ZULFAQAR Journal of
Defence Science, Engineering & Technology

Journal homepage: https://zulfaqar.upnm.edu.my/

ANDROID DESIGNED MALWARE DETECTION CHALLENGES: A FUTURE RESEARCH DIRECTION

Afiqah Mohammad Azahari[a,*], **Arniyati Ahmad**[a], **Syarifah Bahiyah Rahayu**[a], **Nur Diyana Kamarudin**[a], **Mohd Hazali Mohamed Halip**[a]

[a] *Cyber Security Centre, National Defence University of Malaysia*

*Corresponding author: *afiqah.azahari@upnm.edu.my*

ARTICLE INFO

Article history:
Received
14-05-2020
Received in revised
30-10-2020
Accepted
10-02-2021
Available online
30-06-2021

Keywords:
Android, mobile malware, malware detection, malware type, security threats

e-ISSN: 2773-5281
Type: Article

ABSTRACT

Statistically, Android is the most targeted mobile platform when it comes to malicious application. As a result, Android malware detection has become one of the sizing topics in the domain of mobile security. As the researchers focusing on developing a new approach to detect and fight Android malware, there are always a recent report exhibiting cases of Android malware. Multiple motivations cause mobile malware writers to continuously develop an application with malware. Their intentions are to gain access to the private network and to collect sensitive data. This paper categories type of mobile malware. Furthermore, the types of mobile malware that often attacks android's users are discussed. Then, fundamental techniques usually implement to detect mobile malware are deliberated. Basic techniques such as Static, Dynamic and Hybrid analysis are explained in the section. Finally, open issues on detecting and evaluating Android designed malware presented as a guideline for future research directions.

Introduction

Smartphone could be used to perform many functions similar to PCs. In 2010, the smartphone has been surpassing PCs in terms of its shipments. Personalisation and powerful performance factors make smartphones, tablets, and other mobile platforms become ubiquitous of consumer-electronic devices. After iPhone penetrate the market back in 2007, most of the smartphone was designed with a capacitive screen that supports multi-touch gestures in thin, slate-like form factor (Falaki et al., 2010). Its innovation makes users be able to access the Internet wirelessly, offered with an ability to download or purchase applications, use cloud or synchronise cloud storage, access to virtual assistants, and as well as making a payment.

Technically, not every smartphone offers the same hardware features. The features are defined by the technologies that own by the smartphone company. Examples of technologies in smartphones are including face recognition or fingerprint authentication system, various sensors, NFC (Near Field Communication), etc. Due to the different technologies used, the Operating System (OS) is responsible to provide a proper control of it. OS in the smartphone is used to control or perform functions as booting, memory management, loading and execution, data security, risk management, etc. Different smartphone OS is using a different approach to run an application or perform a technological function. Latest and popular smartphone OS can be classified into 5 different categories. The categories are Android, Apple iOS, Microsoft Window Phone and Blackberry.

Recently, Google reports that the Android operating system has reached a breakthrough with more than 2.5 billion monthly active Android user, exceeding the number of Apple iOS users by 1.1 billion (Tung, 2019). One of the prime contributing factors to the popularity of Android is the user's growth. Android is using an open-source OS with a maximum global download application in the market. However, Android's popularity and its attractive environment not only attract users' attention, but it also has encouraged malware authors to develop malicious applications (or apps) to penetrate the security of mobile devices.

Category of Mobile Malware

Malicious software or also known as malware is any software that developed with malicious intention. Mobile malware could be developed to disrupt normal functioning, exploit access control, collect sensitive information, display unwanted ads, or control the mobile without the user's knowledge (Al-khatib & Hammood, 2017). There are many conventional types of mobile malware that existed since the introduction of smartphones. Therefore, Table 1 shows a summary of several types of mobile malware with its behaviour.

Table 1: Category of Malware with Its Threat and Examples (Lachtar et al. 2019; Sigler and Trustwave 2018)

Category	Threats	Example
Trojan	A type of malware that shows itself as a benign app to attract the user to download and install to the device. It will perform a malicious act on the background. Trojan will try to gain remote access to steal, modified, or delete files. Trojan also spy on users' activities either by monitoring the screen, look up for device logs, etc.	*DownAPK, GantSpy, DroidKungFu*
Mobile Worm	The worm is capable to duplicate and spreading itself from device to device through the network. Worm could destroy the host network by utilising bandwidth.	*Cabir, Feakk, Mobler, InSpirit, Ikee.B*
Mobile Virus	The mobile virus spread by attaching itself to an app. Phone infected by the virus may be exposed to the threat of information stealing, network issue, etc.	*Dust, Lasco, Cardblock, CardTrap, and Crossover*
Ransomware	Ransomware encrypts files of infected mobile and will not release the resources until some amount of ransom was made (Lachtar, Ibdah, and Bacha, 2019).	*Ransom.BE78, Simplocker*
Adware	Without permission from the user, adware bundled with other apps and deliver ads. Adware could quietly operate in the background and trawling through private information, such as username, password, contact, etc.	*UAPush*
Spyware	Spyware will keep an eye on any activities of infected mobile devices. These activities consist of collecting key log, screen watching, and stealing account information. Spyware attaches itself to a benign application or Trojan to exploit the vulnerability.	*Zitmo, Acallno, FlaxiSpy*
Mobile Crypto-jacking	Crypto-jacking is when mobile technology was used to mine bitcoin, secretly. Crypto-jacking runs behind the popular app. Some secretly mining while streaming a video. Some secretly mining cryptocurrency while soccer video.	*WebCobra, HiddenMiner*

Malware is evolving to the point where it is more sophisticated and harder for the user to notice. This section has shown that mobile threats are increasing rapidly, and its attack is focusing on a specific target. Therefore, there is important to have efficient detection of malware to stop the malware attack.

Android Malware Detection Strategies

There are multiple approaches used by researchers to analyse Android malware. The detection approaches included are static, dynamic, and hybrid techniques (Tong & Yan, 2017). These techniques are available to detect malicious intentions to the host, which is a mobile phone. Therefore, a different method based on detection techniques and its effectiveness and weakness will be discussed.

i. Static Analysis

Static analysis detection was done when application inspection is made without the execution of the program. The static technique relies on the application source code to differentiate between malware and benign apps. Hence, code coverage was maximised as this technique depends on the analysis of the source code. There are multiple methods developed in detecting malicious malware using static analysis. There is permission analysis, which is among popular techniques. An example of permission-based static analysis is *DroidDetective*. *DroidDetective* detecting malware by comparing several requested permissions in a benign and malicious application. To run the application properly, the developer needs to declare permission in androidmanifest.xml. Showing 96% detection rate, *DroidDetective* reveals permission, such as WRITE_SMS, RECEIVE_SMS, SEND_SMS and READ_SMS mostly invited by malware, but not in benign applications (Liang & Du, 2014). By exploiting and asking for these permissions, a malicious application could send a premium-rate message without interference from the users, therefore, cause financial loss.

The permission-based static analysis could give a high detection rate, but in some cases, it also could result in a high false-positive rate. Moreover, other researchers use behaviour of data flow in determining the malicious application. *FlowDroid* is a detection model that analyse the call backs invoked by the Android framework. To reduce false alarm, *FlowDroid* properly handles call backs whereas analysis on data flow, context, and objects. *FlowDroid* has been successfully detecting malware? (Arzt et al., 2018). There were also several approaches to static analysis of Android malware detection differing in runtime, scope, and focus. Still, most of the objective of developing a malware detection method is to reduce missed leak and false positive. Static analysis helps in revealing apps with malware before actual execution. However, static analysis is ineffective to detect malware with code obfuscation and dynamic code loading. Moreover, this technique also impractical in detecting zero-day malware (Gandotra et al., 2016). Thus, another technique is needed to overcome the issues.

ii. Dynamic Analysis

Dynamic analysis includes monitoring various run-time activities such as registry changes, network activities, or data flow tracking. This was done when the application is executed in a controlled environment. The purpose of dynamic analysis is to find an analysis of the behaviour of the apps while it is executed. An example of a method that uses dynamic analysis is *TaintDroid* that is introduced by Enck et al. (2019) *TaintDroid* tracks the flow of privacy-sensitive data through a third-party application. It assumes that downloadable third-party applications are not trustable. This method will classify misbehave application by logging the tainted data transmitting over the network or leaving the system. It will record data's labels, the application that responsible for transmitting the data, and data's destination (Arzt et al., 2018). This information will give users and security service a greater insight on what mobile applications are doing and potentially could identify the malicious application.

On the other hand, Arzt et al. (2018)analyse the network traffic by generating the URL table and logs all attempts made by application to remote servers. Each log in the URL table will preserve the application identification and URL that the application contacted. By comparing logs with reliable and comprehensive domain blacklist, this method could detect applications that commute with malicious domain. Chen et al. (2018) combine technique taken by Zaman et al. with a machine learning method to identify malicious network behaviour. They capture traffic from over 5,560 mobile malware samples. The accuracy rate of the detection model could reach up to 99.9%. However, this method is server-based analytics. Server-based analytics is impractical to find a newly generated malware that usually initiated at app runtime during its execution. Moreover, malware samples taken for experimentation or proof of concept for this method are from Drebin Project. The samples were collected over the period from August 2010 to October 2012, which is infeasible for today's malware technology.

Although this analysis is more effective compared to the static analysis, these analyses are time and resources consuming. Moreover, executing the malicious software in control and virtual environment may yield different results compare with executing the software in the actual environment. This is because some malicious software is designed to trick the analyst or sometimes its behaviour may only trigger under certain conditions.

iii. Hybrid Analysis

Due to the different effects/behaviours of both malware detection techniques mentioned above, therefore, features on both static and dynamic detection techniques need to be integrated to improve the detection of Android's malware. An in-device malware detection introduces by Martinelli et al. (2017) combines static and dynamic analysis methods. This method analyses n-grams matching for static analysis whilst dynamic analysis is based on multi-level monitoring of devices, apps, and user behaviour. Thousands of samples were processed with detection accuracy that reaches up to 99.7% (Arzt et al., 2018). Besides, there is a novel 3-level hybrid malware detection named *SAMADroid* for the Android operating system. This method combines three (3) different levels of detection, which are static and dynamic analysis, local and remote hosts, and machine learning intelligence.

At level 1, in the static analysis phase, static features are extracted from the manifest file. The static features are grouped by request hardware components, requested permission; intent filters, suspicious API calls, and restricted API calls. Further, in the dynamic phase, the application's runtime behaviour is analysed. The system call was analysed to overcome the limitation of static analysis. Next at level two, on the localhost, the dynamic analysis was performed. Monkey Runner was used to generating non-realistic random input events. All input and behaviour were sent to a remote server, then it will be analysed. At level three, machine learning was implemented to analysed behaviour in the remote host, thus keeping all the training set in a server; therefore, contribute to a big data resource for machine learning. Machine learning intelligence will perform the detection of malicious behaviour of unknown apps and correctly classify them (Arzt et al., 2018). While other malware detection tools scan all the application either running on the background or not, *SAMADroid* only scans and analyse the user application and does not scan the system application. Approach taken by *SAMADroid* is an example of a dynamic malware detection technique where different approaches combined in detecting malicious applications.

This section has briefly explained the methods taken by researchers in detecting Android malware. Researchers have developed numerous methods, which are decent in detecting Android malware.

Android Malware Detection Challenges

Recently, there is malware known as *xHelper* slowly infecting more than 32,000 devices in August 2019 (Cimpanu, 2019). This malware is near impossible to be removed, as it is self-reinstalled even after the infected device was factory reset. In some cases, even *xHelper* service has been removed and the user has disabled the 'Install apps from unknown sources' option, the setting kept turning itself back on and make the device re-infected after a minute being cleaned (Schneier, 2019). Main target users are from India, the United States, and Russia, *xHelper* could not be launched manually as there is no icon visible on the launcher. Instead, the malware was launched when there any external events, such as turning on or off the power supply, rebooting the device, or when there if any apps being installed or uninstalled. Upon successful infecting the victim's device, the additional payload includes droppers, clickers, and rootkits that may be downloaded to further compromise the device (McAfee, 2020). Further, once the device is infected, *xHelper* will find a way to use a process inside the Google Play Store application to trigger the re-install operation. *xHelper* APK will find a way to hide in special directories that it creates when exploiting the device, therefore, surviving factory resets.

Unlike apps, directory and files will stay in the device even factory reset is performed (Cimpanu, 2020). *xHelper* is known as a zero-day type of malware. It takes nearly about 10 months for the researcher to find out how it exploits smartphones and its reliable method of cleaning a smartphone infected by *xHelper*. Detection of new malicious application groups has become challenging as new stealth techniques and encapsulation methods to evade detection tools. Existing mobile antivirus solutions need to be improved to detect and combat highly sophisticated malware. From the review of open literature, it found that researchers have focused on improving detection methods to detect zero-day malware in the malicious application.

Grace et al. (2012) introduce *RiskRanker* in 2012 claimed as a first accurate zero-day android malware detection. Furthermore, *RiskRanker* is the first system that performs large-scale security risk analysis for zero-day malware detection at that time. They proposed a proactive scheme with a two-order risk analysis. In the first-order analysis, *RiskRanker* will construct and analysis high-risk apps and medium risk apps. They will flag high-risk apps if it carries the attack code that exploits a vulnerability in the OS kernel or privilege daemons to obtain superuser privilege. Next, *RiskRanker* will report medium risk apps if the apps secretly monetized users or upload undeniably private information to the remote server. Next, in the second-order analysis, they will collect and correlate various signs or patterns of behaviour among apps with malware. This was done due to mitigating the weakness from first-order analysis, which mainly designed to handle non-obfuscated, encrypted, or dynamically changes payload malicious apps. To demonstrate the effectiveness and *RiskRanker* detection accuracy, the researchers collected in total 104,874 distinct apps, from 15 different Android markets, one from the official marketplace and others from alternative marketplaces. Based on the evaluation, *RiskRanker* has successfully uncovered 718 malicious apps in 29 malware families, including 322 zero-day malware. However, this type of detections has some limitations. Their root exploitation detection scheme depends on signatures, which could only detect known exploits and ignore encrypted or obfuscation exploit during first-order risk analysis. Further, at second order risk analysis, their prototype only considers the `javax.crypto` libraries for convenient encryption detection, while there are multiple types of libraries that could be used by malware writers to defeat the detection method taken by *RiskRanker*.

On the other hand, instead of performing analysis to find malicious behaviour and compare it to other apps to classify malware apps; Zolotukhin et al. (2014) perform an analysis of the behaviour of opcode in benign apps. The executable file of the benign apps is analysed to extract operation code sequences. Then, n-gram analysis was employed to the sequences of operation code to discover essential features. Next, the behaviour model was build based on the finding of the analysis. Finally, the model was used to make a comparison to detect malicious executable of other new files. Operational code or also known as opcode is a machine language instruction that specifies the type of operation to be performed (Kang et al., 2016). Opcode will reveal a significant difference between legitimate apps and malicious apps, which is the right approach to detect malware based on executable files only. To extract features from each of the opcode sequences, the n-gram model was applied. N-gram model is widely used in statistical natural language processing. N-gram word model is applied to transform all opcode sequences to the n-opcodes sequence. For example, this opcode sequence "DEC POP NOP ADD ADD ADD," when transforming to 2-opcodes will become" DEC POP"," POP NOP" and "NOP ADD" and two in the position corresponding to "ADD ADD". This sequence which known as feature matrix will be analysed to find anomalies, therefore, implemented to build a benign software model. To proof the concept, the researchers divided two sets of files. The files were divided into a training set of 600 files in including 22 infected files and second set for a testing set, which includes 400 files with 25 infected files. Moreover, infected files in the testing set belong to five malwares, which is not in the training set, thus it will refer to zero-day malware apps. The motivation is to detect the five malwares in the training set when the benign software model executed to the training set. Once executed, the model successfully detects numbers of malware in the training sets, hence proofing that this method could be implemented in identifying zero-day malware attacks (Zolotukhin & Hämäläinen, 2014).

This technique considers all benign opcode features and builds maximum 2-opcodes sequence; it takes a long time to train the data set to build a benign software model. Therefore, this model is impractical to apps with high instruction, as it will be resulting in big numbers of opcode sequences. Gandotra et al. (2017) show that selecting features obtained from both methods help in shortening the time taken to build a classification model thus hindering the early detection of malware. The project aims to prove that the features selection process would help to improve model-building time without compromising the accuracy of the malware detection system. They illustrated that features selection benefits in shortening the time taken to build classification model, thus may help to overcome the issue arise from Zolotukhin et al. (2014) model, but the discussion on the effectiveness of detecting zero-day malware with features selection was vaguely discussed.

Additionally, Tong and Yan propose a method that dynamically monitors and collects execution data of apps to create both malware and benign pattern set. The malware detection system was done at the server. The execution data are based on the individual system calls and sequential system calls, which are related to the access of files and networks. If the parameters of the data were chosen properly, the detection accuracy of the proposed method could reach up to 91.76% with FPR lower than 4%. To proof the concept that this method could detect a new type of malware, they collect system call patterns of apps with new malware and compare the patterns in both malicious and benign pattern sets. Furthermore, the detection accuracy of this method could be further improved, as there is the implementation of self-learning to the pattern sets (Tong & Yan, 2017).

However, this method does not support real-time detection due to the big data processing and new call patterns of newly detected malware. Privacy of data extracted from the apps to the server also not considered. The data extracted and stored were large, thus need big storage to support. There are multiple methods taken by researchers to detect new types of malware. In addition, the method for zero-day malware detection and analysis are still imperfect, ineffective, and incomprehensive. Since the Android application is available not only at the official store, but there are also many security and privacy problems persuaded. Today, malware detection mechanisms are still incapable to deal with constantly appearing new types of neither malware, such as *xHelper* malware nor the existing ones, until an instant of this malware has damaged several mobile phones. Therefore, it is important to have a new detection method to detect malware unseen previously, in real-time environments with less consumption of resources in terms of memory consumption, use of storage and CPU processing.

This section highlighted some challenges and issues in the perspective of detecting Android malware. In summary, challenges and issues that should be considered as future research direction are as follows:

i. New detection technique to detect the zero-day type of malware.
ii. New detection technique to identify mobile malware in real-time.
iii. Datasets with updated malware family for evaluating malware detection technique.
iv. New mobile malware detection technique with the clear implementation of data privacy.

Conclusion

This work provides state-of-the-art discussions on detecting and evaluating malware focusing on attacking smartphones with Android operating system and its challenges. To do so, this paper has categorized the type of mobile malware such as Trojan, Adware, crypto jacking, etc. This paper also highlighted most popular and fundamental techniques usually used for detecting malware either in the smartphone that used Android or iOS operating system which are through dynamic analysis, static analysis, and hybrid analysis. As a side of contribution, newly type of Android malware is presented and its challenges to detect and evaluate are (such as detection and evaluation/assessment) are addressed. In the end, this paper suggested the need of new detection technique to detect the zero-day type of malware and detection technique to identify Android mobile malware in real-time with a clear implementation of user's data privacy. Moreover, for research purposes of this fast-evolving research, the dataset of with updated malware family also required to evaluating any new detection technique.

Reference

Al-khatib, A. A., & Hammood, W. A. (2017). Mobile malware and defending systems: Comparison study. *International Journal of Electronics and Information Engineering*, *6*(2), 116–123.

Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., & McDaniel, P. (2018). FLOWDROID: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *ACM SIGPLAN Notices*, *49*(6), 259–269. https://doi.org/10.1145/2594291.2594299

Chen, Z., Yan, Q., Han, H., Wang, S., Peng, L., Wang, L., & Yang, B. (2018). Machine learning based mobile malware detection using highly imbalanced network traffic. *Information Sciences*, *433–434*, 346–364. https://doi.org/10.1016/j.ins.2017.04.044

Cimpanu, C. (2019). *New "unremovable" xHelper malware has infected 45,000 Android devices*. ZDNet.

Cimpanu, C. (2020). *There's finally a way to remove xHelper, the unremovable Android malware*. ZDNet.

Enck, W., Gilbert, P., Chun, B. G., Cox, L. P., Jung, J., McDaniel, P., & Sheth, A. N. (2019). TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010*, 393–407.

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 179–194.

Gandotra, E., Bansal, D., & Sofat, S. (2016). Zero-day malware detection. *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, 171–175. https://doi.org/10.1109/ISED.2016.7977076

Gandotra, E., Bansal, D., & Sofat, S. (2017). Zero-day malware detection. *Proceedings - 2016 6th International Symposium on Embedded Computing and System Design, ISED 2016*, 171–175. https://doi.org/10.1109/ISED.2016.7977076

Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). RiskRanker: Scalable and accurate zero-day android malware detection. *MobiSys'12 - Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 281–293. https://doi.org/10.1145/2307636.2307663

Kang, B. J., Yerima, S. Y., McLaughlin, K., & Sezer, S. (2016). N-opcode analysis for android malware classification and categorization. *2016 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2016*, 13–14. https://doi.org/10.1109/CyberSecPODS.2016.7502343

Lachtar, N., Ibdah, D., & Bacha, A. (2019). *The Case for Native Instructions in the Detection of Mobile Ransomware*. *2*(2), 26–29.

Liang, S., & Du, X. (2014). Permission-combination-based scheme for Android mobile malware detection. *2014 IEEE International Conference on Communications, ICC 2014*, 2301–2306. https://doi.org/10.1109/ICC.2014.6883666

Martinelli, F., Mercaldo, F., & Saracino, A. (2017). BRIDEMAID: An Hybrid Tool for Accurate Detection of Android Malware. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 899–901. https://doi.org/10.1145/3052973.3055156

McAfee. (2020). *McAfee Mobile Threat Report Mobile Malware Is Playing Hide and Steal*.

Schneier, B. (2019). *xHelper Malware for Android*. 8 November 2019.

Sigler, K., & Trustwave, S. (2018). Crypto-jacking : how cyber- criminals are exploiting the crypto-currency boom. *Computer Fraud & Security Bulletin*, *2018*(9), 12–14. https://doi.org/10.1016/S1361-3723(18)30086-1

Tong, F., & Yan, Z. (2017). A hybrid approach of mobile malware detection in Android. *Journal of Parallel and Distributed Computing*, *103*, 22–31. https://doi.org/10.1016/j.jpdc.2016.10.012

Tung, L. (2019). *Bigger than Windows, bigger than iOS: Google now has 2.5 billion active Android devices*. ZDNet.

Zolotukhin, M., & Hämäläinen, T. (2014). Detection of zero-day malware based on the analysis of opcode sequences. *2014 IEEE 11th Consumer Communications and Networking Conference, CCNC 2014*, 386–391. https://doi.org/10.1109/CCNC.2014.6866599