International Journal of Cyber-Security and Digital Forensics (IJCSDF) 8(3): 177-187
The Society of Digital Information and Wireless Communications (SDIWC), 2019 ISSN: 2305-001

Analysis of Android Malware Detection Techniques: A Systematic Review

Moses Ashawa and Sarah Morris
Centre for Electronic Warfare, Information, and Cyber
Cranfield University, Defence Academy of the United Kingdom, Shrivenham, SN6 8LA
m.ashawa@cranfield.ac.uk , s.I.morris@cranfield.ac.uk

ABSTRACT

The emergence and rapid development in complexity and popularity of Android mobile phones has created proportionate destructive effects from the world of cyber-attack. Android based device platform is experiencing great threats from different attack angles such as DoS, Botnets, phishing, social engineering, malware and others. Among these threats, malware attacks on android phones has become a daily occurrence. This is due to the fact that Android has millions of user, high computational abilities, popularity, and other essential attributes. These factors influence cybercriminals (especially malware writers) to focus on Android for financial gain, political interest, and revenge. This calls for effective techniques that could detect these malicious applications on android devices. The aim of this paper is to provide a systematic review of the malware detection techniques used for android devices. The results show that most detection techniques are not very effective to detect zero-day malware and other variants that deploy obfuscation to evade detection. The critical appraisal of the study identified some of the limitations in the detection techniques that need improvement for better detection.

KEYWORDS

Malware, detection techniques, Android mobile device, detection rate, static detection, hybrid detection, dynamic detection, cyber security.

1. INTRODUCTION

Android has become the most popular operating system in the world of mobile telephony with largest users in different parts of the world. Vast amount of financial applications such as mobile banking and online banking run on this most popular mobile OS. Other sensitive information like health records, passwords and usernames are stored on android phones.

The technological progression in Android has created proportionate attraction by malware writers who are advancing daily to gain financial roots by creating malware applications which can directly break into android mobile device security and snip victims' personal data or request for ransom usually inform of bitcoin after a successful attack is implemented. Malware attack on android has generated critical issues in mobile and security industry at large. According to Symantec report [41], apart from the openness which existed in Android platform, the mobile OS is also discovered to being ubiquitous. These factors increased the occurrence of mobile malware especially on Android planform. This research provides a systematic review by formulating and answering research questions using a methodology provided in section 3. This helps to identify relevant works that are carried out on android malware detection techniques with their strengths and limitations.

1.2 Malware

Different definitions of malware are given by different scholars and researchers depending on the attack vector deployed or the harm caused. Though these definitions may be different, but all converged to same meaning as malicious applications with evil intent.

The research of [1-3] defined malware as simply as malicious codes. This definition considers malware as any piece of program segment which is developed with harmful intents. This definition does not encompass many attack and harm trajectory followed by malware. Looking at information harvest and data leakage, the study of [4-6] defined malware as any computer program that leaks users' private data without their consent. The research cogitate malware as programs that do not

necessary cause harm to a device but gathers information about the activities of such a device and thereafter create disclosure of them to third parties for future attack (s). In the literature survey on SOA project [7] looked at malware as any suspicious program that affects organizational database. Kaspersky [8] looks at malware as a developed computer program which contaminates and exacts damage on users' computers. As wonderful as the above definitions might be, their coverage is limited when looking at the broad spectrum of malware attacks and effects. This research therefore defines malware as any computer code written with evil motives to get unauthorized access into IT infrastructures and digital devices by breaking their security defense parameters and exploiting their vulnerabilities leading to information harvest, data loss, information leakage, file infections, buffer overflow, interruption of computational operations and leading to subsequent physical or operational damage or both.

1.3 Malware Attacks on Android Phones

Malicious programs that are basically designed to attack android Dalvik virtual machine (DVM) and java core libraries respectively can be described as Android malware. Majority of Android users download free applications without critically considering whether such applications are genuinely provided by Google or not. Many do not turn on their Android device application permission monitor [9, 47] to help confirm Apps checked and guaranteed by Google bouncer [10] before installation even when such application over request permission to different resources of the device. The study of [11, 42-43] asserted that application download from unknown sources constitutes one of the major attack vectors through which mobile devices get infected with malware. Besides, vigilance when given permission privileges [44-46] to applications with is a good security practice. Some Android malicious applications exploit vulnerability in the system when such permissions are granted at installation stage after the application download.

Evasion and obfuscation techniques [48-49] deploy by malware to elude detection has made Google play store insecure despite the tremendous effort of Google and the associated companies to review Android applications to avoid malware distribution. Android malware keep emerging daily. On August 7th, 2018, a security network called Palo Alto [12] again identified about 145 malicious applications on the Google play store which appear to be genuine but are not. There are a lot of detection techniques used by many researchers and security companies to detect and prevent this threat (malware) but none of these seem to be perfect.

For knowledge contribution, this study will provide general knowledge on mobile malware and some of the attack tactics malware deploy to execute their payloads and exploit Android vulnerabilities. This knowledge will help to develop further security parameters and tools that will improve Android mobile security. Evaluation of the detection techniques will give an idea on which is more efficacious in detecting malwares on android platforms and which needs to be improved for better detection of sophisticated malware variants. The study provides some of the challenges that digital forensics and cyber security investigators may encounter when working on Android malware.

The remaining part of this paper is organized into the following sections: Section 2 provides the related work. The methodological approach to the research is provided in section 3. Section 4, 5, 6 and 7 are for detection technique, discussion, evaluation and conclusions.

2. RELATED WORK

Research in Mobile security has become a thing of concern. A range of research in mobile technologies from design, vulnerability, threats and detection techniques is ongoing. Many security industries are spending billions of funds in this field. This paper examined some of them as building blocks. The study of [14] x-rayed dynamic detection and analysis approach on how malware can be detected on Gingerbread Android version. DroidScope detected DroidKungFu and DroidDream android malware where the major malware investigated in this research. Detection features obtained by the research indicated that while DroidDream encrypted IMSI and IMEI numbers into XML String, DroidKungFu carried cage exploit to gain unpermitted access to the mobile device security. The outcome of the study demonstrated

efficiency of the technique used was resistant to malware code obfuscation. The observed limitation was that the research did not consider analyzing mobile RAM for forensics artefact after executing the malware. In addition, fundamental features such as core logic, exploit binaries and native libraries were failed to be analyzed by the technique approach.

The research of [15,] made comparative analysis of static, dynamic and machine learning detection techniques used for malware detection on android applications. Market Centralization in mobile platform applications has made malware detection a very tedious task for most detection techniques even for machine learning and AI techniques. For instance, Google play central market for Android Google bouncer [50] to verify the legality of many apps. This security monitoring is insufficient because of millions of android developers associated with Google who's their Apps are not properly scrutinized before been permitted on the play store. This is similar to phone store and App store for Windows and Apple respectively. This made the security of mobile devices especially Android based not to be 100% absolute. The detection approached considered was based mainly on misuse and anomaly with application, untrusted data and system state as main objectives of analysis. None of the techniques provided 100% detection rate for mobile malware, based on the result obtained.

Anusha [16] dynamically compared malicious and mobile application behavior using mobile API for malware detection. The developed detection system (WMMD) detected obfuscated mobile malware which anti-virus software failed to detect. The detection approach used Finite State Automata (FSA) for malware code sampling and pattern checking. Using analysis by run-time, the model detected viruses both before and after packing. All the six (6) anti-virus software used, none was able to detect those mobile viruses after packing with UPX. It is then perceived that most anti-virus programs are not reactive to malware detection because they are signature based. This result was similar to [17] which is based on behavioural analysis for detecting malware on Android applications. The study mined 216 and 278 for normal and malicious Android applications separately. A variety of trained mathematical algorithms were applied on both the benign and malicious data set. Using correlational analysis, the research realized 97.16% detection accurateness.

Looking at advance malware artefacts in the mobile memory, [18] performed analysis of malware detection in mobile memory using memory forensics methodology. The research detected with 90% accuracy a self-replicating Trojan with 20% unclassified samples. The research observed that significant malware information can be mined from memory dump analysis of android mobile devices. Hidden codes waiting to explode at favorable conditions are easily exposed. The research how ever could not provide any investigation on searching malware attributes which are significant for forensic and security analysis. There has been elaborate work on detection techniques but there has not been a research that identifies and listed the limitations and strengths that existed in those techniques. By identifying both the limitations and the strength will help improve the efficiency of those techniques and enhance better detection of malware on Android devices.

3. METHODOLOGY

An organized appraisal with reference to the comprehensive malware detection framework outlined by [13] will be applied in this paper. The analysis targets to investigate recent studies on android malware detection techniques. The subsequent methodological subheadings expound on the research questions, criteria for paper selection, source of data and research keywords, summary of the research interpretation were presented in Table 1.

3.1 Research Questions (RQs)

Based on the review in section two in order to identify potential research gaps, the following research questions were formulated: What are the techniques used for detecting malware attacks on android devices? What are the key limitations when that exist when using those techniques? Are there some areas of strengths that are found in those techniques?

3.2 Criteria for Papers Selection

In selecting papers for this study, some standards were set. Only papers that matched and met

these criteria were selected. The selection criteria are as follows:

- The paper must be written in English Language.
- The paper must be published within three years (2016 to 2018) apart from frameworks
- It must discuss android malware and detection techniques using "OR" or "AND" operators. The operators here mean a paper can discuss one or both phrases.
- The selected paper must reflect the research experimental evidence on android malware detection in its content and results.

3.3 Justification for the Selection Criteria

English was selected as the language for the study to allow wide coverage for readers. Papers written in native languages are limited to a particular tribe of people with little impact. The choice to go for three years publications is to produce a research with current trend in malware and their detection techniques paradigms. Frameworks could be from any year since they form bedrocks for any study.

The operators "OR" and "AND" means that the selected paper must discuss either android malware, mobile malware detection techniques or both. Finally, the paper must show the experimental results with the approach used for such detection technique.

3.4 Source of Data

To get reliable data source, reputable academic research databases encompassing computational disciplines with high publication reputation. Such databases included but not limited to IEEE, Springer, ACM, Wiley and Inderscience. Conference papers and journals from those sources formed the primary source of data for the study. Papers from untrusted sources such as Wikipedia were not considered since they are rated unreliable.

However, data could be sourced from dependable blog such as SAN blog because of its reputation. As stated in the paper selection criteria above, basic search operators which includes 'OR' and 'AND's will be enforced so as to get request data within the defined research objectives. The main data which we intend to mine are the technical challenges, limitations, strengths of detection techniques used for android malware. Experimental results and the approach deployed by each research will not also be discarded.

4. ANDROID MALWARE DETECTION TECHNIQUES

For the purpose of this survey, the following malware detection techniques will be examined:

4.1 Dynamic Detection Techniques

Android based applications correlates with the device OS via system calls which makes it possible to monitor what transpire between them. Dynamic detection technique monitors android malware in a controlled environment at runtime by taking cognizance of the malware pointers which detection signatures can be modelled using them. It inspects malware interaction with mobile resources and services such as location, network, package, OS activities. For safety of the experimental equipment (physical device), it is recommended that the code execution be carried out in a cybernetic environment.

The study of [19] applied this detection technique on 4034 and 10024 malware and benign dataset respectively. Using ServiceMonitor approach, the random forest classification algorithm detected with 96% accuracy of malwares on those applications. Using k-fold validation and Markov chain, the classifier module was coached to extract the sample features. Information retrieved such as phone IMEI by malware was detected to be 67% accurate. Among the detected malware, 17% of the applications were observed to have attached their payload on the device for premium service rating. The mobile utilities such as CPU and Memory were observed to be infected with an overhead device performance of o.8% and 2% respectively.

Some malware remain dormant on the device after download and installation until an action is triggered. While some do so, others execute their payload at download, installation and runtime. This was demonstrated in [28]. Access authorisation habitually permitted by Android users during application download and installation creates a large space in the device

attack vector even though default permissions are always encountered during download and installation sessions. Malicious code attaches with the benign applications during those exercise. Critical monitoring at these stages is required for better security of mobile platforms.

4.2 Static Detection Techniques

Static detection technique for malware detection does not execute or run the malware code but solely depends on the malware abstraction features. For malware detection using this technique, the dependable features for detection come from the application byte code or its manifest file; unlike dynamic method which focuses on the system calls and application program padding. Android applications are in APK format or archive. This is usually in a zip package. All the Android files, folders and other resources are included. For meaning detection, reverse engineering is mostly applied to the apk files for features mining. When looking for relevant features' extraction, the manifest file "AndroidManifest.xml" is first to be considered. This manifest file contains permission vector features for access to installation, locations, battery optimization, phone state permissions. This study is similar to [38].

Ankita [20] used 103 and 97 malware and benign applications dataset respectively and detected malware on Nexus 5 with API level 19 detected high unauthorized permission attacks by malware. Reverse engineering was used as the experimental approach while applying Naïve Bayes, simple logic, RF 100, RF 10, J.48, Sequential minimal optimization and IBK algorithms. The xml parser extracted the permission request which generated binary features of the malware which was stored in Attribute Relation File Format (ARFF). Result provided 96.6% detection rate when random forest algorithm was used with 0.069% marginal different with the worst detection algorithm.

Malicious applications appearance cannot be easily seen until the code running the application is thoroughly inspected using trained systems. To analyse the raw data which is been processed as a Dalvik byte-code [26], concatenation of the opcode by disassembling the apk files constitutes a good practice. De-compilation of this file provides convolutional direction for extracting and analysing further Android applications files such as xml and other resource files. This approach is similar when using n-gram procedure in detecting malware. To ascertain and validate experimental result obtained in terms of detection accuracy, [27] used four different detection algorithms to detect android malware with large dataset of 5,560 malware sample. Bytecode dichotomize CFGs from the object node at initialization. Different detection obtained by the trained algorithms characterised the power of DSA when applied at the input and extraction layer of the model. Random forest algorithm obtained a detection accuracy of 97% higher than the rest.

4.3 Hybrid Detection Techniques

This technique combines both the features of dynamic and static techniques to provide a more robust detection result when analysing malware. Hybrid detection approach to malware detection involves basically training and detection phases which could be done by dynamic and static techniques respectively. This seems to give a better detection rate than dynamic and static techniques since the strengths in both methods are synergised. Using deep learning aspect of the artificial intelligence, [21] developed and trained a DroidDetector [51] model with some algorithms for android malware detection. The hybrid technique collected a total of 192 android malware and benign samples for training. The model yielded detection result accuracy of 96.60% with 0.0021% disparity amongst the algorithms used.

In some complicated cases were the malware sample is unknow, training and detection may not be done simultaneous to avoid features meddling. Hidden Markov Models proves to have high performance features when it comes to bisectional improvement in malware detection. Hybrid technique helps to make a comparative analysis of static and dynamic detection rate accuracy. Through semantic approach of this technique, [22] opcode and API call malware sample sequence were extracted using Hidden Markov Models. Recall, precision and specificity determined the threshold of the ROC curve. To define and establish the maliciousness and benignity of an application, Android Buster Sandbox was used as an analyser. Android malware detection by applying API call sequence could not however overcome the

problem of malware obfuscation. In additional, the observational sequence of the malware features does not produce a relational correspondence to the HMMs distinct states, this approach cannot generate the initial malware distribution state in the call graph and sequence respectively. Similar to this was the research of [23] which using API call graphs extracted malware smali files. Out of the 1,216 suspicious Android applications, a total of 1022 extraction was made. The resultant detection accuracy was found to be 96.12%. Though evasion attacks in Android malware was overcome by this approach, android poisoning attack could not be addressed using Machine learning. Focusing at API block calls, the study of [24] designed a detective tool (Droiddelver) with Deep Belief Network algorithm mined asemantic traces of both known and unknown malware. Boltzmann generated a restricted bipartite graph at the model input layer during malware probabilistic distribution. The malware print tack most at times in such a scenario provides indices of the smali program the malware might be intending to carve on the Android mobile kernel. The unzipping and decompiling of Android applications before extraction the API call layer requires a smali code to stand between the Dalvik VM and the App interface.

Some Android malware are basically designed to harvest information related to system calls, filesystems, mobile location and images captured by the device camera. Malicious app with this target makes the user of the device physically and informationally vulnerable. He can be easily tracked down and attacked or his system files could be exploited for financial gain or otherwise. This was demonstrated in the study of [25] with a relatively small Android malware dataset.

4.4 Permission-Based Detection Techniques

This technique involves detail analysis of the all the network traffic packets coming from the http server. The analysis of such packets indicates the nature of data which an application or device is sending to or receiving from an isolated server. some categories of mobile malware do not execute a visible and noticeable harm to the host device but only leak PII information such as list of apps, address, photos, IMEI and IMSI, location and mac address to malicious URL especially when insecure channel of communication is used. When this traffics are captured by software like Wireshark and are analysed, information leaked might be obtained.

It is observed that when a proper approach is deployed, sniffed data by such malware could be detected. When analysing malware with this method, features such as communication protocol and apk files [29] should be the target. In addition, a check can be performed to see if PII data is involved in the captured traffic. However, some Android malware do not generate network traffic to http url. This then becomes very difficult to detect malware conducting premium rate to contact numbers. This research is similar to [30], [31] and [32] respectively.

4.5 Emulation Based Detection

This technique requires providing a simulated ecosystem by an emulator for running of malware samples to separate them from the actual physical resources of the device. Simulation can be done on the Android OS or hardware. However, detection becomes much more tough when the execution of malware is done in the mobile real OS. This technique requires building of sandboxes and configuring virtual machines in a systematic and secure manner to avoid infecting other devices on the network. This technique is effective especially when the Dalvik file (.dex) [33] are properly monitored.

Malicious applications can be detected in the sandbox system by obtaining the dex file and converting into a form that can be understood by humans. Zero-day malware [34] and malware that escalate privileges [35] are effectively captured with this technique. However, some malware become aware of the virtual nature of the environment and tend to evade detection.

5. DISCUSSION

From the comparative analysis of this study, basic observations where made regarding the detection techniques. Use of small malware dataset was the one of the basic limitations observed in the investigated techniques. This hinders true evaluation of the detection efficiency since the sample size could not cut across different edges of malware families. With

such sample size, the technique might have seemed to perform proficiently but when implemented on a larger dataset, the opposite of the result is the case. This could produce lopsided ratio with little optimization.

Another pragmatic observation was the execution of malware in the sandbox without disabling the Android supervisory calls. This is clear that some malware would likely evade detection by detecting the presence of a sandbox environment by comparing different pieces of information from the system with strings such as "VMware" or "QEMU." Malware families or samples with the ability to test sys_vendor files will be able to detect the sandbox analysis environment especially when the analysis is performed with root privileges. According to the research of [39, 40], some of the malware can detect ***chroot*** by matching ***/proc/1/mountinfo*** with the PID of the malicious application information. This can be seen in the case when a known Linux malware known as Handofthief tried to evade IBM virtual machinery. When a virtual environment is perceived, dangerous malware can delay their execution while in the environment and wait at the suitable time to resurface, thus escaping detection. It therefore means that some of the detection techniques whose detection rate amounted to over 90% without disabling the Android supervisory call during malware analysis in the sandbox might be questionable.

Variation in detection rate by same algorithm in different detection scenarios is worrisome. For instance, in the study of [27], Naïve algorithm was implemented for detection using CFG approach. The detection rate was found to be 87.0%. In the same manner, 67.64% detection accuracy was witnessed in [28] when CFG was used. It is significant investigating into this wide variation.

6. EVALUATION

Analysis of android malware detection techniques is significant to building an efficient detection tool by applying both the strengths and limitations identified in all the studied approaches. The static detection methodology mined android metadata and other artefacts from Android malicious applications. Malicious interaction with the dex files at the Dalvik layer provides a dynamic approach to detection. A combination of static and dynamic techniques constitutes a hybrid method which provides a better detection accuracy. Other techniques studied are content and emulation-based detection. Worthy to note in this study is fact that both dynamic and static techniques can be approached in different ways by applying diverse set of trained algorithms (see Appendix). Dynamic technique can overcome strings of detection issues such as malware fitting and oligomorphism. The observed limitation is its susceptibility to transformation attacks, vulnerability to mimicry attacks and its inability to run on unrooted android devices.

When opcode sequence approached overcomes the need for hand-engineering. This could not however address the problem of malware encryption. Malware Obfuscation to escape detection is tranquil when this approach is used. Some elements of human interference could lead to unreliable outcome by using CFG. CFG is Susceptible to malware loading and replication, though highly scalable. Content-based approach provides speedy execution of large malware dataset. This approach achieves virtually no tangible detection result when apk file is not generating network traffic to http, TCP and UDP servers. As observed by [36, 37], there are a number of challenges confronting android mobile forensic including malware detection.

The detection challenges range from known to unknown, simple to sophistication. Transformation of the designed behavioural model could lead to malware obfuscation when the trained algorithm(s) and mutation approach are known by hackers or malware writers. Mutation and obfuscation make detection very difficult. Malware sandboxing is observed to be a delicate exercise. Little mistake to put the physical device at risk. From this research, it is clear that no detection technique developed and used by industries and individuals is 100% efficient in malware detection. As a result, occurrence of android malware has become a daily attack threat to the users. The comparative analysis of each detection technique studied in this research is summarized in Table 1.

Table 1

A Comparative Analysis of the Studied papers

S/No	Research	Detection technique	Detection Approach	Algorithms	Detection accuracy	Strength	limitations
19	(Salehi and Morteza, 2017)	Dynamic	ServiceMonitor	Random forest, Markov chain	86%	Overcame fitting problem	Susceptible to transformation and mimicry attacks
20	(Kapratar et al., 2017)	Static	Reverse engineering	Naïve Bayes, simple logic	96.6%	Overcomes issues of Bytecode Encryption	Fail to execute using Monkey Runner
21	(Yuan et al, 2016)	Hybrid	AI, Deep learning DroidDetector	Multi-layer perceptron, Naïve Bayes and Logistic regression	94.60%	High-level learning representation	Lopsided ratio, Little optimization
22	(Damodaran et al, 2017)	Hybrid	Hidden Markov Models (HMMs)	N/A	N/A	Known and unknown malware samples were detection	Problem of imbalance and obfuscation
24	(Hou et al., 2016)	Hybrid	Deep learning framework	Neural Network	92.66 %	Malware image recognition	Malware depth features were not extracted, assembly language is required
25	(Leeds et al., 2016)	Hybrid	Permissions data flow	Machine learning algorithm	80%	N/A	Sample was not streamed-lined
26	(McLaughlin et al, 2017)	Static	Opcode sequence	Convolutional neural network (CNN)	87%	The need for hand-engineered was removed	This could not address the problem of malware encryption
27	(Meng et al, 2016)	static	CFG and Bigram using DSA	Random Forest, Naïve Bayes, AdaBoost, Linear SVM	87.0%	Efficiency and scalability can be achieved with this approach	Susceptible to malware loading and replication
28	(Mahindu and Paramvir, 2017)	Dynamic	Machine learning and CFG	Naive Bayes, Decision Tree (J48), RF, Simple Logistic, and k-star	Simple Logistic 84.08%, Baiyes 67.64%	Overcame malware oligomorphism	Some samples evaded detection
29	(Malik and Rishabh, 2016)	Content based	CREDROID and Web of Trust	N/A	63%	Fast execution	Fails when APK is not generating network traffic.
33	(Costa and Hamidre,2017)	Emulation Based	Machine learning	Random forest Decision tree Nearest neighbours AdaBoost	Random forest 98.7%, Nearest neighbour 96.1% AdaBoost 99%	Detects Zero-day,privilege escalation malware	Cannot determine how malware processes the data affected, detected virtual environment
38	(Zhu et al., 2017)	Static	Support vector machine	Random forest	89.9%	Very fast and cost effective	Bias and variance in features detection

7. CONCLUSION

In this paper, a comparative examination of different Android mobile malware detection techniques was presented. The study was able to identify each of the limitations and strengths in each of the studies detection techniques through critical evaluation procedure. The results obtained from this study reinforce the assertion that detection approaches designed for Android malware do not produce 100% efficient detection accuracy. This segment of the research presents a critical evaluation of the reviewed papers. The rationale behind making this comparative analysis is to give a well-defined understanding on the strengths and weaknesses that were identified during the study in the selected detection techniques. A comparative survey on detection techniques focusing primarily on identifying Android malware detection techniques with their respective detection approaches, detection accuracy, and their corresponding strengths and limitations has not been explored before. For further research, we intend to carry out a study how to provide a security perimeter defence around Google bouncer for efficient Android applications review from third parties before uploading to the play store.

8. FUTURE WORK

There are many prospects and opportunities to further the work presented in this research as some of the future gaps have been identified.

First, hybrid and dynamic detection frameworks can be improved with better detection simulation using AI techniques and deep learning tools rather than just applying machine learning algorithms. As identified in section 6 (see Table 1), improvement on hybrid detection solutions can help increase efficiency in code coverage and sample streamline.

Furthermore, the integration of hybrid detection emulators and physical Android phones will help solve the problem of VM-ware detection and evasion by sophisticated malware such as polymorphic malware that can detect virtual environment. This future gap when closed will improve the accuracy of this solution.

Finally, further research should be carried out on investigating and evaluating detection parameters optimization when comparing Android detection techniques.

REFERENCES

[1] Li., F., Tegawendé, Y., Klein, D., Traon, L.: Understanding android app piggybacking: A systematic study of malicious code grafting. In: Tran. 2017. IEEE conference on Information Forensics and Security, vol. 12, pp. 1269--1284 (2017).

[2] Abdul, R., Daud, M., Mohamad, M.: Securing sensor to cloud ecosystem using internet of things (iot) security framework. In: Proc. 2016. ACM International Conference on Internet of things and Cloud Computing, pp. 79, ACM press (2016).

[3] Yan, C., Zahedi, F.: Individuals' Internet Security Perceptions and Behaviors: Polycontextual Contrasts Between the United States and China. Mis Quarterly, vol. 40, no. 1, pp. 205--222 (2016).

[4] Zhaoguo, W., Li, C., Yuan, Y., Xue, Y.: DroidChain: A novel Android malware detection method based on 9behaviour chains: Pervasive and Mobile Computing, vol.32, pp. 3--14 (2016).

[5] Songyang, W., Wang, P., Zhang, Y.: Effective detection of android malware based on the usage of data flow APIs and machine learning: Information and Software Technology, vol. 75, pp. 17--25 (2016).

[6] Andrea, S., Sgandurra, D., Dini, G., Martinelli, F.: Effective and efficient behaviour-based android malware detection and prevention: IEEE Transactions on Dependable and Secure Computing, (2016).

[7] Anurag, S., Kumar, D., Chanana, L.: An end to end security framework for service-oriented architecture: In Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), IEEE International Conference, pp. 475--480 (2017).

[8] Kaspersky, https://www.kaspersky.co.uk/resource-center/preemptive-safety/what-is-malware-and-how-to-protect-against-it.

[9] Muhammad, I., Vallina-Rodriguez, N., Seneviratne, M., Paxson, V.: An analysis of the privacy and security risks of android vpn permission-enabled apps. In: Proc. 2016. ACM conference on Internet Measurement, pp. 349--364 (2016).

[10] Wenrui, D., Liu, X., Li, Z., Zhang, K.: Evading android runtime analysis through detecting programmed interactions. In: Proc. 2016. ACM Conference on Security & Privacy in Wireless and Mobile Networks, pp. 159--164 (2016).

[11] Yasuyuki, T., Goto, A.: Analysis of malware download sites by focusing on time series variation of

malware. Journal of computational science, vol. 22, pp. 301--313 (2017).
[12] Android malware infection, https://www.lincolnshirelive.co.uk/news/local-news/check-your-apps-now-145-1876747.
[13] Hao, X.: Advanced Android Malware Detection Framework. MIT Press, Cambridge, MA (2013).
[14] Lok-Kwong, Y., Yin, H.: DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In: Symp. 2017. USENIX security symposium, pp. 569--584 (2017).
[15] Anastasia, S., Gamayunov, D.: Review of the mobile malware detection approaches: Parallel, Distributed and Network-Based Processing (PDP). In: Proc. 2015. IEEE 23rd Euro micro International Conference, pp. 600--603(2015).
[16] Anusha, D., Troia, F. D., Visaggio, C. A., Austin, T. H., Stamp, M.: A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 1--12 (2017).
[17] Shuaifu, D. Y., Liu, T., Wang, T., Zou, W.: Behavior-based malware detection on mobile phone," In Wireless Communications Networking and Mobile Computing. IEEE International Conference, pp. 1--4 (2016).
[18] Latika, S., Hofmann, M.: Dynamic behaviour analysis of android applications for malware detection. In IEEE International Conference on Intelligent Communication and Computational Techniques (ICCT), pp. 1--7 (2017).
[19] Majid, S., Amini, M.: Android Malware Detection using Markov Chain Model of Application Behaviors in Requesting System Services" arXiv preprint arXiv:1711.05731 (2017).
[20] Ankita, K., Troia, F. D., Stamp, M.: Static and Dynamic Analysis of Android Malware: In ICISSP, pp. 653--662 (2017).
[21] Zhenlong, Y., Lu, Y., Xue Y.: Droiddetector: android malware characterization and detection using deep learning: Tsinghua Science and Technology, pp. 114-123. IEEE Press, (2016).
[22] Stamp, M.' Anusha, D. F.: A comparison of static, dynamic, and hybrid analysis for malware detection," Journal of Computer Virology and Hacking Techniques, vo. 13, no. 1, pp. 1--12 (2017).
[23] Lingwei, C., Hou, S., Ye, Y., Chen, L.: An Adversarial Machine Learning Model Against Android Malware Evasion Attacks: In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data, pp. 43--55, Springer, Cham (2017).
[24] Shifu, H., Saas, A., Ye, Y., Chen, L.: Droiddelver: An android malware detection system using deep belief network based on api call blocks: In International Conference on Web-Age Information Management, pp. 54--66, Springer, Cham (2016).
[25] Matthew, L., Atkison, T.: A comparison of features for android malware detection. In: Proc. 2017. ACM South East Conference, pp. 63--68. ACM (2017).
[26] Niall, M. D., Rincon, B., Kang, S.: Yerima, P. Miller, S. Sezer and Y. Safaei, "Deep android malware detection. In: Proc. 2017. ACM Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 301--308 (2017).
[27] Guozhu, M., Yinxing, X., Zhengzi, X.: Semantic modelling of android malware for effective malware comprehension, detection, and classification. In: Proc. 2016. ACM Proceedings of the 25th International Symposium on Software Testing and Analysis, pp. 306-317 (2016).
[28] Arvind, M. and Singh, P.: Dynamic permissions-based Android malware detection using machine learning techniques. In: Proc. 2017. ACM Proceedings of the 10th Innovations in Software Engineering Conference, pp. 202-210 (2017).
[29] Jyoti, M., and Kaushal, R.: CREDROID: Android malware detection by network traffic analysis. In: Proc. 2016. Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing, pp. 28-36(2016).
[30] Santosh, J., H. Upadhyay, L., Lagos, N. S., Akkipeddi, and Guerra, V.: Machine Learning Approach for Malware Detection Using Random Forest Classifier on Process List Data Structure. In: Proc. 2018. ACM Proceedings of the 2nd International Conference on Information System and Data Mining, pp. 98-102(2018).
[31] Abhishek, B. R., Goswami, T., and Mukherjee, K.: A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission-based detection of Android malwares. International Journal of Machine Learning and Cybernetics, vol. 2, pp. 1--15(2018).
[32] Jungsoo, P., Chun, H., and Jung, S.: API and permission-based classification system for Android malware analysis. In Information Networking (ICOIN), 2018 IEEE International Conference, pp. 930--935(2018).
[33] Gabriele, C., and Aria, H.: Android Malware Detection Using Network Behavior Analysis and Machine Learning Classifiers, pp. 25—32(2017).
[34] Bhargav, A. R., Day, J. C., Steiner, D.: System and method for automated machine-learning, zero-day malware detection: U.S. Patent 9,292,688 issued (2016).
[35] Lee, H.T., Kim, D., Park, M., Cho, S.J.: Protecting data on android platform against privilege escalation attack. International Journal of Computer Mathematics, vol. 93, no. 2, pp. 401--414(2016).
[36] Mustafa, I., Al-Khateeb, H. M., Mansour, A., Ashawa, M., Hamisu, M.: Effective methods to detect metamorphic malware: a systematic review. International Journal of Electronic Security and Digital Forensics, vol. 10, no. 2, pp. 138--154 (2018).
[37] Milda, P., Dehghantanha, A., Epiphaniou, G.: Mobile phone forensics: an investigative framework based on user impulsivity and secure collaboration errors: In Contemporary Digital Forensic Investigations of Cloud and Mobile Applications, pp. 79--89 (2017).
[38] Hui-Juan, Z., Jiang, T., B., Shi, W., Cheng, L.: HEMD: a highly efficient random forest-based malware detection framework for Android. Neural Computing and Applications, pp.1--9 (2017).

[39] Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D.: Understanding Linux Malware: In: Symp. 2018. IEEE Symposium on Security & Privacy, pp. 7--15 (2018).
[40] Ezzati- Jivan, N., Dagenais, M.R.: Multi- scale navigation of large trace data: A survey. Concurrency and Computation: Practice and Experience, vol. 29, no.10, pp. 4068 (2018).
[41] Chien, E.: Motivations of recent android malware. Symantec Security Response, Culver City Press, California (2011).
[42] Baykara, M., Çolak, E.: A review of cloned mobile malware applications for android devices. In: Symp. 2018. IEEE 6th International Symposium on Digital Forensic and Security (ISDFS), pp. 1--5 (2018).
[43] Ahmadi, M., Sotgiu, A., Giacinto, G.: IntelliAV: Building an Effective On-Device Android Malware Detector. arXiv preprint arXiv:1802.01185 (2018).
[44] Wang, Z.Q., and Xinzhi, T.: Research on the Security Risks and the Preventive Strategies of Android Smart-Phones. In: Conf. 2019. the International Conference on Cyber Security Intelligence and Analytics, pp. 230-235. Springer, Cham (2019).
[45] Kabakus, A.T., and Dogru, I.A.: An in-depth analysis of Android malware using hybrid techniques. Digital Investigation, vol. 24, pp.25-33 (2018).
[46] Ma, Z., Ge, H., Liu, Y., Zhao, M. and Ma, J.: A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. IEEE Access, vol. 7, pp. 21235--21245 (2019).
[47] Tang, X., Song, T., Wang, K., Liang, A.: Fine Grained Access Control on Android Through Behavior Monitoring. InAdvances in Computer Communication and Computational Sciences. pp. 525-532. Springer, Singapore (2019).
[48] Chua, M., and Balachandran, V.: Effectiveness of Android Obfuscation on Evading Anti-malware. In: Proc. 2018. ACM Proceedings of the Eighth Conference on Data and Application Security and Privacy, pp. 143-145 (2018).
[49] Qamar, A., Karim, A., Chang, V.: Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems. 2019.
[50] Wang, H., Li, H. and Guo, Y.: Understanding the Evolution of Mobile App Ecosystems: A Longitudinal Measurement Study of Google Play. In: Conf. 2019. ACM World Wide Web Conference. pp. 1988—1999 (2019).
[51] Demetrio, L., Biggio, B., Lagorio, G., Roli, F.,Armando, A.: Explaining Vulnerabilities of Deep Learning to Adversarial Malware Binaries. arXiv preprint arXiv:1901.03583 (2019).