Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

RESOURCEFUL AND EMPLOYABLE CLICK FRAUD IDENTIFICATION FOR HANDHELD APPLICATIONS

Mrs.B.Rajitha, *Assistant Professor CSE, Vaagdevi College of Engineering(Autonomous), India*
G.Raju, *UG Student,CSE, Vaagdevi College of Engineering(Autonomous), India*
P.Saikeerthi, *UG Student,CSE, Vaagdevi College of Engineering(Autonomous), India*
K.Chandra Shekar, *UG Student,CSE, Vaagdevi College of Engineering(Autonomous), India*
B.Karthik, *UG Student,CSE, Vaagdevi College of Engineering(Autonomous), India*

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

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% of apps make ad requests while running in the background. 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 serverside. 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% using various machine learning techniques. Given the insufficient precision of server-side approaches, a natural question comes up: how about client-side approaches? In fact, compared with the server-side approaches, it is easier to tell whether there is an actual user input at the client side. However, the attacker of the click fraud could be the app developers themselves, since the developers will get paid for those fraudulent ad clicks[20]. Due to this conflict-of-interest problem, we cannot assume the existence of coordination from developers in designing a client-side approach for click fraud detection, e.g., a click fraud detection SDK. Therefore, in this paper, we focus on designing a client-side approach to detect click frauds in mobile apps, without coordination from developers.

There are two major challenges in designing such a system. First [21],[22],[23] for a mobile client, its resources are constrained in terms of computation, memory, and energy. Therefore, the proposed approach must perform the complete fraud detection process. See efficiently, without causing significant overhead. This means that we need to design new algorithms to detect click frauds since existing machine-learning algorithms used by server-side approaches are not suitable for the client side. Second, the click fraud detection should be able to execute under practical user scenarios, instead of a controlled environment dedicated to fraud detection. In MAdFraud [5], a controlled environment (i.e., only one app is running and the HTTP requests are collected for offline analysis) is used to measure the ad fraud behavior of a vast number of apps. However, in our case, the click fraud detection should happen inside the mobile client without outside support, i.e., be deployable in real-world scenarios.

In this paper, we propose AdSherlock, an efficient and deployable click fraud detection approach for mobile apps at the client side. Note that as a client-side approach, AdSherlock is orthogonal to existing server-side approaches. AdSherlock is designed to be used by app stores to ensure a healthy mobile app ecosystem. [24],[25] AdSherlock's high accuracy helps market operators to fight both in-app frauds and bots-driven frauds. Note that, AdSherlock can also be used by any third parties to detect in-app frauds. For example, ad providers can employ AdSherlock to check whether apps embedding their libraries have in-app fraudulent behaviors. To achieve these goals, AdSherlock relies on an accurate offline pattern extractor and a lightweight online fraud detector. AdSherlock works in two stages. At the first stage, the offline pattern extractor automatically executes each app and generates a set of traffic patterns for efficient ad request identification, i.e., extracts common token patterns across different ad requests.

Specifically, after tokenization of the network requests, AdSherlock generates both exact patterns and probabilistic patterns for robust matching. Using the offline pattern extractor, AdSherlock can perform the computation and I/O intensive pattern generation operations in an offline manner, without degrading the online fraud detection operations. At the second stage, the online fraud detector as well as the generated patterns are instrumented into the app and run with the app in actual user scenarios. Inside the app,[8] AdSherlock uses an ad request tree model to identify click requests accurately and efficiently. Since the online fraud detector runs inside the app, it can obtain the fine-grained user input events which are further employed for click fraud detection.

We implement AdSherlock and evaluate its performance using real apps. Results show that AdSherlock achieves higher click fraud detection accuracy compared with state of the art, with negligible runtime overhead.

The contributions of this paper are summarized as follows:

We present the design and implementation of AdSherlock[25], the first system which can achieve efficient and deployable click fraud detection at the client side. We propose a pattern generation mechanism that generates patterns for ad requests and non-ad requests with high accuracy. We also propose an efficient method for online click fraud detection based on an ad request tree model. We

implement AdSherlock and compare its performance with the state-of-art approach. Results show that Ad- Sherlock achieves higher detection accuracy with lower overhead.

2. PROBLEM STATEMENT

since existing machine-learning algorithms used by server-side approaches are not suitable for the client side. Second,[12]-[17] the click fraud detection should be able to execute under practical user scenarios, instead of a controlled environment dedicated to fraud detection. In MAdFraud [5], a controlled environment (i.e., only one app is running and the HTTP requests are collected for offline analysis) is used to measure the ad fraud behavior of a vast number of apps. However, in our case, the click fraud detection should happen inside the mobile client without outside support, i.e., be deployable in real-world scenarios. In this paper, we propose AdSherlock, an efficient and deployable click fraud detection approach for mobile apps at the client side. Note that as a client-side approach, AdSherlock [9] is orthogonal to existing server-side approaches. AdSherlock is designed to be used by app stores to ensure a healthy mobile app ecosystem. AdSherlock's high accuracy helps market operators to fight both in-app frauds and bots-driven frauds. Note that, AdSherlock can also be used by any third parties to detect in-app frauds. For example, ad providers can employ AdSherlock to check whether apps embedding their libraries have in-app fraudulent behaviors..

3. ADSHERLOCK

we propose two pattern classes: exact patterns and probabilistic patterns. Both of them are built from invariant substrings in the HTTP [10] header. We refer to these substrings as tokens. Exact patterns consist of a set of sequential tokens and match an HTTP request if and only if the request contains all tokens in the set with the same ordering. Probabilistic patterns consist of a set of tokens, each of which is associated with an ad score, and a non-ad score[18],[19]. We describe the details of pattern generation in the following sections.

4. SYSTEM ARCHITECTURE

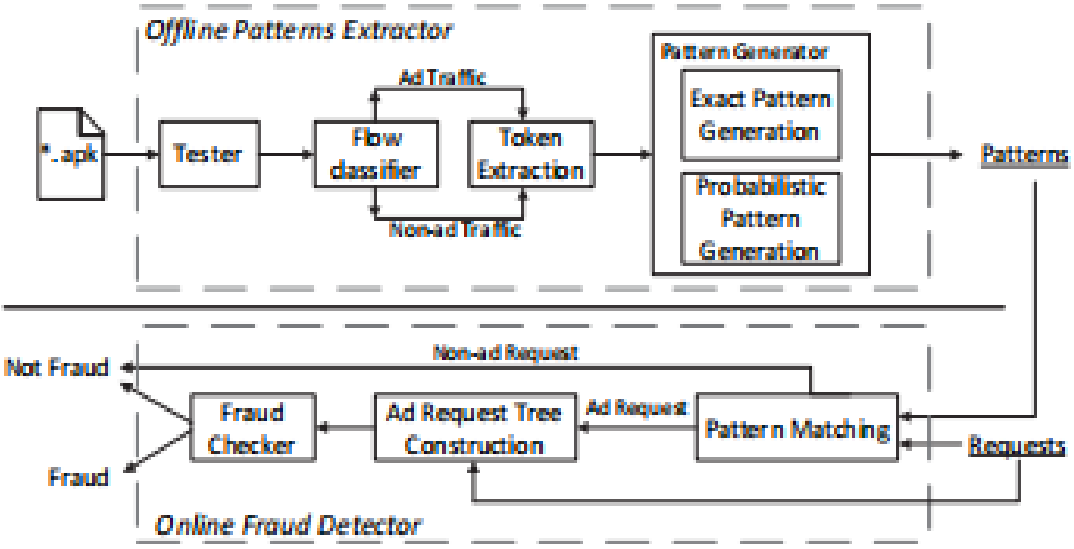

Fig. 1: Overview of AdSherlock.

Fig 7.1 Home page

EXCEPTED OUTCOMES

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

Fig 7.2 User Login page

Fig 7.3 Owner Home page

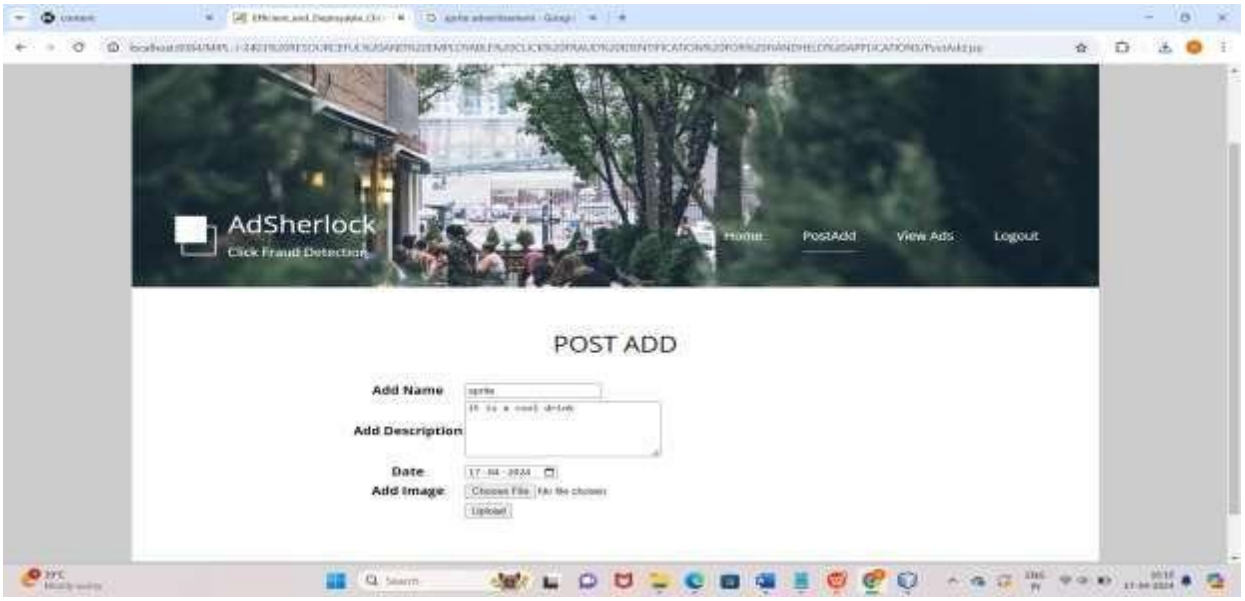

Fig 7.4 Post Adds by owner

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

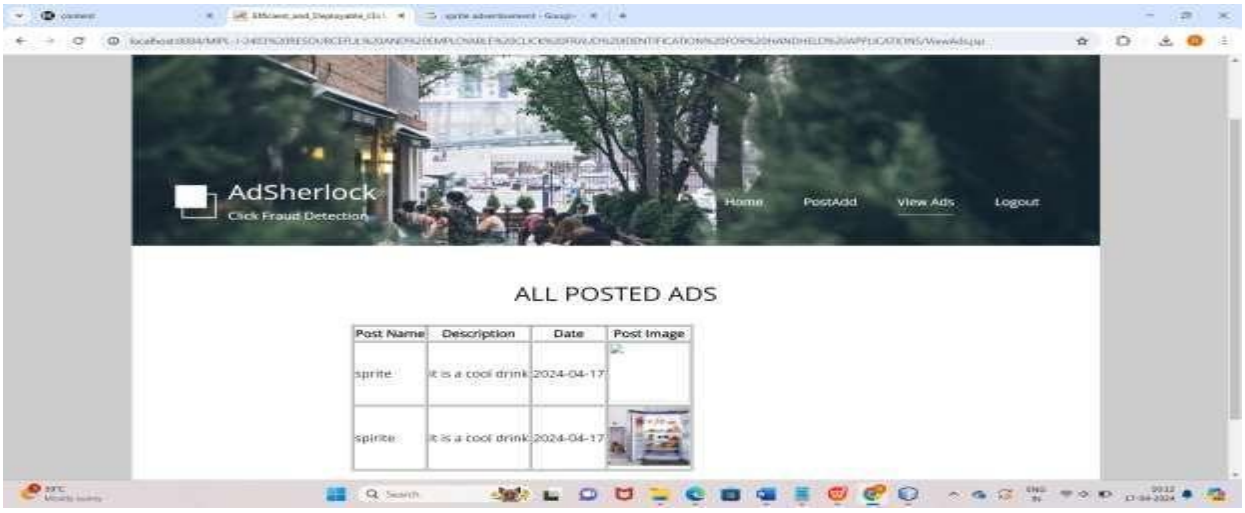

Fig 7.5 Post Adds viewed by admin

Fig 7.6 Adds View by User

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

Fig 7.7 Customer login page

Fig 7.8 Home page of customer

5. 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. 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.

FUTURE SCOPE

The future scope for a resourceful and employable click fraud identification system for handheld applications is promising. It entails leveraging advanced algorithms, real-time detection capabilities, and seamless integration with ad networks to provide advertisers with user-friendly interfaces and actionable insights. Cross-platform compatibility, customization options, and robust data privacy measures are crucial for scalability and global reach. By addressing these aspects, such a system can effectively combat click fraud in the evolving landscape of mobile advertising, safeguarding advertisers' investments and enhancing the integrity of digital advertising ecosystems.

6. 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.
[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.
[16] W. Li, H. Li, H. Chen, and Y. Xia, "Adattester: Secure online mobile advertisement attestation using trustzone," in Proc. of ACM MobySys, 2015.
[17] "Monkeyrunner." [Online]. Available: http://developer.android.com/studio/test/monkeyrunner/index.html
[18] "Zedge." [Online]. Available: https://play.google.com/store/apps/
[19] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "Networkprofiler: Towards automatic fingerprinting of android apps," in Proc. of IEEE INFOCOM, 2013.
[20] "Mopub." [Online]. Available: https://www.mopub.com/
[21] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in Proc. of IEEE S&P, 2005.
[22] "Android montionevent." [Online]. Available: https://developer.android.com/reference/android/view/MotionEvent.html
[23] X. Jin, P. Huang, T. Xu, and Y. Zhou, "Nchecker: saving mobile app developers from network disruptions," in Proc. of ACM EuroSys, 2016.
[24] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in Proc. of ACM ICML, 2006.
[25] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," PloS one, vol. 10, no. 3, p. e0118432, 2015.