Gaming Mobile Applications: Proof of Concept for Security Exploitation

Mohd Haizam Saudi[1], Madihah Mohd Saudi [2, *], Arief Rahmana[3], Muhammad Afif Husainiamer [4]

[1] Widyatama University
[2]CyberSecurity and Systems (CSS) Research Unit, Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), 71800 Nilai, Negeri Sembilan, Malaysia
[3] Widyatama University
[4]CyberSecurity and Systems (CSS) Research Unit, Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), 71800 Nilai, Negeri Sembilan, Malaysia; afif@raudah.usim.edu.my
[2]madihah@usim.edu.my

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 20 April 2021

Abstract: Playing games is fun for game lovers. Many of us especially kids and teenagers spend more time and prefer playing online game compared to the traditional way of the game. They could get more cyber friends and it is a more challenging experience when playing online games. Yet, many of the gamers lack knowledge in preventing security exploitation when playing online mobile game applications(apps). For example, many malwares such as Trojans and worms camouflaged and embedded themselves inside the game especially during installation. Hence this paper presents a proof of concept (POC) security exploitation for mobile gaming applications by using our developed model called Mobotder. It will detect any possible data breach or security exploitation based on geolocation (GPS), permissions, and Application Programming Interface (API) calls. The Mobotder was created and hybrid analyzed in a controlled lab environment, by using open source tools and datasets from Drebin and Google Play Store for training and evaluation. Furthermore, ten (10) anonymous mobile games were downloaded from Google Play Store and evaluated by using the Mobotder model. The result showed that 7 of the games were identified as medium risk. The POC and the model developed could be used as guidance to build secure mobile gaming in the future.

Keywords: gaming exploitation, malware, mobile security, data breach, permission, Application Programming Interface (API).

1. Introduction

Currently, there are so many different types of games available on the market. Even education and awareness programs are also available in game form[1]. For example, a serious game called Riskoi was invented for cybersecurity awareness and education [2]. Furthermore, a paper by [3] also discussing in detail how gamification and Alternate Reality Games (ARGs) could be used to teach privacy and security subjects. In terms of game security exploitation, there are so many scenarios where the unauthorized game installation was carried out from untrusted app stores [4]. The worst part would be where these untrusted app stores hosted cloned games with malware embedded in it. Furthermore, many gamers lack knowledge in preventing security exploitation when playing online mobile game applications (apps). For example, many malwares such as Trojans and worms camouflaged and embedded themselves inside the game especially during installation. As for the game that requires online purchases, the security main concern is on the payment gateway security exploitation. While for online games, security exploitation inclusive of steal confidential information such as username, password and bank account details, online attacks, and malicious attempts to harm other players [5]. Surprisingly, in December 2020, four security vulnerabilities were identified from Valve, a popular gaming platform on the gaming networking sockets of Steam [6]. Hence, based on the security challenges identified, this paper presents proof of concept (POC) of possible security exploitation for mobile game apps by using the Mobotder model. This model would be able to detect any possible cyberattacks by using permission, API, and system call on a mobile phone. To prove the effectiveness of this model, ten (10) mobile game apps were evaluated to check their level of possible data security exploitation.

This paper is organized as follows: Section 2 presents the related existing works and methods used, followed by experimental results in Section 3, findings in Section 4, and finally Section 5 we conclude this paper together with future work.

2. Methods

Relevant existing works by [7-12] were reviewed and analyzed in designing the Mobotder model. Based on our analysis, we identified that feature selection is crucial prior designing the mobile detection model. The right

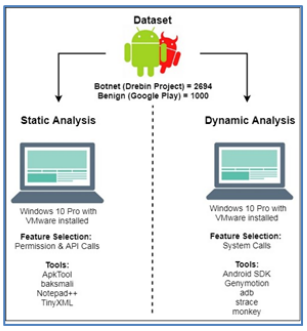
selection of feature selection will be affecting the accuracy rate produced [13]. As for the Mobotder model, we identified that geolocation(GPS) is one of the most affected surveillance features used in a mobile phone for security exploitation together with permissions, APIs and system calls [14,15].

2.1 Data collection

The dataset comprising 5560 malwares from 179 different botnet families from Drebin [16]. We selected 2694 botnet samples from 44 different botnet families with datasets were divided into 70% for training and 30% for evaluation. Percentage split is chosen for easy implementation in real mobile devices and to avoid overfitting [17]. Many other existing works such as by [18-23] used the Drebin dataset for their experiment. As for POC for this paper, we used 10 anonymous mobile game apps from Google Apps Store.

2.2 Lab Architecture

Figure 1 demonstrates the lab architecture of this paper which was conducted in a controlled lab environment by using open source tools.



Software	Function
VMware Workstation	For virtual workstation in allowing multiple operating systems to run on a single computer.
Genymotion Emulator	It was used as a smartphone android emulator.
VirusTotal (online)	It was used as a scan tool.
ApkTool	It was used for reverse engineering.
TinyXML	It was used to parse Java code.
Smali/baksmali	It was used for assembler/disassembler dex format file.
Android SDK	It was used for android application development.
Java (Weka)	It was used for data testing and simulation.
Notepad++	It was used as a text editor to view source code.

Figure 1. Experiment Lab Architecture and Software Installed.

2.3 Data Extraction and Analysis

Figure 2 depicted the data extraction and data analysis for this paper. We used permission and GPS as basic components for the Mobotder model development. This paper used hybrid analysis which consists of static and dynamic analyses. Permission protects the privacy of Android users and it is derived from AndroidManifest.xml. Permission consent is required from the user before installing any mobile Android application.

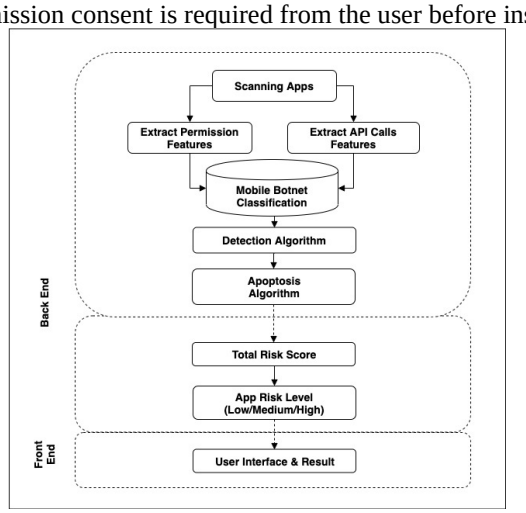


Figure2. Mobotder Development Method

3. Findings

To evaluate the effectiveness of the Mobotder model, a comparison with the existing works by [24-27] was made and summarized in Table 1. Mobotder showed a 99.1% accuracy rate with Random Forest as the classifier and has outperformed other existing works.

Table 1. Comparison with Existing Works.

Feature	Work by [24]	Work by [25]	Work by [26]	Work by [27]	Mobotder
*ML Classifier	Random Forest 93.9	PSO-ANFIS 89	Random Forest 97.24	Random Forest 97.48	Random Forest 99.1

Accuracy Rate(%)	

*ML=Machine Learning

Then we have selected ten (10) random mobile game apps from the Google Play Store for further evaluation with the Mobotder model. The experiment results as shown in Table 2. The detailed description for each permission is displayed in Table 3. Based on Table 2, 30% of the mobile game apps are considered as low risk, while 70% as medium risk.

Feature	Risk	Description
Application1	Low	PER8+ PER12
Application2	Medium	PER4+PER5+PER9+PER10+PER12+PER17+PER18
Application3	Medium	PER4+PER5+PER6+PER9+PER10+PER12+PER17+PER18+PER20
Application4	Medium	PER4+PER5+PER6+PER7+PER9+PER10+PER12+PER13+PER17+PER18+PER20+PER23+PER25
Application5	Medium	PER4+PER5+PER10+PER12+PER17+PER18
Application6	Medium	PER1+PER4+PER5+PER10+PER12+PER17+PER18
Application7	Medium	PER1+PER2+PER4+PER5+PER6+PER9+PER10+PER12+PER17+PER18
Application8	Low	PER4+PER5+PER12
Application9	Low	PER4+PER5+PER6+PER7+PER9+PER12+PER17
Application10	Medium	PER4+PER5+PER10+PER12+PER17+PER20

Table 2. Mobile Gaming Apps Experiment Results.

Table 3. Description of Related Permissions for Mobotder Model

Feature	Description
PER1: Access_Coarse_Location	Allows an app to access approximate location.
PER2: Access_Fine_Location	Allows an app to access precise location.
PER3: Access_Location_Extra_Commands	Allows an app to access extra location provider commands
PER4: Access_Wifi_State	Allows an app to access information about Wi-Fi networks.
PER5: Bluetooth	Allows an app to connect to paired bluetooth devices.
PER6: Bluetooth_Admin	Allows an app to discover and pair bluetooth devices.
PER7: Change_Wifi_Multicast_State	Allows an app to enter Wi-Fi Multicast mode.
PER8: Change_Wifi_State	Allows an app to change Wi-Fi connectivity state.
PER9: Install_Packages	Allows an app to install packages.
PER10: Install_Shortcut	Allows an app to install a shortcut in Launcher.
PER11: Internet	Allows an app to open network sockets.
PER12: Kill_Background_Processes	Allows an app to kill the background process
PER13: NFC	Allows an app to perform I/O operations over NFC
PER14: Read_Call_Log	Allows an app to read the user's call log.
PER15: Read_Contacts	Allows an app to read the user's contact data.
PER16: Read_External_Storage	Allows an app to only read the external storage
PER17: Read_Phone_State	Allows read-only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any phone accounts registered on the device.
PER18: Read_SMS	Allows an app to read SMS messages.

PER19: Receive_Boot_ComRleted	Allows an app to receive the Intent.ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.
PER20: Receive_MMS	Allows an app to monitor incoming MMS messages.
PER21: Receive_SMS	Allows an app to receive SMS messages.
PER22: Restart_Packages	Allows an app to close processes of other applications.
PER23: Send_SMS	Allows an app to send SMS messages.
PER24: System_Alert_Window	Allows an app to create windows using the type WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY, shown on top of all other applications.
PER25: Uninstall_Shortcut	Allows an app to uninstall shortcut.
R26: Update_Device_Stats	Allows an app to update device statistics
R27: Write_Apn_Settings	Allows an app to write the APN settings.
PER28: Write_Call_Log	Allows an app write to user's call log
PER29: Write_Contacts	Allows an app to write the user's contact data.
PER30: Write_SMS	Allows an app to send SMS

4. Discussion

Based on our findings in the earlier section, it showed that 70% of the mobile game apps were exposed to possible medium-risk security exploitation by the attackers and followed by 30% with low risk. By using the Mobotder model, the gaming developer will be able to use it as guidance in developing a secure gaming mobile app. As for users or gamers, they could use Mobotder as a layer of defense to detect possible security exploitation in their installed game apps. Nonetheless, users must understand what kind of mobile apps they have installed since these permissions might pose financial risks, especially at the payment gateway. Furthermore, it is highly recommended for the users not to install mobile game apps from untrusted parties, understand the End User License Agreement (EULA) that allowing tracking cookies or spyware, avoid installing crack mobile game apps, and only allow related permission for the installed mobile apps.

5. Conclusions

Based on the findings presented in this paper, the Mobotder model has successfully detected possible security exploitation for mobile gaming apps. Yet at users and software game developers site, mitigation mechanism against malware exploitation is very significant. This includes monitoring the flaws in the In-App purchasing system, unauthorized installation, app security in real-time, securing payment gateway, securing the mobile devices used to install the game apps, and secure gaming coding. On top of that, user awareness and solution such as Mobotder offers proper mitigation mechanism. For future work, software gaming developers could use this paper as a guideline in developing secure mobile gaming apps.

Acknowledgment

The authors would like to express their gratitude to Widyatama University, Indonesia and Universiti Sains Islam Malaysia (USIM) (USIM grant no: P1-17-16120-UNI-CVD-FST) for the funding, support, and facilities provided.

References

1. Khoury, J., & Nassar, M. (2020, April). A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1-9). IEEE.
2. Hart, S., Margheri, A., Paci, F., & Sassone, V. (2020). Riskio: A serious game for cyber security awareness and education. Computers & Security, 95, 101827.
3. Karagiannis, S., Papaioannou, T., Magkos, E., & Tsohou, A. (2020, November). Game-Based Information Security/Privacy Education and Awareness: Theory and Practice.
4. Parizi, R. M., Dehghantanha, A., Choo, K. K. R., Hammoudeh, M., & Epiphaniou, G. (2019). Security in online games: Current implementations and challenges. In Handbook of Big Data and IoT Security (pp. 367-384). Springer, Cham.
5. Maguluri, N. S. N. (2017). Multi-Class Classification of Textual Data: Detection and Mitigation of Cheating in Massively Multiplayer Online Role Playing Games. (Thesis), Wright State University, Browse all Theses and Dissertations.

6. Sue Poremba, (2021, January). Online Gaming Adds More Risk to WFH.Security Boulevard. Available online: https://securityboulevard.com/2021/01/online-gaming-adds-more-risk-to-wfh/[accessed 24th March 2021].
7. Irwin,L. (2021). List of data breaches and cyber attacks in February 2021 – 2.3 billion records breached, IT Governance UK Blog, Available online: https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-february-2021-2-3-billion-records-breached [accessed: 28th March 2021]
8. Alqatawna, J. F., Ala'M, A. Z., Hassonah, M. A., & Faris, H. (2021). Android botnet detection using machine learning models based on a comprehensive static analysis approach. Journal of Information Security and Applications, 58, 102735.
9. Yerima, S. Y., & Alzaylaee, M. K. (2020, June). Mobile botnet detection: A deep learning approach using convolutional neural networks. In 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA) (pp. 1-8). IEEE.
10. Takahashi, T., & Ban, T. (2019). Android Application Analysis Using Machine Learning Techniques. In L. F. Sikos (Ed.), AI in Cybersecurity. Intelligent Systems Reference Library (pp. 181–205). Springer, Cham. https://doi.org/10.1007/978-3-319-98842-9.
11. Alshahrani, H., Mansourt, H., Thorn, S., Alshehri, A., Alzahrani, A., & Fu, H. (2018). Defender: Android Application Threat Detection Using Static and Dynamic Analysis. In Proceeding of 2018 IEEE International Conference on Consumer Electronics, ICCE 2018 (Vol. January, pp. 1–6). https://doi.org/10.1109/ICCE.2018.8326293
12. Sun, M., Li, X., Lui, J. C. S., Ma, R. T. B., & Liang, Z. (2017). Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android. IEEE Transactions on Information Forensics and Security, 12(5), 1103–1112. https://doi.org/10.1109/TIFS.2016.2646641
13. Yang, F., Zhuang, Y., & Wang, J. (2017). Android Malware Detection Using Hybrid Analysis and Machine Learning Technique. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 10603 LNCS, pp. 565–575). https://doi.org/10.1007/978-3-319-68542-7_48
14. Bhattacharya, A., & Goswami, R. T. (2017). Comparative Analysis of Different Feature Ranking Techniques in Data Mining-Based Android Malware Detection. In J. K. Mandal, S. C. Satapathy, M. K. Sanyal, & V. Bhateja (Eds.), Proceedings of the First International Conference on Intelligent Computing and Communication (Vol. 515, pp. 39–49). Springer, Singapore. https://doi.org/10.1007/978-981-10-3153-3
15. Pieterse, H., & Olivier, M. (2013). Design of a hybrid command and control mobile botnet. Journal of Information Warfare, 12(1), 70–82. Retrieved from https://researchspace.csir.co.za/dspace/bitstream/handle/10204/7385/Pieterse2_2013.pdf
16. Saudi, M. M., Amran, L., & Ridzuan, F. (2020). Go-Detect Application Inspired by Apoptosis to Detect SMS Exploitation by Malwares. In RITA 2018 (pp. 101-116). Springer, Singapore.
17. Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., & Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceeding of the Symposium on Network and Distributed System Security (NDSS) (pp. 23–26). https://doi.org/10.14722/ndss.2014.23247
18. Allibhai, E. (2018). Hold-out vs. Cross-validation in Machine Learning. Available online: https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f [accessed: 28th March 2021]
19. Zhang, H., Luo, S., Zhang, Y., & Pan, L. (2019). An Efficient Android Malware Detection System Based on Method-Level Behavioral Semantic Analysis. IEEE Access, 7, 69246–69256. https://doi.org/10.1109/ACCESS.2019.2919796
20. Shi-Qi et al. (2019)Shi-Qi, L., Bo, N., Ping, J., Sheng-Wei, T., Long, Y., & Rui-Jin, W. (2019). Deep Learning in Drebin: Android Malware Image Texture Median Filter Analysis and Detection. KSII Transactions on Internet and Information Systems, 13(7), 3654–3670. https://doi.org/10.3837/tiis.2019.07.018
21. Onwuzurike, L., Mariconti, E., Andriotis, P., De Cristofaro, E., Ross, G., & Stringhini, G. (2019). Mamadroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. ACM Transactions on Privacy and Security, 22(2). https://doi.org/10.1145/3313391
22. Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G., & Furnell, S. (2017). AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. Computers and Security, 65, 121–134. https://doi.org/10.1016/j.cose.2016.11.007
23. Karbab, E. M. B., Debbabi, M., Derhab, A., & Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. In Proceedings of the 5th Annual DFRWS 2018 Europe (Vol. 24, pp. S48–S59). Elsevier. https://doi.org/10.1016/j.diin.2018.01.007

24. Tansettanakorn, C., Thongprasit, S., Thamkongka, S., & Visoottiviseth, V. (2016). ABIS: A prototype of Android Botnet Identification System. Proceedings of the 2016 5th ICT International Student Project Conference, ICT-ISPC 2016, 1–5. https://doi.org/10.1109/ICT-ISPC.2016.7519221
25. Altaher, A., & Mohammed, O. (2017). Intelligent Hybrid Approach for Android Malware Detection based on Permissions and API Calls. International Journal of Advanced Computer Science and Applications, 8(6).
26. Rana, M. S., Motiur Rahman, S. S. M., & Sung, A. H. (2018). Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 11056 LNAI, pp. 377–385). Springer International Publishing. https://doi.org/10.1007/978-3-319-98446-9_35
27. Qamar, A., Karim, A., & Shamshirband, S. (2019). A Learning Based Framework for Detection of Android C&C Enabled Applications Using Hybrid Analysis. Preprints. https://doi.org/10.20944/preprints201906.0060.v1