International Journal of Advanced Smart Convergence Vol.11 No.3 23-27 (2022)
http://dx.doi.org/10.7236/IJASC.2022.11.3.23

IJASC 22-3-4

Investigation of the SPRT-Based Android Evasive Malware

Jun-Won Ho

Professor, Department of Information Security, Seoul Women's University, Korea
jwho@swu.ac.kr

Abstract

In this paper, we explore a new type of Android evasive malware based on the Sequential Probability Ratio Test (SPRT) that does not perform malicious task when it discerns that dynamic analyzer is input generator. More specifically, a new type of Android evasive malware leverages the intuition that dynamic analyzer provides as many inputs within a certain amount of time as possible to Android apps to be tested, while human users generally provide necessary inputs to Android apps to be used. Under this intuition, it harnesses the SPRT to discern whether dynamic analyzer runs in Android system or not in such a way that the number of inputs per time slot exceeding a preset threshold is regarded as evidence that inputs are provided by dynamic analyzer, expediting the SPRT to decide that dynamic analyzer operates in Android system and evasive malware does not carry out malicious task.

Keywords: *Dynamic Analysis, Sequential Probability Ratio Test (SPRT), Android*

1. Introduction

Dynamic analysis has been used for detecting Android malware apps that are intricately obfuscated and thus can avoid being detected by static analysis. By providing diverse inputs to Android benign and malign apps to be tested and logging the outputs generated as responses to these inputs in separate secure environments, dynamic analyzer can discern malicious activities of Android malign apps through examining log information. To deal with the possible attack with evasive malware against dynamic analyzer in Android system, we look into a new type of Android evasive malware, which is rooted on the Sequential Probability Ratio Test (SPRT) devised by Wald [6]. By leveraging the intuition that dynamic analyzer needs to test as diverse user interfaces and functionalities of many Android apps as possible within a limited time in Android system and thus Android apps receive likely more number of inputs from dynamic analyzer than human users in a time slot, the SPRT-based Android evasive malware decides whether inputs are generated and provided by dynamic analyzer or not and it does not perform malicious task if it determines that input generator is dynamic analyzer.

Manuscript Received: June. 17, 2022 / Revised: June. 20, 2022 / Accepted: June. 23, 2022
Corresponding Author: jwho@swu.ac.kr
Tel: +82-2-970-5607, Fax: +82-2-970-5981
Professor, Department of Information Security, Seoul Women's University, South Korea

2. Related Work

In [4], dynamic analyzer based on Trust-Zone is proposed in Android system. [5] scrutinizes dynamic analysis on Android application framework. Several researchers devise various schemes that generate inputs and provide them to Android applications [1], [2], [7]. Android system provides a Monkey tool through which Android apps are automatically tested [3]. Our prior work explores the interactions among input-driven evasive malware, dynamic analyzer, and human user in the IoT through game theoretic analysis [8]. In [9], the SPRT with evidence of user input preference is utilized for intelligent malicious apps to determine whether to initiate malicious activities. In [10], the SPRT takes it as evidence the number of events that are highly likely selected by dynamic analysis algorithm out of randomly chosen events and decides whether malicious app detection application based on dynamic analysis algorithm can be evaded by malicious apps in user smartphone or not.

Moreover, there are diverse related work in the field of evasive malware [11-24].

3. SPRT-Based Android Evasive Malware

We define dynamic analyzer as an entity providing various inputs to Android benign and malign applications and performing dynamic analysis of them to detect Android malign applications in Android system. We also define Android evasive malware as Android malign application that does not discharge malicious task under the condition that dynamic analyzer runs and provides inputs to it in Android system, evading the detection by dynamic analyzer. Moreover, we assume that Android evasive malware runs the Sequential Probability Ratio Test (SPRT) [6] to decide whether to conduct malign task.

Android evasive malware runs the SPRT as follows: It first defines two hypotheses as follows: H_0 is a null hypothesis that inputs are provided by human user and thus discharges malign task. H_1 is an alternate hypothesis that inputs are provided by dynamic analyzer and thus discharges benign task.
Let $IN_q (q \geq 1)$ denote the number of inputs provided to Android evasive malware in the *qth* time slot.

We have a Bernoulli random variable IR_q as follows:

$$IR_q = 0 \quad \text{if} \quad IN_q < IN^+$$

$$IR_q = 1 \quad \text{if} \quad IN_q \geq IN^+$$

The success probability SP of the Bernoulli distribution is specified by

$$SP = \Pr(IR_q = 1) = 1 - \Pr(IR_q = 0)$$

Note that IN^+ is a pre-defined threshold that is used as a basis in configuring the values of Bernoulli random variable IR_q. Furthermore, SP_0 and SP_1 such that $SP_0 < SP_1$ are pre-defined. By associating SP_0 and SP_1 with H_0 and H_1, the SPRT discontinues with likely adopting H_0 (resp. H_1) when $SP \leq SP_0$ (resp. $SP \geq SP_1$) holds. The SPRT is described as LR_q, which is the log-probability ratio on *q* samples such that each sample is represented as IR_q. LR_q is computed as follows.

$$LR_q = \ln \frac{\Pr(IR_1, \ldots, IR_q \mid H_1)}{\Pr(IR_1, \ldots, IR_q \mid H_0)} \tag{1}$$

Under the assumption that IR_q is independent and identically distributed, LR_q can be expressed as:

$$LR_q = \ln \frac{\prod_{a=1}^{q} \Pr(IR_a \mid H_1)}{\prod_{a=1}^{q} \Pr(IR_a \mid H_0)} = \sum_{a=1}^{q} \ln \frac{\Pr(IR_a \mid H_1)}{\Pr(IR_a \mid H_0)} \tag{2}$$

We set NT_q to the number of times that $IR_a = 1$ in the *q* samples. Then Equation 2 is reshaped to

$$LR_q = NT_q \ln \frac{SP_1}{SP_0} + (q - NT_q) \ln \frac{1 - SP_1}{1 - SP_0} \tag{3}$$

where $SP_0 = \Pr(IR_a = 1 \mid H_0)$, $SP_1 = \Pr(IR_a = 1 \mid H_1)$.

If $NT_q \le \frac{\ln \frac{B}{1-A} + q \ln \frac{1-SP_0}{1-SP_1}}{\ln \frac{SP_1}{SP_0} - \ln \frac{1-SP_1}{1-SP_0}}$ holds, the SPRT discontinues in the adoption of H_0. If $NT_q \ge \frac{\ln \frac{1-B}{A} + q \ln \frac{1-SP_0}{1-SP_1}}{\ln \frac{SP_1}{SP_0} - \ln \frac{1-SP_1}{1-SP_0}}$ holds, the SPRT discontinues in the adoption of H_1. If $\frac{\ln \frac{B}{1-A} + q \ln \frac{1-SP_0}{1-SP_1}}{\ln \frac{SP_1}{SP_0} - \ln \frac{1-SP_1}{1-SP_0}} < NT_q < \frac{\ln \frac{1-B}{A} + q \ln \frac{1-SP_0}{1-SP_1}}{\ln \frac{SP_1}{SP_0} - \ln \frac{1-SP_1}{1-SP_0}}$ holds, the SPRT continues with an additional sample. Note that *A* is a false positive error rate set by user and *B* is a false negative error rate set by user.

If the SPRT adopts H_1, Android evasive malware determines that dynamic analyzer runs in Android system and provides inputs to it, and thus it performs benign task. If the SPRT adopts H_0, Android evasive malware determines that human user operates in Android system and provides inputs to it, and hence it executes malign task. If the SPRT does not adopt any hypothesis, Android evasive malware runs benign task. As long as Android evasive malware is running, it keeps performing the SPRT.

In our prior work [10], the SPRT takes it as evidence the number of events that are highly likely chosen by dynamic analysis algorithm out of randomly chosen events, while the number of inputs provided to Android evasive malware in a time slot is used as evidence in our proposed work. Putting it in different way, the evidence of the SPRT is selected from the perspective of malicious app detection application based on dynamic analysis algorithm in [10], while the evidence of the SPRT is chosen from the perspective of the SPRT-based Android evasive malware in our proposed work. Moreover, [10] focuses on deciding whether the malicious app detection application based on dynamic analysis algorithm has vulnerability of being eluded by malicious apps or not. On the other hand, our proposed work focuses on determining whether dynamic analyzer provides inputs to Android evasive malware while running in Android system or not. The SPRT-based android evasive malware decides whether to execute malicious task in accordance with decision result. Hence, the SPRT-based Android evasive malware is different from [10].

In addition, our another prior work [8] utilizes game theory to perform security analysis against input-driven evasive malware in the IoT. Although game theoretic analysis is mainly dealt in [8], the SPRT also plays a role of deciding whether input-driven evasive malware launches malicious task or not in multi-stage game. However, toward-benign-decision and toward-malicious-decision strategies taken by input-driven evasive malware in multi-stage game are used as the evidence of the SPRT in [8], while the number of inputs given to Android evasive malware in a time slot is used as evidence in our proposed work. Therefore, our proposed work is distinct to our prior work [8].

4. Conclusion

In this paper, we explore the SPRT-based Android evasive malware, which is a new type of Android evasive malware that does not initiate malicious task when it discerns that dynamic analyzer operates and provides inputs to it in Android system. In the sense that the SPRT-based Android evasive malware can run away from the detection of dynamic analyzer running in Android system, it is detrimental to the security of Android systems, thus the effective and secure defense mechanism against it needs to be devised.

Acknowledgement

This work was supported by a research grant from Seoul Women's University (2022-0136).

References

[1] S. Hao, B. Liu, S. Nathy, W.G.J. Halfond, R. Govindan. PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps. In ACM MobiSys, 2014. DOI: https://doi.org/10.1145/2594368.2594390.

[2] Y. Li, Z. Yang, Y. Guo, and X. Chen. DroidBot: A Lightweight UI-Guided Test Input Generator for Android. In IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 2017. DOI: https://doi.org/10.1109/ICSE-C.2017.8.

[3] https://developer.android.com/studio/test/monkey.

[4] S. D. Yalew, G. Q. Maguire, S. Haridi, and M. Correia. T2Droid: A TrustZone-Based Dynamic Analyser for Android Applications. In *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 240-247. DOI: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.243.

[5] A. Dawoud, S. Bugiel. Bringing Balance to the Force: Dynamic Analysis of the Android Application Framwork. In NDSS 2021. DOI: https://doi.org/10.14722/ndss.2021.23106.

[6] A. Wald. Sequential Analysis, Dover, 2004.

[7] Y. Li, Z. Yang, Y. Guo, and X. Chen. Humanoid: A Deep Learning-Based Approach to Automated Black-box Android App Testing. In 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 1070-1073. DOI: https://doi.org/ 10.1109/ASE.2019.00104.

[8] Ho, Jun-Won (2022): Game Theoretic Security Analysis against Input-Driven Evasive Malware in the IoT. TechRxiv. Preprint. DOI: https://doi.org/10.36227/techrxiv.19633677.v1

[9] Ho, Jun-Won. GAME THEORY BASED DYNAMIC ANALYSIS INPUT SYSTEM AND METHOD FOR INTELLIGENT MALICIOUS APP DETECTION. Republic of Korea Patent. Registration Number/Date: 1022106590000/(2021.01.27).

[10] Ho, Jun-Won. METHOD AND APPARATUS FOR DIAGNOSING MALICIOUS APP DETECTED APPLICATION. Republic of Korea Patent. Registration Number/Date: 1020995060000 (2020.04.03).

[11] J. Blackthorne, A. Bulazel, A. Fasano, P. Biernat, and B. Yener. AVLeak: Fingerprinting Antivirus Emulators Through Black-Box Testing. In USENIX Workshop on Offensive Technologies, 2016.

[12] D. C. DElia, E. Coppa, F. Palmaro, and L. Cavallaro. On the Dissection of Evasive Malware. In IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2750-2765, 2020. DOI: https://doi.org/10.1109/TIFS.2020.2976559.

[13] W. Diao, X. Liu, Z. Li, and K. Zhang. Evading Android Runtime Analysis Through Detecting Programmed Interactions. In ACM WiSec, 2016. DOI: https://doi.org/10.1145/2939918.2939926.

[14] Y. Jing, Z. Zhao, G.-J. Ahn, and H. Hu. Morpheus: Automatically Generating Heuristics to Detect Android Emulators. In Proceedings of the Annual Computer Security Applications Conference (ACSAC), 2014. DOI: https://doi.org/10.1145/2664243.2664250.

[15] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis. Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artifacts. *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 1009-1024, DOI: https://doi.org/ 10.1109/SP.2017.42.

[16] H. Shi, J. Mirkovic, and A. Alwabel. Handling Anti-Virtual Machine Techniques in Malicious Software. In ACM Transactions on Privacy and Security, Article No.2, December 2017. DOI: https://doi.org/10.1145/3139292.
[17] J. Wampler, I. Martiny, and E. Wustrow. ExSpectre: Hiding Malware in Speculative Execution. In Network and Distributed Systems Security(NDSS) Symposium, 2019.
[18] D. Kirat, G. Vigna, C. Kruegel. BareCloud: Bare-metal Analysis-based Evasive Malware Detection. In Usenix Security, 2014.
[19] X. Wang, S. Zhu, D. Zhou, and Y. Yang. Droid-AntiRM: Taming Control Flow Anti-analysis to Support Automated Dynamic Analysis of Android Malware. In ACSAC, 2017, Pages 350–361. DOI: https://doi.org/10.1145/3134600.3134601
[20] X. Wang, Y. Yang, and S. Zhu. Automated Hybrid Analysis of Android Malware through Augmenting Fuzzing with Forced Execution. In IEEE Transactions on Mobile Computing, vol. 18, no. 12, pp. 2768-2782, 2019. DOI: https://doi.org/10.1109/TMC.2018.2886881.
[21] J. Zhang, Z. Gu, J. Jang, D. Kirat, M. Stoecklin, X. Shu, H. Huang. Scarecrow: Deactivating Evasive Malware via Its Own Evasive Logic. In50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2020, pp. 76-87.
[22] X. Pan, X. Wang, Y. Duan, X. Wang, and H. Yin. Dark Hazard: Learning-based, Large-scale Discovery of Hidden Sensitive Operations in Android Apps. In NDSS, 2017.
[23] L. Bello and M. Pistoia. Ares: Triggering Payload of Evasive Android Malware. In IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft), 2018, pp. 2-12.
[24] S. Mutti, Y. Fratantonio, A. Bianchi, L. Invernizzi, J. Corbetta, D. Kirat,C. Kruegel, and G. Vigna. BareDroid: Large-Scale Analysis of Android Apps on Real Devices. In ACSAC, 2015. DOI: https://doi.org/10.1145/2818000.2818036.