Mobile Malware Behavior through Opcode Analysis

Noor Azleen Anuar[1], Mohd Zaki Mas'ud[1], Nazrulazhar Bahaman[1] and Nor Azman Mat Ariff[1]

[1]Information Security, Digital Forensic and Computer Networking (INSFORNET), Faculty of Information Technology and Communication, Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

***Abstract*:** As the popularity of mobile devices are on the rise, millions of users are now exposed to mobile malware threats. Malware is known for its ability in causing damage to mobile devices. Attackers often use it as a way to use the resources available and for other cybercriminal benefits such stealing users' data, credentials and credit card number. Various detection techniques have been introduced in mitigating mobile malware, yet the malware author has its own method to overcome the detection method. This paper presents mobile malware analysis approaches through opcode analysis. Opcode analysis on mobile malware reveals the behavior of malicious application in the binary level. The comparison made between the numbers of opcode occurrence from a malicious application and benign shows several significant traits. These differences can be used in classifying the malicious and benign mobile application.

***Keywords*:** Mobile Malware, Mobile Malware Detection, Android Malware, Mobile Malware Behavior, Opcode**.**

1. Introduction

With the vast evolvement of technology and connectivity, community worldwide are depending on mobile devices to solve various daily tasks [1]. Mobile devices allow everyone to connect with each other at any time and at any place. Nonetheless, the existence of mobile devices had opened enough rooms for attackers to exploits mobile devices by injecting malware into the devices without the users' acknowledgement. Due to the intricacy in detecting mobile malware, mobile malware now possesses the ability of being unrecognized in the application store [2]which causes users to be tricked into installing the infected programs. Once the applications are installed in the users' devices, the infiltration and replication of malicious codes begins which could cause some serious problems if it stays undetected.

Malware which is also known as malicious software is known for its ability in causing damage to computers software and hardware. Attackers often use it as a way to use the resources available and for other cybercriminal benefits such stealing users' data, credentials and credit card number. This attack could happen when victims open an email or when they download an infected software. When the malicious link or attachments sent in the emails are clicked, malware could be installed without user's consent, ransomware attack could occur and freeze the entire system. This could also lead the sensitive information, financial and business information to be revealed or destroyed. The attacker could also gain and extract the victim's login credentials and account information.

According to the 2018 Internet Security Trend Report issued by Symantec Corporation [3], the discovery of latest mobile malware variations has significantly increased over the course of three years. In 2017, the discovery of new mobile malware had surged up to 54% compared to 2016. However, in 2018, the number of new mobile malware variations had shown a slight decreased of 10% as compared to those in 2017. Even though the discovery of new variants is lower compared to the previous years, the impact of mobile malware attacks is still significantly high. In this paper, all of the issues encountered by previous researchers in combatting the spread of mobile malware have been taken into consideration. Moreover, the mobile malware behavior is analyzed to identify a new approached to enhanced mobile malware detection. In order to improve accuracy and minimize false alarm rates, researchers have implemented several techniques to improve mobile malware detection.

The latest insights on methods used in the recent mobile malware detection system is included in this paper. In response to that, this paper is organized based on the research questions presented below.

RQ1: What is the taxonomy of mobile malware and its behavior?

RQ2: What are the features used in mobile malware detection?

2. Related Works

2.1. Types of Mobile Malware and its Behavior

A study had shown that with the vast connectivity through wireless network, malware is able to spread itself[4] and taint victims' smartphones and mobile devices [5]. Research by [6] also agreed that malware are often installed when victims accidentally clicked and downloaded malicious applications or links. Malware are classified into several classes [7].

Virus: Virus tends to infiltrate mobile devices and smartphones with the absence of user's consent. Once they managed to invade the system, the viruses then bind with any program files and starts to execute malicious function that had been programmed.

Worm: Worms are usually programmed to duplicate itself in the computer system. It then works by wrecking data and files available in the computer or mobile devices.

Trojan: Trojans are designed to extract banking information or credentials, and flood the computer's resources with Denial of Service (DoS) attacks.

Backdoor: programmers invent Backdoors so that it is easier for them to administer the programs remotely. However, when it is used for harmful purposes attackers are able to send malware, viruses, and even gain access to computer system to perform malicious activities.

Spyware: Spyware are used to keep an eye on the computer activities, which can also be utilized as a way to steal victims' login credentials.

Adware: Adware gives minimal threats to computers or mobile devices as it is only used to send ads that can sometimes be malicious.

Ransomware: Ransomware are a type of malicious programs that lock users' data and files. In order to unlock or

decrypt the files and data, victims will be asked to pay a hefty sum of money to the attackers.

Rootkit: On the other hand, rootkit is a malicious tool placed in a computer system so that uncertified personnel are able to enter the system. The attackers will then run files or even adjust the system configurations remotely.

Botnet: Attackers often used Botnets [8] as a medium to execute massive network attacks such DoS attacks to flood the resources [4].

Keylogger: Keylogger works by keeping all keystroke entry that you made. The recorded values are then used to obtain login credentials and other financial information.

Previous studies had shown that mobile malware tend to perform certain traits after they had infected any mobile devices [7]. The general mobile malware behaviors are classified as follows.

- Collect sensitive device and user credentials.
- Interact with Command and Control (C&C) server.
- Send premium rates SMS and spam.
- Optimizing search engines.
- Update and download package without users' consent.
- Causes an exhaustion of resources.

As the current mobile malware are continuously evolving [9], an enhanced mobile malware detection technique is needed to combat these issues.

2.2. Taxonomy of Mobile Malware Detection

Mobile malware analysis is critical in enlightening how malware operates in its environment. These several analysis approaches which can be categorized into three different approaches that is static analysis, dynamic analysis and lastly the hybrid analysis [10].

Static analysis mainly works by examining the APK code that have been disassembled without any form of execution [11]. In other words, [12] stated that the permissions, API calls, intent filters, network addresses, hardware components, Java code and binary values will be inspected and reviewed for any dubious form. This analysis approach requires lesser computational resources [13], making it the best approach to be adopted in mobile devices. However, there are some fall back that might occur when static analysis is implemented [14] as it is unable to perceive any code obfuscation, zero-day and polymorphic malware.

On the other hand, dynamic analysis is used to investigate the behavior of mobile malware [15] while it is running. Several features that are frequently inspected [16] are the system calls, network traffic, system components, and the user interactions. Dynamic analysis is highly reliable as it is able to detect any code obfuscation, polymorphism, and any modified runtime [17]. Nonetheless, there are few drawbacks when using this type of analysis as it requires excessive amount of resources and time and it tends to produce a significantly larger amount of false alarm rates [14].

Next, hybrid analysis is the combination of static and dynamic analysis. One of the common approaches implemented using the hybrid analysis is the heuristic technique. Heuristic technique is used in distinguishing between a mobile malware or a legitimate application by utilizing the rules that had been set by experts or Machine Learning algorithm [18]. This analysis is known for its ability in detecting zero-day and polymorphic malware, but in the end, it has its own limitations. According to [19], the usage of heuristic analysis might promote false positive results in which a legitimate application might be diagnosed as a mobile malware. Table 1 below summarized the analysis approach used by previous researchers.

Table 1. Analysis Approach

Author/Year	Analysis Approach	
	Static	Dynamic
Wang et al., 2019		/
Razak et al., 2019	/	
Yen & Sun, 2019	/	
P. et al., 2019		/
L. Zhang et al., 2019	/	
Papadopoulos et al., 2018		/
Kabakus & Dogru, 2018	/	/
Chen et al., 2018		/
Tong & Yan, 2017	/	/
Sheen et al., 2015		/
Idrees et al., 2017	/	
Zhao et al., 2018	/	
P. Zhang et al., 2018	/	
Raphael et al., 2014	/	
Yerima et al., 2013	/	
Feldman et al., 2014	/	
Kang, Yerima, Mclaughlin, & Sezer, 2016	/	
Bakhshinejad & Hamzeh, 2017	/	
Canfora, Lorenzo, Medvet, Mercaldo, & Visaggio, 2015	/	
Aminordin et al., 2018	/	
Abdelkhalki et al., 2020	/	
Wan Ahmad Ramzi et al., 2017		/
Obeidat 2017		/

2.3. Audit Data Source

Audit data source is generally an essential resource in constructing a good mobile malware detection model. Any malicious behavior left traces that can be used to distinguish malicious applications from benign mobile applications.

Audit data sources can be gathered from five Android framework layers that is the application layer, application framework, Android runtime, libraries and Linux kernel. Each of the framework layers provide a different set of audit data. As an example, when doing a collection on the Linux kernel interaction with network interface will yield the network traffic data. These types of data are usually obtained through dynamic analysis prior to its runtime. Data flow, system calls, network traffic, system components, and user interactions are the example of audit data source that can be collected through dynamic analysis. Collection of audit data source in dynamic analysis tends to produce a large number of logged data which can consume a lot of space in mobile devices. On the other hand, static analysis traces are not producing as large size of audit source data log, however some of the audit data source type can be obfuscated to hide the true behavior of the mobile malware. The audit data source that can be collected in static analysis are permission, manifest files, API, intent, and byte codes. Refer Table 2 for the audit data source extracted in previous researches.

This research explores mobile malware detection through opcode or bytecode analysis. As stated by [33] and [34] opcode which is an abbreviation of operation code act as a set of machine language instructions. These instructions are used in initiating certain operation in a computerized system [35]. Besides that, [24] agreed that each opcode is specified according to each data type and it is available in a byte codes

form. According to [36] during the initial state, a set of malware codes will be broken up into their respective binary forms. The binaries will then be analyzed and any hidden codes placed in the binary forms will be extracted [26]. Next, the binaries will then be decompiled to obtain the opcodes sequence. In a study done by [37], the researchers implemented the static analysis and uses deep learning to detect malware. The malware undergoes a process of unpacking where the opcode is extracted and later converted into binary image. A prediction on the binary image is done using ResNet in which the model achieves an accuracy of 93.5% when distinguishing malware and benign software. One of the benefits in using opcode is that features can be analyzed by only using raw data [38]. Refer Table 2 that is attached in Appendix A for the audit data source.

2.4. Dataset Used

In this section, all of the dataset that had been used by previous researchers will be listed and analyzed. Refer Table 3 for the datasets obtained in previous research.

Based on the table provided, it can be seen that most researchers are using the Drebin Project dataset. With over 5,560 malware samples, most researchers are using Drebin that offers a significant amount of malware samples and as a mean in using a standardized dataset. However, this research uses the Android Malware Dataset provided by ArgusLab [39] as it carries a larger number of the latest mobile malware. Table 3 that is in Appendix B shows the dataset used in previous research.

3. Methodology

In this research, a static analysis will be carried out on each mobile malware sample in order to reveal its behavior. In static analysis, the selected .apk files are decompiled and related features are extracted and examined. This analysis is widely used in inspecting permissions, API calls, and to determine the code structures and components involved in a certain .apk files. When the .apk files is decompiled, several archives are found residing in it such as the META-INF directory, lib, res, assets, AndroidManifest.xml, classes.dex, and resources.arsc. The AndroidManifest.xml and classes.dex are commonly used in static analysis as it can reveal the real intention of any suspicious applications. Figure 1 shows the proposed approach for this research. Figure 1 shows the steps involved to determine the mobile malware behaviors.

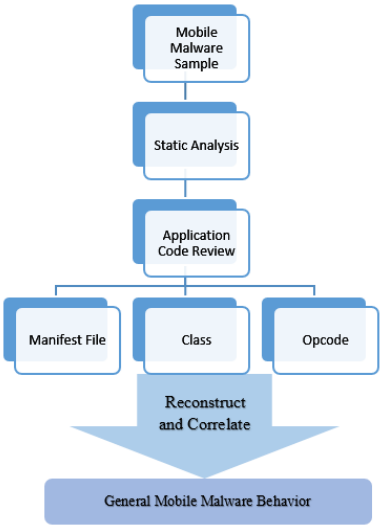

Figure 1. Research Approach

3.1. Experimental Setup

In static analysis, the framework involved are divided into five different phases. The phases include isolation of environment setup, extraction of Android package, disassembling of codes, opcode and class reviews, and reconstruction and correlation analysis. Figure 2 shows the graphical representation of the phases involved in this experiment

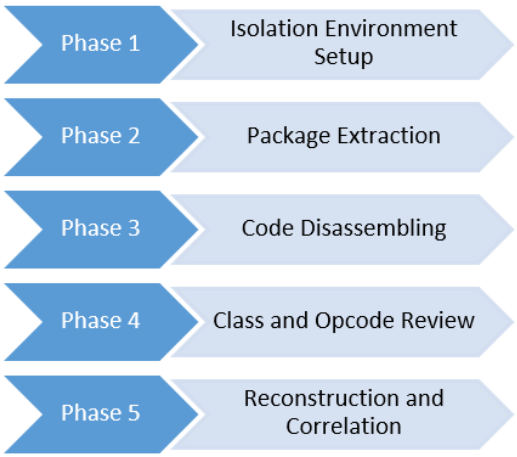

Figure 2. Research Phases

In the first phase, a controlled environment is set up using VMware workstation in order to analyze mobile malware without risking the host PC from being infected. Secondly, the .apk files are extracted in order to obtain the classes and manifest file. Manifest file stores information regarding the configuration, activity and permissions invoked by the application while classes file contains all of the Java codes used. Next, the classes.dex file are decompiled into a Java class file also known as .jar. Any malicious request can be seen by analyzing the codes and methods contained in Java class. Some of the common malicious activities are requesting root permission, steal sensitive information such as IMEI number and country number, and sending and receiving commands from a C&C server. Last but not least, in the fifth phase, the results obtained in static analysis are used to depict chain of malicious activities. The final step is essential as it helps researcher to plot the patterns carried out by the malware during an attack.

Based on the results of the research findings, a proposed solution was developed from the results of the data collection to solve the problem. The implementation of the solution is presented in the following sections.

4. Results and Discussion

4.1. Results and Analysis

An initial experiment was done by examining a total of 500 benign and malicious samples. 250 malicious .apk were downloaded from the Android Malware Dataset shared by ArgusLab [39] and 250 benign software was downloaded from the Google PlayStore. The benign samples were then uploaded to VirusTotal [42] to ensure that they do not contain any viruses.

4.2. Malicious Traces for AndroRat

Figure 3, in Appendix C, shows the opcode sequence involved in an effort to obtain root access. There are several ‘invoke-static’ and ‘invoke-direct’ operation involved in the process shown which is used to invoke a virtual method. As

seen in the argument field, it can be seen that one of the commands invoked are 'su' which is associated in gaining root access. Besides that, this piece of malware is also attempting to access system files based on the filename requested which is '#!/system/bin/sh'. The activities carried out are suspicious and could cause major problems.

4.3. Malicious Traces for BankBot

Figure 4, in Appendix C, shows the opcode sequence for the connection made by the program to an external server. Based on the connections made, several sensitive information had been sent to the C&C server. According to the arguments shown in process 1, the IP address of the C&C server is 'http://113.10.137.171'. In the second and third process, it can be seen that the device information such as phone number and IMSI number are being requested to be send over to the remote server. All of the device information requested are sensitive and should not be shared as attackers could use it to clone the affected devices.

4.4. Malicious Traces for Airpush

In the first and second process shown in the Figure 5, in Appendix C, it can be seen that there is an argument requesting for API key and IMEI number. This particular piece of device information is essential and should be kept as a secret as it can be used to clone the device. The cloned device can be used to carry out any malicious activity without the victim's knowledge.

4.5. Discussion

One of the ways to represent the behavior obtained is by plotting a bar graph of the frequency of importance against the opcode sequence. The frequency of importance for each opcode sequence is measured using Term Frequency-Inverse Document Frequency (TF-IDF) that is also used by [40] and [41]. TF-IDF is used to measure the degree of importance for each phrase in a document. The term frequency evaluates the number of opcode occurrence in each application whereas the inverse document frequency calculates the level of importance for each opcode value. The orange colored bars are used to represent the frequency of importance from malicious mobile applications whereas the blue colored bars represent the benign mobile applications. Each opcode is represented using its own unique value that is located at the X-axis. In contrast, the Y-axis represents the TF-IDF for each opcode in the selected mobile applications. Considering that the blue chart represents the benign mobile applications, it can be seen that the infected applications are producing a larger number of opcodes compared to those of benign applications. Figure 6 and Figure 7 shows the TF-IDF of each opcode sequence in malicious and benign applications respectively whereas Figure 8, Appendix D, shows the comparison of TF-IDF values in both applications.

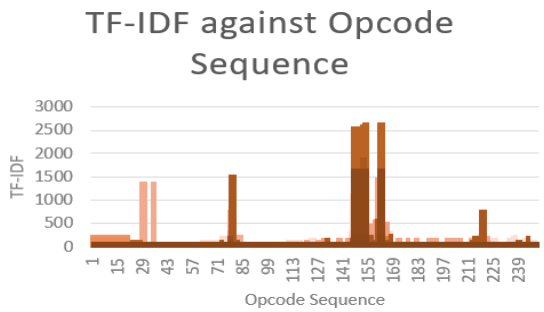

Figure 6. Frequency of Opcode Occurrence in Malicious Applications

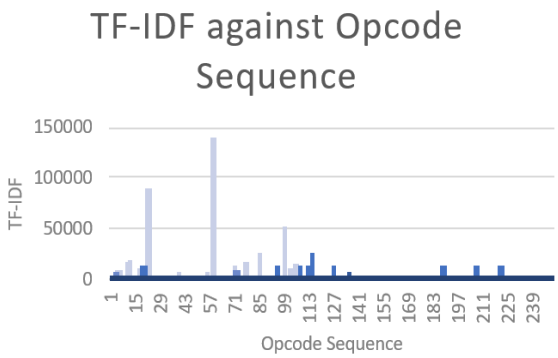

Figure 7. Frequency of Opcode Occurrence in Benign Applications

Based on the graph shown, it can be observed that the highest number of opcode occurrence in malicious applications are the aput-byte, iget-byte, sget-short, sput-short, iget-short, shl-int/lit8, mul-double/2addr, aput-char, mul-int/2addr, and add-int. Whereas in the benign applications, the highest opcode occurrence are const-string/jumbo, not-int, and-int, mul-double, move-wide/from16, if-ltz, sget, and-long/2addr, mul-int, and invoke-static/range.

According to the findings obtained from the bar chart, it can be seen that some of the opcode invoked have its own importance in both malicious and benign samples. However, those belonging to malicious applications shows a higher level of importance. Table 4 shows TF-IDF value for each opcode sequence occurring in malicious and benign applications

Table 4. Percentage of Opcode Occurrence in Malicious and Benign Applications

Opcode (Malicious)	TF-IDF	Opcode (Benign)	TF-IDF
aput-byte	5044.546	const-string/jumbo	431481.9
iget-byte	4862.899	not-int	125414.3
sget-short	4121.822	and-int	29259.81
sput-short	3040.008	mul-double	28041.54
iget-short	2446.469	move-wide/from16	25152.73
shl-int/lit8	2373.745	if-ltz	23255.92
mul-double/2addr	2281.891	sget	22057.04
aput-char	2141.033	and-long/2addr	19477.04
mul-int/2addr	2092.251	mul-int	19165.74
add-int	1842.727	invoke-static/range	18778.57

Based on the table depicted, it can be observed that the ten highest opcode sequence in malicious applications are linked to suspicious activities. The opcode listed are usually used to carry out the operation that had been determined on the selected index.

5. Conclusion

In this paper, we analyzed the behavior of mobile malware through opcode analysis. Based on the result obtained, it can be seen that there is a huge difference in the frequency of occurrence for opcode in mobile malware and benign applications. Other than that, the behavior of malicious

applications is observed and the opcode extracted are mapped to its suspicious activities. In future, we would like to propose a static analysis based on the ensemble of n-opcode features in detecting mobile malware.

6. Acknowledgement

The authors are grateful to InforsNet Research Group, Faculty of Information and Communication Technology (FTMK) of Universiti Teknikal Malaysia Melaka (UTeM) for the support and special acknowledgement to Ministry of Higher Education Malaysia (MoHE) for providing financial support through the Fundamental Research Grant Scheme (FRGS/2018/FTMK-CACT/F00391).

References

[1] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for android," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 15–25, 2011, doi: 10.1145/2046614.2046619.

[2] L. Tenenboim-Chekina *et al.*, "Detecting application update attack on mobile devices through network featur," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 91–92, doi: 10.1109/INFCOMW.2013.6970755.

[3] S. Corporation, "Executive Summary | 2018 Internet Security Threat Report," 2019. https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-executive-summary-en.pdf (accessed May 24, 2019).

[4] W. Y. Wan Ahmad Ramzi, M. A. Faizal, M. D. Rudy Fadhlee, and M. S. Nur Hidayah, "Revealing influenced selected feature for P2P botnet detection," *Int. J. Commun. Networks Inf. Secur.*, vol. 9, no. 3, pp. 500–506, 2017.

[5] S. Peng, S. Yu, and A. Yang, "Smartphone malware and its propagation modeling: A survey," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 2, pp. 925–941, 2014, doi: 10.1109/SURV.2013.070813.00214.

[6] Futai Zou, Tianqi Wan, Siyu Zhang, and Li Pan, "A survey of android mobile platform security," in *10th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2014)*, 2014, pp. 520–527, doi: 10.1049/ic.2014.0155.

[7] A. Skovoroda and D. Gamayunov, "Review of the Mobile Malware Detection Approaches," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Mar. 2015, pp. 600–603, doi: 10.1109/PDP.2015.54.

[8] A. A. Obeidat, "Hybrid approach for botnet detection using k-means and k-medoids with Hopfield neural network," *Int. J. Commun. Networks Inf. Secur.*, vol. 9, no. 3, pp. 305–313, 2017.

[9] G. Canfora, F. Mercaldo, E. Medvet, and C. A. Visaggio, "Detecting Android malware using sequences of system calls," *3rd Int. Work. Softw. Dev. Lifecycle Mobile, DeMobile 2015 - Proc.*, pp. 13–20, 2015, doi: 10.1145/2804345.2804349.

[10] Y. Salah, I. Hamed, S. Nabil, A. Abdulkader, and M. M. Mostafa, "Mobile Malware Detection : A Survey," *Int. J. Comput. Sci. Inf. Secur.*, vol. 17, no. 1, 2019.

[11] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic," *J. Netw. Comput. Appl.*, vol. 133, pp. 15–25, May 2019, doi: 10.1016/j.jnca.2018.12.014.

[12] A. Feizollah, N. B. Anuar, R. Salleh, G. Suarez-Tangil, and S. Furnell, "AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection," *Comput. Secur.*, vol. 65, pp. 121–134, Mar. 2017, doi: 10.1016/j.cose.2016.11.007.

[13] V. P., A. Zemmari, and M. Conti, "A machine learning based approach to detect malicious android apps using discriminant system calls," *Futur. Gener. Comput. Syst.*, vol. 94, pp. 333–350, May 2019, doi: 10.1016/j.future.2018.11.021.

[14] J. Abawajy, S. Huda, S. Sharmeen, M. M. Hassan, and A. Almogren, "Identifying cyber threats to mobile-IoT applications in edge computing paradigm," *Futur. Gener. Comput. Syst.*, vol. 89, pp. 525–538, Dec. 2018, doi: 10.1016/j.future.2018.06.053.

[15] Y.-S. Yen and H.-M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectron. Reliab.*, vol. 93, no. December 2018, pp. 109–114, Feb. 2019, doi: 10.1016/j.microrel.2019.01.007.

[16] L. Zhang, V. L. L. Thing, and Y. Cheng, "A scalable and extensible framework for android malware detection and family attribution," *Comput. Secur.*, vol. 80, pp. 120–133, Jan. 2019, doi: 10.1016/j.cose.2018.10.001.

[17] H. Papadopoulos, N. Georgiou, C. Eliades, and A. Konstantinidis, "Android malware detection with unbiased confidence guarantees," *Neurocomputing*, vol. 280, pp. 3–12, Mar. 2018, doi: 10.1016/j.neucom.2017.08.072.

[18] A. T. Kabakus and I. A. Dogru, "An in-depth analysis of Android malware using hybrid techniques," *Digit. Investig.*, vol. 24, pp. 25–33, Mar. 2018, doi: 10.1016/j.diin.2018.01.001.

[19] Z. Chen *et al.*, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Inf. Sci. (Ny).*, vol. 433–434, pp. 346–364, Apr. 2018, doi: 10.1016/j.ins.2017.04.044.

[20] M. F. A. Razak, N. B. Anuar, R. Salleh, A. Firdaus, M. Faiz, and H. S. Alamri, "'Less Give More': Evaluate and zoning Android applications," *Measurement*, vol. 133, pp. 396–411, Feb. 2019, doi: 10.1016/j.measurement.2018.10.034.

[21] F. Tong and Z. Yan, "A hybrid approach of mobile malware detection in Android," *J. Parallel Distrib. Comput.*, vol. 103, pp. 22–31, May 2017, doi: 10.1016/j.jpdc.2016.10.012.

[22] S. Sheen, R. Anitha, and V. Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach," *Neurocomputing*, vol. 151, no. P2, pp. 905–912, Mar. 2015, doi: 10.1016/j.neucom.2014.10.004.

[23] F. Idrees, M. Rajarajan, M. Conti, T. M. Chen, and Y. Rahulamathavan, "PIndroid: A novel Android malware detection system using ensemble learning

methods," *Comput. Secur.*, vol. 68, pp. 36–46, Jul. 2017, doi: 10.1016/j.cose.2017.03.011.

[24] L. Zhao, D. Li, G. Zheng, and W. Shi, "Deep Neural Network Based on Android Mobile Malware Detection System Using Opcode Sequences," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, Oct. 2018, vol. 2019-Octob, pp. 1141–1147, doi: 10.1109/ICCT.2018.8600052.

[25] P. Zhang, S. Cheng, S. Lou, and F. Jiang, "A Novel Android Malware Detection Approach Using Operand Sequences," in *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, Oct. 2018, no. Ssi C, pp. 1–5, doi: 10.1109/SSIC.2018.8556755.

[26] R. Raphael, V. P., and B. Omman, "X-ANOVA and X-Utest features for Android malware analysis," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2014, pp. 1643–1649, doi: 10.1109/ICACCI.2014.6968608.

[27] S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2013, pp. 121–128, doi: 10.1109/AINA.2013.88.

[28] S. Feldman, D. Stadther, and B. Wang, "Manilyzer: Automated Android Malware Detection through Manifest Analysis," in *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, Oct. 2014, pp. 767–772, doi: 10.1109/MASS.2014.65.

[29] B. Kang, S. Y. Yerima, K. Mclaughlin, and S. Sezer, "N-opcode analysis for android malware classification and categorization," in *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, Jun. 2016, pp. 1–7, doi: 10.1109/CyberSecPODS.2016.7502343.

[30] N. Bakhshinejad and A. Hamzeh, "A new compression based method for android malware detection using opcodes," in *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, Oct. 2017, vol. 2018-Janua, pp. 256–261, doi: 10.1109/AISP.2017.8324092.

[31] G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Effectiveness of Opcode ngrams for Detection of Multi Family Android Malware," in *2015 10th International Conference on Availability, Reliability and Security*, Aug. 2015, pp. 333–340, doi: 10.1109/ARES.2015.57.

[32] A. Aminordin, F. MA, R. Y.-J. of T. and Applied, and U. 2018, "Android Malware Classification Base on Application Category Using Static Code Analysis," *Jatit.Org*, vol. 96, no. 20, 2018, [Online]. Available: http://www.jatit.org/volumes/Vol96No20/16Vol96No20.pdf.

[33] P. Faruki *et al.*, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015, doi: 10.1109/COMST.2014.2386139.

[34] S. Rezaei, A. Afraz, F. Rezaei, and M. R. Shamani, "Malware detection using opcodes statistical features," in *2016 8th International Symposium on Telecommunications (IST)*, Sep. 2016, pp. 151–155, doi: 10.1109/ISTEL.2016.7881800.

[35] P. Feng, J. Ma, C. Sun, X. Xu, and Y. Ma, "A Novel Dynamic Android Malware Detection System With Ensemble Learning," *IEEE Access*, vol. 6, no. c, pp. 30996–31011, 2018, doi: 10.1109/ACCESS.2018.2844349.

[36] M. Zaki, S. Shahib, M. F. Abdollah, S. R. Selamat, and C. Y. Huoy, "A Comparative Study on Feature Selection Method for N-gram Mobile Malware Detection," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 1–7, 2017, doi: 10.6633/IJNS.201709.19(5).10.

[37] J. El Abdelkhalki, M. Ben Ahmed, and B. A. Abdelhakim, "Image Malware Detection using Deep Learning," *Int. J. Commun. Networks Inf. Secur.*, vol. 12, no. 2, pp. 180–189, 2020, [Online]. Available: https://search.proquest.com/openview/fb31622f0f2b391af47271dfcdd44444/1?pq-origsite=gscholar&cbl=52057.

[38] A. Yewale and M. Singh, "Malware detection based on opcode frequency," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, May 2016, no. 978, pp. 646–649, doi: 10.1109/ICACCCT.2016.7831719.

[39] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep Ground Truth Analysis of Current Android Malware," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10327 LNCS, 2017, pp. 252–276.

[40] M. Fan *et al.*, "Frequent Subgraph Based Familial Classification of Android Malware," *Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE*, vol. 0, pp. 24–35, 2016, doi: 10.1109/ISSRE.2016.14.

[41] W. Li, J. Ge, and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach," *Proc. - 2nd IEEE Int. Conf. Cyber Secur. Cloud Comput. CSCloud 2015 - IEEE Int. Symp. Smart Cloud, IEEE SSC 2015*, pp. 464–469, 2016, doi: 10.1109/CSCloud.2015.50.

[42] VirusTotal, "VirusTotal," 2019. [Online]. Available: https://virustotal.com/gui/home/upload. [Accessed 22 September 2019].

Appendix A

Table 2. Audit Data Source

Author/Year	Static Features						Dynamic Features			
	Permission	Java code	Intent	API calls	Strings	Hardware component	System call	Network traffic	System component	User interaction
Wang et al., 2019								/		
Razak et al., 2019	/									
Yen & Sun, 2019		/								
P. et al., 2019							/			
L. Zhang et al., 2019	/				/					
Papadopoulos et al., 2018	/					/		/		
Kabakus & Dogru, 2018	/			/				/		
Chen et al., 2018	/									
Tong & Yan, 2017								/		
Sheen et al., 2015							/			
Idrees et al., 2017	/			/						
Zhao et al., 2018	/		/							
P. Zhang et al., 2018		/								
Raphael et al., 2014				/						
Yerima et al., 2013	/	/								
Feldman et al., 2014				/						
Kang, Yerima, Mclaughlin, & Sezer, 2016	/									
Bakhshinejad & Hamzeh, 2017		/								
Canfora, Lorenzo, Medvet, Mercaldo, & Visaggio, 2015		/								
Aminordin et al., 2018	/			/						
Abdelkhalki et al., 2020		/								
Wan Ahmad Ramzi et al., 2017								/		
Obeidat 2017								/		

Appendix B

Table 3. Dataset Used in Previous Research

Malicious Software		**Benign Software**		**Used in**
Collected from	**Amount**	**Collected from**	**Amount**	
Drebin Project	5560	Downloaded from multiple app market by app crawler	8321	Wang et al., 2019
Drebin Project	5000	AndroZoo	5000	Razak et al., 2019
Apk files	720	Apk files	720	Yen & Sun, 2019
Dataset 1: Drebin Project Dataset 2: Koodous, user agencies and collection of ransomwares	2520 Not mentioned	Dataset 1: Google Playstore, Chines market, Koodous, and third-party Android market Dataset 2: Similar with the first dataset	3130 3130	P. et al., 2019
Genome Project Drebin Project In-the-wild	928 5560 33259	In-the-wild	37224	L. Zhang et al., 2019
AndroZoo	1866	AndroZoo	4816	Papadopoulos et al., 2018
ASHISHB Genome Project Drebin Project Contagio Mobile **Total**	58 728 1953 70 **2809**	Google PlayStore	2999	Kabakus & Dogru, 2018
Dataset 1: Drebin Project Dataset 2: Contagio Dump, AndroTotal, and AndroMalShare	5553 4317	Dataset 1: PlayDrone archive Dataset 2: PlayDrone archive	5818 587	Chen et al., 2018
Drebin project	5560	A collection of apps installed from Hiapk market, 91Play market, Baidu market and Qihu360 market	6178	Tong & Yan, 2017
Malware Genome Project Dataset 1: Dataset 2:	 300 643	Obtained from Ministry of Education Key Laboratory of Intelligent Network and Network Security, Xi'an JiaoTong University, China Dataset 1: Dataset 2:	 130 267	Sheen et al., 2015
Malware Genome Project	1073	Not mentioned	904	Idrees et al., 2017
Contagio Dump Drebin Project Malware Genome VirusTotal AndroZoo MalShare VirusShare	60 100 1000 70 20 25 25	Google Playstore	445	Zhao et al., 2018
Drebin Project	5560	Obtained from various Chinese market	10000	P. Zhang et al., 2018
Drebin Project Contagio Mobile Malware	5560 361	Google Playstore	59000	Raphael et al., 2014
Contagio Dump	Not mentioned	Google Playstore	Not mentioned	Yerima et al., 2013
Malware Genome Project	1000	Android markets	1000	Feldman et al., 2014
Malware Genome Project	1260	Google Playstore	1260	Kang, Yerima, Mclaughlin, & Sezer, 2016
Drebin Project	3000	Google Playstore	3000	Bakhshinejad & Hamzeh, 2017
Drebin Project	5560	Google Playstore	5560	Canfora, Lorenzo, Medvet, Mercaldo, & Visaggio, 2015
AndroZoo	4434	Google Playstore	5823	Aminordin et al., 2018
Not mentioned	3012	Not mentioned	3000	Abdelkhalki et al., 2020

Appendix C

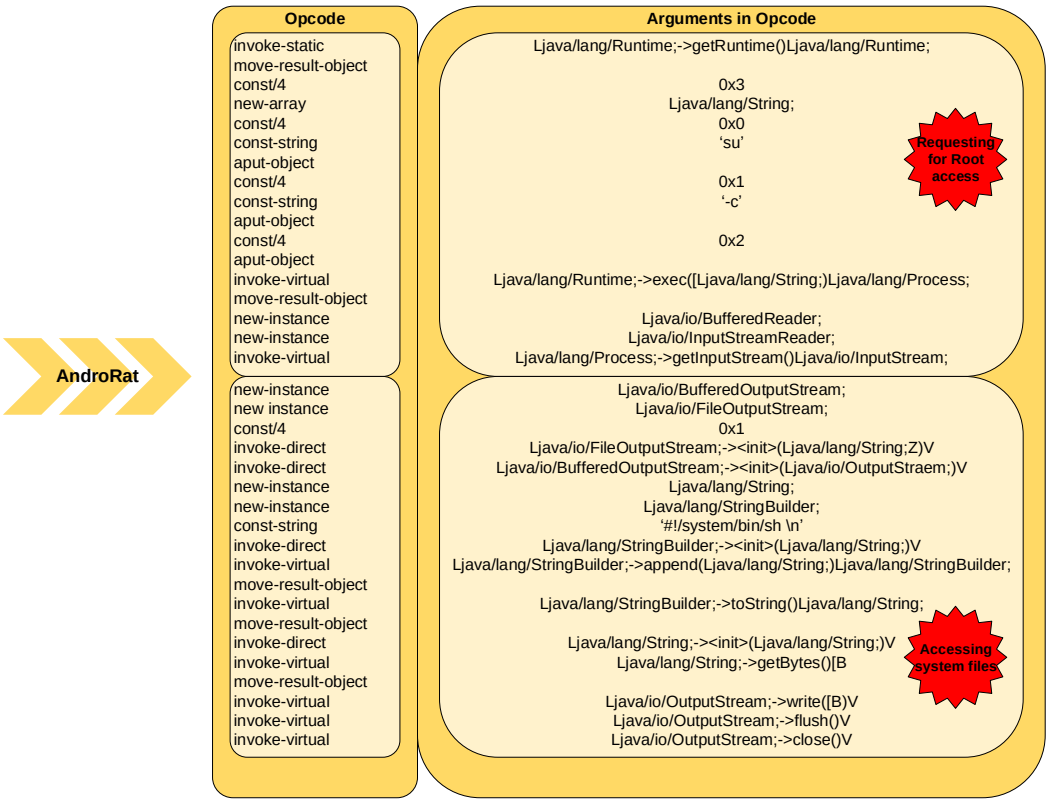

Figure 3. Malicious Traits Found in AndroRat

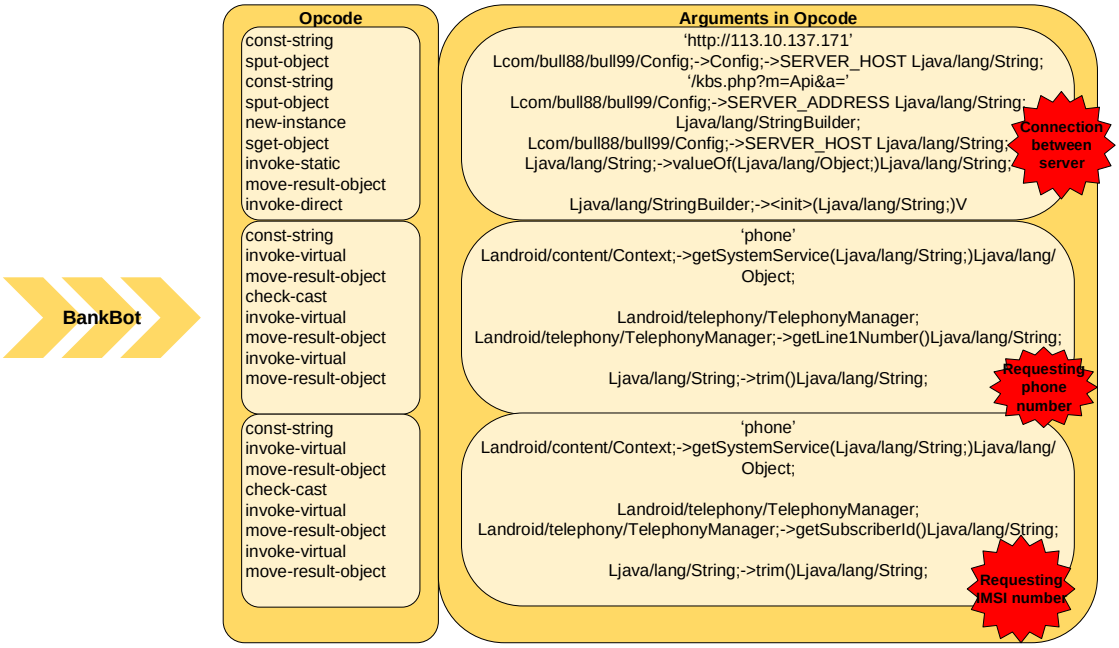

Figure 4. Malicious Traits Found in BankBot

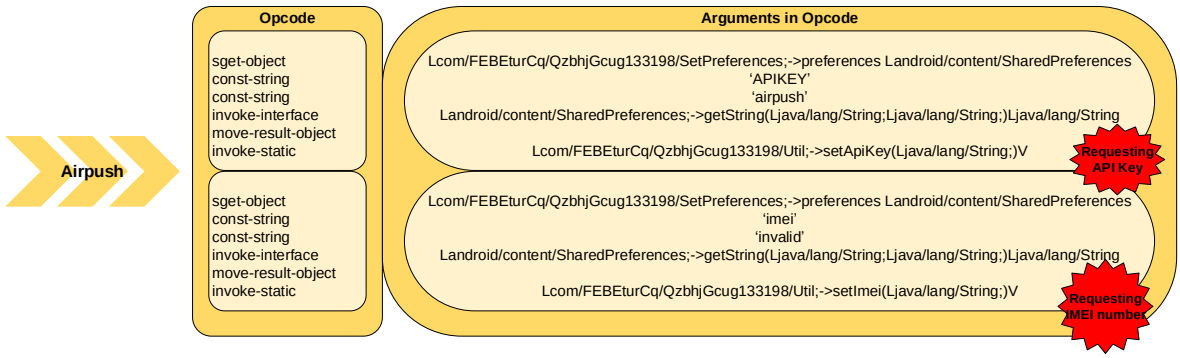

Figure 5. Malicious Traits Found in Airpush

Appendix D

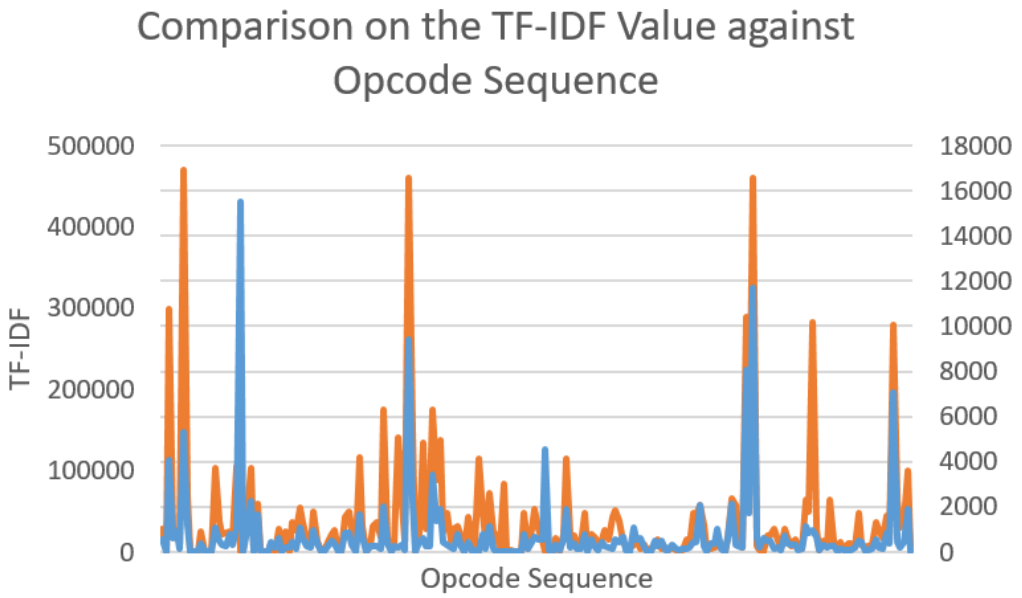

Figure 8. Comparison on TF-IDF for Opcode Occurrence in Malicious and Benign Applications