFraud

[5] conducts a large-scale measurement of ad fraud in real-world applications, allowing for a quantification of inapp ad fraud. Ad requests are made by around 30% of apps in a sample of about 130K Android apps, according to MAdFraud.

Another recent piece of study examines bot-driven click fraud by employing the automated click generation programme ClickDroid [4] to conduct real-world click fraud attacks against eight of the most popular ad networks. Based on the data [4], it seems that six of the eight ad networks are susceptible to these kinds of attacks. An easy method for spotting click fraud in mobile apps is to use a server-side detection method based on a predetermined threshold. Clicks from the same device identifier (for example, IP address) on an ad server within a short time frame may be suspicious and blocked. However, when clicks are hidden behind proxies or geographically dispersed, this simplistic strategy may produce a large number of false negatives.

2. OVERVIEW OF

AdSherlock is designed to be used by app stores. Before an app is released for download, the app store can use AdSherlock to analyze the app and instrument the online fraud detector into the app for click fraud detection at runtime. Only app binaries (e.g., APKs(Android application package)) are needed, and AdSherlock does not assume any developer input.

AdSherlock is mainly composed of two components: *offline pattern extractor* and *online fraud detector*. First, the offline pattern extractor takes the app as input and automatically executes the app to collect network traffics. Then, it classifies the traffics and extract traffic patterns for ad and non-ad traffics. Next, the online fraud detector is generated based on the extracted traffic patterns. The online fraud detector is responsible for network traffic monitoring, ad request identification, and click fraud detection. Finally, AdSherlock instruments the online fraud detector into the app binaries which are then released by the app store.

We show the main building blocks of AdSherlock in Fig 1. Each app is fed to the offline pattern extractor after uploading to the app store. This extractor automatically executes the app and generates traffic patterns for ad and non-ad traffics. These patterns are injected into the application together with the online fraud detector. When the app is running on the end user's device, the online fraud detector quickly checks every HTTP request with the offline generated patterns and identify the ad request. Next, the click request can be identified quickly

by building an ad request tree. The abnormal click requests are detected by examining the related user input events. This operation is efficient. We mark a click request fraudulent if it does not accompany with any real user input events. For ease of understanding, we further introduce these two components

in details. Offline Pattern Extractor The offline pattern extractor

automatically generates traffic patterns of ad and non-ad traffics for each app. To capture network traffics, we employ an automation tool to execute each mobile app binary automatically. The automation tool we build is called *Tester* using an Android testing tool, *monkeyrunner* [17]. Tester launches multiple instances of the Android emulator concurrently. Due to the scalability concerns, we do not drive the app along complete and exhaustive execution paths, which is time-consuming and not scalable. Instead, we allow incomplete app execution to ensure the scalability. For each app, it is executed in one emulator instance twice lasting 10 minutes, and the Tester automatically navigates through

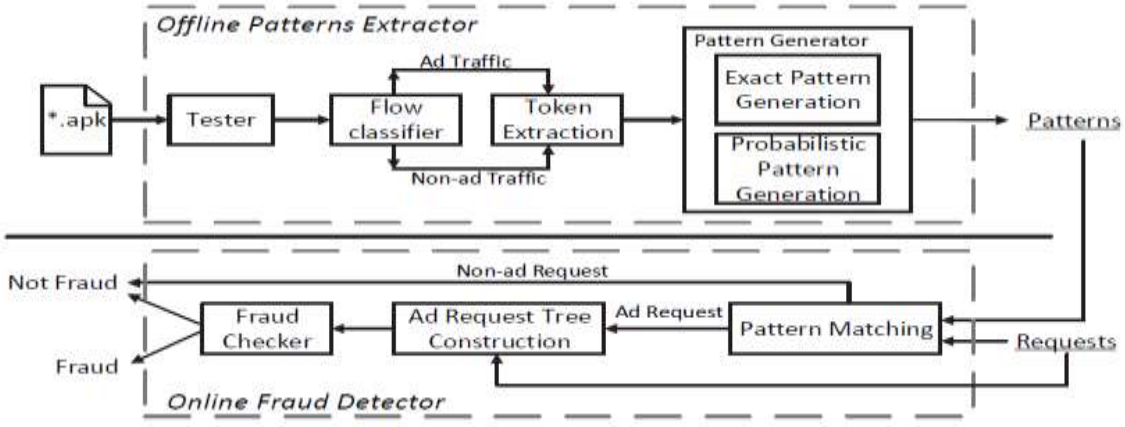

Fig. 1: Overview of AdSherlock.

3. PROPOSED WORK

ONLINE FRAUD DETECTION

A. Click Identification

We build ad request trees to identify click requests. For example, a browser loading a web page may fetch many other static resources, such as CSS, JavaScript or images, to embed in the HTML. In this case, a request tree, whose root is the request to the HTML page, can be built using the HTTP

referer header.

The ad request tree construction starts when an ad request is identified. Each node represents a request and its corresponding response. For each request comes after the ad request, we

add its corresponding node to the ad request tree if:

- The referer field of the latter request is set to be the request in the ad request tree. We consider the latter one as a child of the related request and mark the edge as "REFERER".

- The location header in the response of the request in the tree is set along with a redirection status code to redirect the client to another URL. We consider the original request URL as the parent of the redirected URL and mark the edge as “LOCATION”.
- The latter request URL is in the response body of the ad request. We consider the latter request as a child of the ad request and mark the edge as “BODYURL”. When a user clicks on an ad, the application generates an HTTP request to the ad network. The ad network then redirects the user to the advertiser’s page. The address of the advertiser’s page is typically provided in the location header of the response. Therefore, in our ad request trees, we mark a node as an ad click if its edge link to the child is marked as “LOCATION” and the child node represents a third-party website

B. Fraud Checker

When an ad click is identified, AdSherlock invokes the fraudchecker. The checker exploits the information of fine-grained user input events to deviate human and non-human 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. 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

4. IMPLIEMENTATION

Host Server

The Web Server must enter a valid login name and password to access this section. The user has access to features like View AppDevelopers and Authorize after a successful login. Look At Users And Give Permissions, View Symmetric Key Requests, Click Frauds, All Applications' Positive Behaviour, All Applications' Negative Behaviour, Add Review Filters, View All Uploaded Apps with Rank and Ratings Details, View All Apps with Review, Co Review and Recommend Details, and more. Check out the popularity and download stats for your app in one convenient place.

Creater of Apps

Embed App

The administrator will be able to add the apps through this section. To register a new app, the administrator must first provide the app's name, description, mobile platform, users, file name, and picture files. Every last detail will be filed away in the database.

Check out the Form

In this section, when the administrator selects view application, the app's title, description, mobile platform, users, file name, and screenshots will be shown.

A Prioritized List of Fraud Information

In this section, the ranking fraud count, user name, mobile type, application name, application ID, date, and time are presented when the admin clicks on ranking fraud information.

Suspect documents for fraud investigations

When the module's administrator selects a piece of evidence for fraud information, the user's name, mobile type, application name, application ID, fraud IP address, fraud system name, and the date and time will be presented.

A total of n users are currently logged into this module. Registration is required for certain actions. After his registration has been accepted, he will be able to access the system with his unique user ID and password. After successfully logging in, he will be able to perform actions such as viewing his profile and searching for apps, requesting a symmetric key, viewing the top K apps, and viewing user-recommended apps.

Find and get app(s) for your mobile device

When a user accesses this section of the site, he or she can look for a specific kind of mobile app, click on it to access a search bar, then fill in the program's name, photos, details, ID, and key before finally downloading the app.

and relay the user's response.

Explore the best K software

The user enters the name of the app they're looking for and then chooses from among the top N results to see information about the app's features, users, ratings, and more.

5. CONCLUSION

Ad Sherlock is a client-side solution to detecting click fraud in mobile apps that is both effective and deployable. AdSherlock, being client-side, is orthogonal to the prevalent server-side methods. It does this by decomposing the online component of click request identification-which requires a lot of computation-into a separate offline procedure. AdSherlock uses a probabilistic model based on url tokenization to create accurate patterns in the offline process in addition to the more traditional, deterministic patterns. Together with an ad request tree model, these patterns are utilised to identify click requests in the online process, hence detecting click fraud. According to the results, AdSherlock is able to detect click fraud with a high degree of accuracy and almost no performance impact. We hope to increase ad request identification accuracy and investigate AdSherlock-avoiding assaults by combining static analysis with traffic analysis in the future.

REFERENCES

[1] "Mobile advertising spending worldwide." [Online]. Available:https://www.statista.com/statistics/280640/mobile-advertisingspending-worldwide/

[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] 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] 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] 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.