Advances in Computing and Intelligent System, Vol. 1 No. 1 (2019) p. 1-7

ACIS

FAZ Publishing

Advances in Computing and Intelligent System

Journal homepage : www.fazpublishing.com/acis

e-ISSN : 2682-7425

Android Trojan Detection Based on Dynamic Analysis

Nurul Izzati Aminuddin*, Zubaile Abdullah

Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia

* Corresponding Author

izzatiaminuddin1903@gmail.com, zubaile@gmail.com

Received 20 November 2018; Accepted 20 January 2019; Available online 10 March 2019

Abstract: Nowadays, mobile devices such as smartphones are no longer limited for phone calling or sending messages but it also being used for web browsing, social networking, meeting scheduling, gaming, file downloading and online banking transaction. To certain extend smartphones user kept confidential information such as contacts, bank account number, username and password for online banking, credit card number, memorable and private pictures in these devices. There are various type of smartphone operating system such as Apple iOS, Google Android, and Microsoft's Windows Phone. The most popular operating system for smartphones are Android. This become as one of the main target for attackers to spread the mobile malware especially Trojans virus. Based on that, this research will use dynamic analysis to detect their features. So, the parameter such as system call will be investigate in this project. Dataset that will use are Trojan dataset and benign application. The Trojan dataset can be download by Drebin and benign from Google Play store in apk version. The proposed of this research are to detect android Trojan based on dynamic analysis. The information gain algorithm is used to select the most significant system call, then the classification algorithm Random Forest, J48 and NaiveBayes used to classify the Android apps as Trojan or benign apps. The experimental results show that Random Forest Algorithm achieved highest accuracy of 81.2 % with lowest false positive rate of 0.188.

Keywords: dynamic analysis, malware, trojan

1. INTRODUCTION

Nowadays, mobile devices such as smartphones are no longer limited for phone calling or sending messages but it also being used for web browsing, social networking, meeting scheduling, gaming, file downloading and online banking transaction. To certain extend smartphones user kept confidential information such as contacts, bank account number, username and password for online banking, credit card number, memorable and private pictures in these devices [1]. There are various type of smartphone operating system such as Apple iOS, Google Android, and Microsoft's Windows Phone. Among others operating system for smartphone Android had dominated the market. According to Gatner February 2017 operating system that is installed in millions of devices and accounted for more than 82 percent of the total market in the fourth quarter are Android. This become as one of the main target for attackers to spread the mobile malware especially Trojans virus.

This incident happened resulted from the openness of Android OS environment which allows an applications downloading and installation into Android application provider from various sourced. Users are supposed to download and install Android application (app) from Google official applications market named Google Play Store, however users can also can download an application from other third party market. These unofficial markets provide free or non-paying apps and games which if users download from Google Play Store, they have to pay for those apps.

**Corresponding author: izzatiaminuddin1903@gmail.com*

Table 1 : Worldwide Smartphone Sales to End Users by Operating System in 4Q16 (Thousands of Units) (Adapted from Gatner February 2017)

Operating System	**4Q16 Units**	**4Q16 Market Share (%)**	**4Q15 Units**	**4Q15 Market Share (%)**
Android	352,669.9	81.7	325,394.4	80.7
iOS	77,038.9	17.9	71,525.9	17.7
Windows	1,092.2	0.3	4,395.0	1.1
BlackBerry	207.9	0.0	906.9	0.2
Other OS	530.4	0.1	887.3	0.2
Total	**431,539.3**	**100.0**	**403,109.4**	**100.0**

This attracts users to the unofficial market and possibly expose smartphone users to download and install malicious application (malware) which camouflaged as benign software from these markets. Furthermore, anybody can upload any type of app to this unofficial app market which is low in security implementation thus it is easier to malware author to upload their malware app to this market. The malware apps payload of smartphone come in various forms, such as viruses, Trojans, worms and mobile botnet

Among these malware, Trojan is a type of malware that is often disguised as legitimate software. Lot of information about malware such as virus, Trojan, worm, spyware and botnet can be found on the internet. These malware had their own characteristic but the public usually term all the malware as virus. Thus, it is critical to differentiate each malware to ensure the detection and response techniques are suitable based on malware characteristics. A paper by B. Torregrosa [6] had comprehensively defined and categorized these malware.

Malware detection techniques on the Android platform are similar to techniques used on any platform. An important concern on the growing Android platform is malware detection. There are two common technique for malware detection which is by static analysis and dynamic analysis. Static analysis involves no dynamic execution of the software under test and can detect possible defects in an early stage, before running the program. This may have approached automated tools can provide a false sense of security that everything is being addressed and vulnerabilities in the runtime environment. Meanwhile for this research dynamic analysis based on the system execution to find any malicious attack in code the best method to use. The important part it identifies vulnerabilities in runtime environment. Dynamic analysis can be conducted against any application which is match with our objective with any application for android based.

Therefore, dynamic analysis will be using in this project to analyze the android malware. It will focus on the behavior of attack, determine how and what it gets installed, how it run, what files have been added or modified, who they are communicate to, etc (Dennis Distler, 2007). So, the parameter such as system call will be investigate in this project.

2. Related Work

In this chapter, existence analysis that has been used to be implemented in this study will be discussed elaborately. There are many analyses that have been made throughout this past few years. Each of the analysis use different types of framework to have result.

A. *Android Application*

Android application can be available bt developer in their websites but to enhance the application user need to install from Google Play Store. Somehow some of the application in the Google Play Store required user to purchased or paid. From that reason make sure to find alternative way by download application from third party that possibility download together with Trojan.

B. *Trojan*

It is an evitable to have clear understanding of a Trojan in order to detect and response to it threats accurately and efficiently. In this section, the definition, comparison of Trojan virus with other malware and propagation are explained.

C. *Definition*

Most of malware detected in 2013 were Trojans. There were many well- constructed Trojans, which were difficult to analyze [2]. Trojan application downloads some HD wallpapers with user's permission but this permission may allow this app to access the ser's contacts or other personal information and it leaks user's confidential data to some other server from the device secretly [3]. This SMS Trojans exploit the premium services to incur financial loss to the victim. Malicious applications are the most common infection channel and are comparable to trojanized programs on desktop platforms. They provide high convenience for malware developers, as the Android Market and third-party app markets potentially gives access to all Android users. Malicious code can be packaged and redistributed with popular applications [4].

According to recent reports, nearly half of malicious files targeting Android devices are multifunctional Trojans that steal data from user's phones like contact names, email addresses, phone number and are capable of downloading additional modules from servers running by malicipus developers [5]. For the purpose of this research a Trojan is defined as a malicious program incidentally installed to smartphone by user which then communicate with the malicious program writers through command and control (C&C) server to receive attackers command. The command might be to leaks user confidential information, subscribing to premium number services and spread malware to other applications in android.

D. *Trojan Comparison with Others Malicious Software (Malware)*

Table 2: Comparison between Trojan with other malware

Type	Description
Virus	A program that has infected some executable software and, when run causes a virus to spread to other executable. A virus might corrupt or delete data on a computer, use email programs to infect other computers or even erase everything on a hard disk.
Trojan	Computer programs that appear to have a useful function, but that also have a hidden and potentially malicious function that evades security mechanisms by for example, masquerading as a useful program that a user would likely execute.
Spyware	Software that covertly gathers user information through an Internet connection without the user's knowledge for advertising purposes or to steal confidential information.
Ransomware	Malware that restricts access to the computer system that it infects and demands that a ransom be paid to the distributor of the ransomware in order for the restriction to be removed.
Botnet	A collection of compromised computers connected to the internet on which malware is running. Each compromised computer is called a bot. The human controlling a botnet is called a botmaster. Command and control servers are web servers that control the botnet under the direction of a botmaster.

E. *Android Malware Detection Techniques*

The main purpose of an android malware detection and response technique is to detect the presence of mobile malware in application which, if found could be cleaned, quarantined, blocked or deleted. Several approaches to android malware detection techniques have been attempted by [6], [3], [7], [8], [9], [10], [11], [12], [13], [14]. In those papers, some of the common techniques used for android malware detection can be categorized into static analysis and dynamic analysis.

F. *Static analysis*

In static analysis, an application is analyzed without executing it. Static analysis can directly be employed either on the source code of the application or the corresponding binary file and use reverse engineering techniques to extract certain features or methods might be invoked into from the source code. In android application, features and methods also can be analyzed from manifest file. Extracted features or methods not only can be used to detect malicious payload but also to profile and weigh malware threat [15]. A paper by [14] has listed features and methods that usually extracted from application source code which are; Requested Permission, imported Package, API Calls, Instructions or Operation Code (Opcode), Data Flow and Control Flow. Details of researchers that have used these features and methods will be discussed in next subchapter. Static analysis is simple and efficient in providing fast detection and classification for known mobile malware however it drawback is unable to detects unknown or mutated mobile malware because of obfuscation and encryption techniques employed by mobile malware writers. To overcome this limitation, dynamic analysis is used to detect mobile malware.

G. *Dynamic Analysis*

In contrast to static analysis, dynamic analysis does not inspect the source code but the application sample is analyzed while it is executed within controlled environment. In current studies, the behavior of the application can be monitored through Logged Behavior Sequence, System Calls and Dynamic Tainting, Data Flow and Control Flow [14]. By monitoring and logging every relevant operation of the execution to find any malicious attack in code the best method to use. The important part it identifies vulnerabilities in runtime environment. Dynamic analysis can be conducted against any application which is match with our objective with any application for android based.

H. *Related Studies of Android Malware Detection*

Table 3: Related Works of Malware Detection on Android

Related Work	**Type of Analysis**	**Key Features**	**Description**	TP Rate	FP Rate	Accuracy
[12]	Dynamic	System Calls Frequency	Proposes a new malware detection method based on system calls frequency. This paper determines 23 system calls list which can effectively characterize the characteristics of the application behaviour	0.98	0.024	98 %
[14]	Static & Dynamic	Malware Detection Through Dynamic & Dynamic Analysis	Analysed the Android malwares and their penetration techniques used for attacking the systems and antivirus programs that act against malwares to protect Android systems. We categorize many of the most recent antimalware techniques on the basis of their detection methods. We aim to provide an easy and concise view of the malware detection and protection mechanisms and deduce their benefits and limitations	Not stated	Not Stated	97%
[16]	Static	System Calls Pattern	Have extracted the system call trace of 345 malicious application from 10 Android malware families	Not Stated	Not Stated	Not Stated
[13]	Dynamic	Abnormal Behaviour	Android Malware	0.91	0.87	90 %
[11]	Dynamic	Amount of System Calls	Android Malware	Not Stated	Not Stated	Not Stated
Proposed model	Dynamic	Amount of System Calls	Generates Many False Positives	Not Stated	Not Stated	Not Stated

Published by FAZ Publishing
http://www.fazpublishing.com/acis

3. METHODOLOGY

This chapter explains on how this research is conducted. The research framework including explanation of research tools and environment, method used in data collection and analysis, overview of proposed model and research schedule including preliminary work that have been done. In understanding the research framework are illustrated in Figure 1. The next chapter will be focus on analysis, result, senility test and compare result.

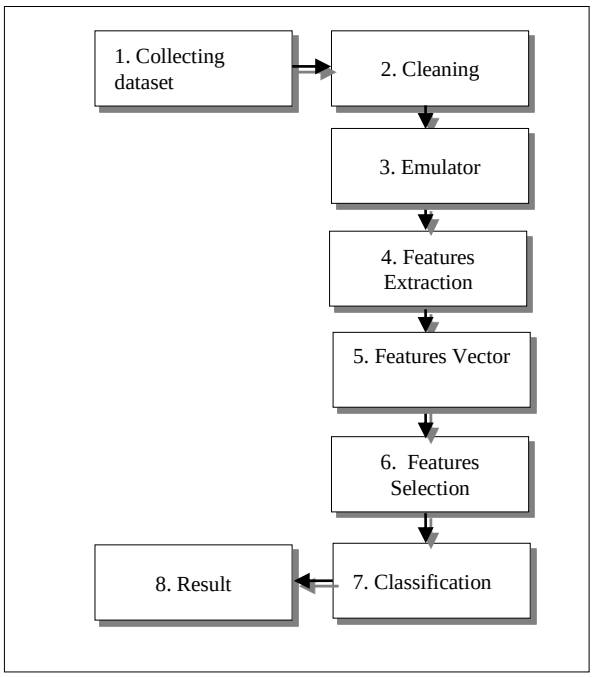

Figure 1: The research framework structure

A. Testing Lab Environment

The lab architecture in this research is illustrated in Figure 2. It is controlled lab environment with the software used for the testing is open source software which freely available on the Internet. No outgoing connection allows in this architecture so no harm of mobile malware threats will exposed to public. Tools and software installed used for this research is summarized in Table 4.

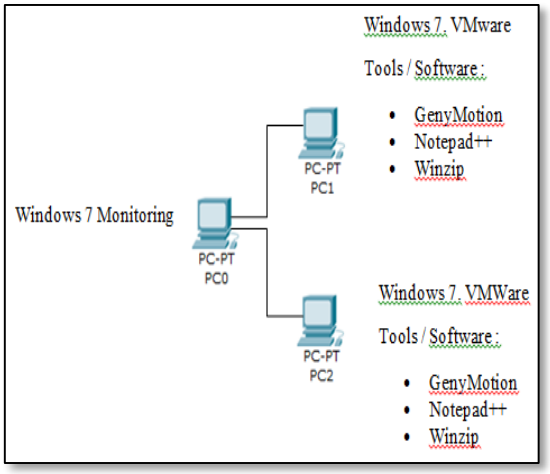

Figure 2: Lab architecture Model

Each tools and software that used for this research are in accordance with the research aim and objective. First virtual personal computer that complete with processor dual core 2.4 GHz, RAM 16 GB, operating system Windows 7 and with sentence software that importance. Software that required are Genymotion, Notepad++ and Winzip. Genymotion as a emulator for run the applications. The experiments employed the Waikato Environment for Knowledge Analysis (WEKA) machine learning tool due to its simplicity and user-friendly environment [17]. The Java Runtime Environment (JRE) version 1.7.0_21 operated the WEKA software.

Table 4: Software Installed in Testing lab Computers

Function	Software / Tools	Purpose of Action(s)
Virtual PC VMware	Work Station	To allow multiple operating systems to run on single computer To provide operating system for installation of tools or software for the mobile malware detection
Android Emulator	Genymotion	To provide smartphone emulator for installations of applications (benign and malicious applications) To crawl Google Play and download applications
Scan Tools	• Kaspersky Internet Security for Android • avast! Free Mobile Security • F-Secure Mobile Security • Andrubis (Online)	To scan and detect Android application and datasets used in this research
Unpack Tool	• Winzip	To unpack Android application package and Drebin dataset
Software for data testing and simulation	Java (WEKA)	To perform data mining analysis and testing

B. Data Collection Phase

Two datasets gave been chosen in this phase which are Android Trojan dataset and benign dataset.

1. *Trojan Dataset*

DREBIN dataset

A malware family is fundamentally a gathering of malware introducing a similar behaviour. The dataset contains 5,560 applications from 179 different malware families. The samples have been collected in the period of August 2010 to Octobeer 2012 and were made available to us by the MobileSandbox project [18].

2. *Benign Applications*

All the applications are selected at Google Play Store as benign applications. Benign something that is gentle, kind, mild or unharmful. The dataset that was used is in the form of .apk file downloaded from Google Play Store [19]. For the clean application samples, we choose 500 downloadable applications from the Google Play Store. Nevertheless Android applications in the Google Play Store may be infected with malware. In order to ensure that the clean applications chosen were really clean need to test. The applications tested by uploading each sample individually to VirusTotal (http://www.virustotal.com/) where five robust AV engines (Sophos, Symantec, F-Secure, TrendMicro and MacAfee) were used to verify cleanness.

C. Emulator

In this phase, the main task is to execute the APK files. In order to achieve this, have made use of the Android Emulator which is Genymotion. In this research execute each Android Trojan and benign application in separate emulator and record the system calls as soon as the application is installed in the emulator. This methodology records the frequency of all the successful system calls recorded. The log file contains percentage of the time utilized by the system call, seconds of the time for which the system call executed, count of the successful execution of the system call, number of errors in the interaction of system call and the name of the system call.

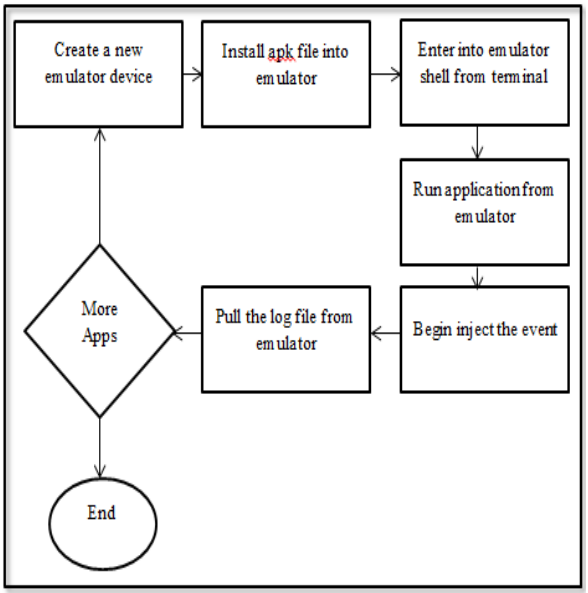

Figure 3: Flow chart of system call features extraction using emulator

D. Feature Extraction and Selection Phase

In this phase, dynamically examines the collected benign an malware .apk samples and extract the requisite features. Next, it will analyses the decompiled source code of an application, by identifying system calls by each of the application. For extracting the system call features of android application are used self-developed tool WEKA. System call analysis is an effective technique for detecting the Android Malware as in this technique; this do not require bytecode of the malicious application for doing the analysis [16]. Behaviour of both malicious and benign applications are profiled with a set of feature vectors [20]. In order to apply any Machine Learning technique, relevant features form every application have been extracted.

Android applications are delivered as .apk files. This is expected since permissions allow applications to perform actions that can potentially harm the user. Malicious applications also tend to request unusual permissions when compared to legitimate ones [21]. Every each sample, if requested permission matches with the Android permission, the is assigned as 1 to indicate it presence those sample while 0 indicate the absence of the permission. Feature selection are subdivide can be depends on top malware and information gain. Top malware is a selection when compared between the benign and Trojan dataset. In other cases for the information gain we need to complete the system call form the literature review and then compared. This research we choose to use information gain as the feature selection to complete it.

E. Feature Vector

The result from feature extraction phase is transform into vector in comma separated value (CSV) file for each 1000 Android applications. The vector starts either with value 0 or 1 which indicates the presences of android system calls and end with the class either Trojan or benign. The system calls is noted as 1 to indicate it presence in the sample while 0 indicates the absence of the system calls. Let R be a vector containing set of 61 system calls. For every application in the Android application dataset (Trojan and benign), Ri = {r1,r2,r3,….rj} and r is determined in formula below :

$$r_j = \begin{cases} 1 & \textit{If permission jth exist} \\ 0 & \textit{otherwise} \end{cases}$$

F. System Call Selection

Table 5: System Call Selection

No.	System Call	No.	System Call
1.	sendto	32.	socket
2.	recvfrom	33.	uname
3.	ioctl	34.	bind
4.	clock_gettime	35.	connect
5.	epoll_ctl	36.	select
6.	getpid	37.	rt_sigreturn
7.	munmap	38.	dup
8.	getuid32	39.	fcntl64
9.	write	40.	unlink
10.	read	41.	_llseek
11.	futex	42.	mkdir
12.	lseek	43.	rename
13.	gettid	44.	restart_syscall
14.	open	45.	fork
15.	mmap2	46.	umask
16.	sigprocmask	47.	sched_yield
17.	gettimeofday	48.	fchown32
18.	clone	49.	nanosleep
19.	mprotect	50.	getdents64
20.	writev	51.	poll
21.	stat64	52.	getsockname

22.	madvise	53.	getsockopt
23.	fstat64	54.	setsockopt
24.	brk	55.	flock
25.	pread64	56.	fchmod
26.	fdatasync	57.	wait4
27.	fsync	58.	inotify_add_watch
28.	pwrite64	59.	statfs64
29.	chmod	60.	epoll_wait
30.	access	61.	close
31.	lstat64		

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of different classification algorithms, the following metrics are used, True Positive Rate (TPR), False Positive Rate (FPR) and Precision. Performance of patterns set are compared in terms of True Positive Rate (TPR), False Positive Rate (FPR) and accuracy. These metrics are calculated using the confusion matrix as shown in Table 4.2 generated from the four measures :

$$\text{True positive rate} = \frac{TP}{TP + FN}$$

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Confusion Matrix : Results of an experiment can be represented in the form of table known as confusion matrix.

Table 6 : Confusion Matrix

Actual Class	Detected as Trojan	Detected as Benign
Trojan	True Positive (TP)	False Negative (FN)
Benign	False Positive (FP)	True Negative (TN)

Where, True Positive (TP) is the number of Android Trojan apps classified as Trojan. True Negative (TN) is the number of benign apps classified as benign. False Positives (FP) is the number of benign.Table 6 shows the comparison between the three classifications algorithm.

Table 7: Comparison between three classifications algorithms with 61 top system call

Algorithm	TP Rate	FP Rate	Accuracy
.Random Forest	0.812	0.188	81.2
J48	0.741	0.259	74.1
NaiveBayes	0.648	0.352	64.8

It is clear from Table 6 that Random Forest algorithm achieved the best performance in terms of all used metrics. In order to highlight the significance of this research result, acomparison is made with previous similar research. Table 7 show how the results in this paper measures against the best results of Chen Da (2017), Sato (2013) and Wu Dong- Jie Mao (2012) respectively.

Table 8 : The performance of our research with others

Related Works	C. Da (2017)	Sato (2013)	Wu, Dong-Jie Mao (2012)	ATD-DA
System Call Used	350	100	Not Stated	61
Sample	51	365	1500	1000
TP Rate	0.95	0.91	Not Stated	0.812
FP Rate	0.024	0.87	Not Stated	0.188
Accuracy	98 %	90 %	97 %	81.2 %

```
Correctly Classified Instances         812               81.2    %
Incorrectly Classified Instances       188               18.8    %
Kappa statistic                          0.624
Mean absolute error                      0.2866
Root mean squared error                  0.375
Relative absolute error                 57.3277 %
Root relative squared error             75.0035 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure
                 0.848    0.224      0.791     0.848      0.819
                 0.776    0.152      0.836     0.776      0.805
Weighted Avg.    0.812    0.188      0.814     0.812      0.812

=== Confusion Matrix ===

   a   b   <-- classified as
 424  76 |   a = trojan
 112 388 |   b = benign
```

Figure 4 : Test results under WEKA Experimental

Experimental results show that this method can distinguish benign and malicious application. Although other researcher have higher TP Rate but they use small sample with only 51 sample. Which this research use large of sample with 1000 sample. The other researcher use 1500 sample with higher TP rate but not mention how many system call that has been used.

V. Conclusion & Future Work

Android Trojan apps are real threats for smartphones as they effect more serious threats compared to other Android malwares. The research proposed approach uses the system call used in the android apps as features to differentiate between the Trojan Apps and benign apps. The Information Gain algorithm is used to select the most significant system

calls then the classification algorithms, Random Forest, Tress.J48, Rules. Part and Naïve Bayes used to classify the android apps as Trojan or benign apps. The experimental results show that Random Forest algorithm achieved the highest detection accuracy of 0.812 or 81.2 % with lowest false positive rate 0.188. Although the accuracy is low but this research manages to reduce the use of the system call. For future research, more significant features of system call will be used to achieve better detection accuracy with this help of this system calls order list.

References

[1] Z. Abdullah, M. M. Saudi, and N. B. Anuar, "Mobile botnet detection: Proof of concept," *Proc. - 2014 5th IEEE Control Syst. Grad. Res. Colloquium, ICSGRC 2014*, pp. 257–262, 2014.

[2] Z. Liu, Y. Li, H. Yang, and J. I. E. Qiu, "A CASE STUDY ON KEY TECHNOLOGIES OF ANDROID TROJANS," pp. 321–324.

[3] S. Arshad, M. Ali, A. Khan, and M. Ahmed, "Android Malware Detection & Protection: A Survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 463–475, 2016.

[4] A. P. Evaluation, "Android OS Security : Risks and Limitations," 2012.

[5] N. Peiravian, "Data Mining Heuristic--Based Malware Detection for Android Applications," no. December 2013, 2013.

[6] B. Torregrosa, "A framework for detection of malicious software in Android handheld systems using Machine Learning techniques .," 2015.

[7] A. Schmidt *et al.*, "Static analysis of executables for collaborative malware detection on android," *IEEE Int. Conf. Commun. 2009*, pp. 0–4, 2009.

[8] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, 2009, p. 235.

[9] T. Taylor *et al.*, "Using hardware features for increased debugging transparency," *Proc. - IEEE Symp. Secur. Priv.*, vol. 6829 LNCS, no. February, p. 83, 2014.

[10] T. Bläsing, L. Batyuk, A. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox system for suspicious software detection," in *2010 5th International Conference on Malicious and Unwanted Software*, 2010, pp. 55–62.

[11] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android," *Proc. 1st ACM Work. Secur. Priv. smartphones Mob. devices - SPSM '11*, p. 15, 2011.

[12] C. Da, H. Zhang, and X. Zhang, "Detection of Android malware security on system calls," *Proc. 2016 IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf. IMCEC 2016*, pp. 974–978, 2017.

[13] R. Sato, D. Chiba, and S. Goto, "Detecting Android Malware by Analyzing Manifest Files," *Proc. Asia-Pacific Adv. Netw.*, vol. 36, no. 0, p. 23, 2013.

[14] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," *2012 Seventh Asia Jt. Conf. Inf. Secur.*, pp. 62–69, Aug. 2012.

[15] S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," *2013 IEEE 27th Int. Conf. Adv. Inf. Netw. Appl.*, pp. 121–128, Mar. 2013.

[16] S. Malik and K. Khatter, "System call analysis of Android Malware families," *Indian J. Sci. Technol.*, vol. 9, no. 21, 2016.

[17] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Comput.*, vol. 20, no. 1, pp. 343–357, 2016.

[18] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," *Proc. 2014 Netw. Distrib. Syst. Secur. Symp.*, no. February, 2014.

[19] V. Wahanggara and Y. Prayudi, "Malware Detection through Call System on Android Smartphone Using Vector Machine Method," *Proc. - 4th Int. Conf. Cyber Secur. Cyber Warf. Digit. Forensics, CyberSec 2015*, pp. 62–67, 2016.

[20] B. Amos, H. A. Turner, B. Amos, H. Turner, and J. White, "Applying machine learning classifiers to dynamic Android malware detection at scale Applying machine learning classifiers to dynamic Android malware detection at scale," no. November, 2014.

[21] D. Barrera, H. G. üne ş Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," *Proc. 17th ACM Conf. Comput. Commun. Secur. - CCS '10*, no. 1, p. 73, 2010.