International Journal of Engineering & Technology, 7 (4.15) (2018) 103-108

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Mobile Botnet Classification by using Hybrid Analysis

Muhammad Yusof[1], Madihah Mohd Saudi[1,2]*, Farida Ridzuan[1,2]

[1]Faculty of Science and Technology (FST),
[2]CyberSecurity and Systems Research Unit, Islamic Science Institute (ISI)
Universiti Sains Islam Malaysia (USIM), Negeri Sembilan, Malaysia
**Corresponding author E-mail: madihah@usim.edu.my*

Abstract

The popularity and adoption of Android smartphones has attracted malware authors to spread the malware to smartphone users. The malware on smartphone comes in various forms such as Trojans, viruses, worms and mobile botnet. However, mobile botnet or Android botnet are more dangerous since they pose serious threats by stealing user credential information, distributing spam and sending distributed denial of service (DDoS) attacks. Mobile botnet is defined as a collection of compromised mobile smartphones and controlled by a botmaster through a command and control (C&C) channel to serve a malicious purpose. Current research is still lacking in terms of their low detection rate due to their selected features. It is expected that a hybrid analysis could improve the detection rate. Therefore, machine learning methods and hybrid analysis which combines static and dynamic analyses were used to analyse and classify system calls, permission and API calls. The objective of this paper is to leverage machine learning techniques to classify the Android applications (apps) as botnet or benign. The experiment used malware dataset from the Drebin for the training and mobile applications from Google Play Store for testing. The results showed that Random Forest Algorithm achieved the highest accuracy rate of 97.9%. In future, more significant approach by using different feature selection such as intent, string and system component will be further explored for a better detection and accuracy rate.

Keywords: *Android; Classification Algorithm; Hybrid Analysis; Mobile Botnet.*

1. Introduction

Nowadays, smartphones are playing an important role in this modern life due to the capabilities offered by smartphones and rapid development of their computing power in the recent years. People prefer to use smartphones or mobile devices for financial activities and store sensitive data on their mobile devices rather than in computers [1]. According to Gartner Newsroom 2017, 380 million smartphones were sold worldwide for the first quarter of 2017, which is a 9.1% increase as compared to the same quarter last year. Gartner also released an operating system (OS) market shares report which showed that Android has led the market with 86.1%, an increase from 84.1% in the first quarter of 2017 [2].

Android has become the most targeted due to the nature of its operating system ecosystem [3]. Android users can download an apps not only from the official Google Play Store, but they can also choose any third-party apps markets, torrents or direct downloads from the Internet. Many manufacturing companies, such as Amazon, Samsung, LG, Huawei, Lenovo, Dell, Cisco and Docomo, own specific Android apps market to cater for their hardware. Study from [4] showed that 5-8% malicious apps are available in the third-party market. In the third-party marketplace, apps developer can openly publish any malicious or benign applications in the market. Android had introduced Bouncer in 2012 as the first security line for every apps that will be uploaded to Google Play Store [5]. Unfortunately, some security researchers can still manipulate this security process [6-7]. One of the attack methods is through malicious apps installation. Android users are supposed to download and install apps from the official Google Play Store. However, user can also download apps from the third-party market. Many malware authors submit their malicious apps to this third-party or alternative market to attack users. Normally, the unofficial markets provide free, non-paying apps or cheaper apps as compared to Google Play Store. This attracts more users to use the unofficial market and thus can possibly expose the smartphone user to download and install malicious applications from this market.

Mobile malware is a malicious software that was targeted at mobile phones instead of the traditional computer system. This malware is designed to exploit a mobile device without the owner's consent. Meanwhile, mobile botnet is defined as a network consisting of a collection of affected mobile devices, controlled by a botmaster through a command and control (C&C) network [8]. The C&C network is the core of any botnet and botmaster that use the C&C network to issue commands and control the whole botnet [9]. In the early days, mobile botnet was not the choice for the attackers because of the limitation in mobile network communication and mobile device performance. Nowadays, with the major deployment of 4G technologies, such as LTE and WiMAX, the mobile network communication has become more stable for mobile botnet to launch the service to manipulate mobile users. The improvement in smartphone capabilities with full-featured operating systems (OS) that is incorporated with powerful hardware and long-lasting battery, together with the popularity associated with mobile devices have caused mobile botnet developers to shift their attack towards mobile devices.

This paper aims to produce a new mobile botnet classification based on permissions, API calls and system calls features. According to [10], permissions, API calls and system calls are the most popular and comprehensive feature selections that were used by previous researchers for mobile malware detection. 5,560 samples

from Drebin dataset and 800 samples from Google Play Store were extracted for training and testing by using both static and dynamic analyses. As a result, 20 permissions, 38 API calls and 44 system calls that are most related to mobile botnet were extracted by using feature selection and later classified and tested by using Naïve Bayes, K-Nearest Neighbors, Random Forest and Support Vector Machine algorithms.

The rest of this paper is organised as follows: Section 2 presents the related work. Section 3 discusses the methodology used, Section 4 discusses the experimental results and Section 5 presents conclusion and future research.

2. Related Work

Existing mobile malware detection research is performed based on static, dynamic and hybrid analysis. In static analysis, the features are extracted from the application (app) file without executing the application. This method is resource and time efficient as the application is not executed [11]. Meanwhile, dynamic analysis does not examine the source code but its execution is carried out within a controlled laboratory environment, often called sandbox. In [12] provided a complete overview of automated dynamic malware analysis techniques. The third type of analysis is called hybrid analysis, which combines static and dynamic analyses. With the combination of static and dynamic analyses, more features can be extracted from the analysis. Using relevant features for the analysis will produce a much better result. Hybrid analysis can be used to overcome the drawback from static and dynamic analysis weaknesses such as obfuscation techniques and sandboxed environment. Although all methods are usually used in malware analysis and detection, these methods are also applicable as a basis for mobile botnet analysis and detection. Hence, hybrid analysis is applied in this research.

Many tools were developed for extracting various dynamic and static features of android application package (APK) files. For static analysis, in [13] were among the first researchers who investigated the mobile malware detection. They introduced Kirin, a security service for Android, which provides security rules for the apps during the installation. In [14] proposed machine learning with Bayesian Classification models for mobile malware detection along with static code analysis by using feature selection, such as permission, API calls and Linux system commands.

Meanwhile, dynamic analysis does not examine the source code, but rather executes the apps and monitors and logs every relevant execution operation. In [15] proposed Andromaly, a host-based Android mobile malware detection which has deployed machine learning on it. It was tested by using 88 features to describe behaviours, such as CPU and memory usage, power consumption, network utilisation and number of running processes. Crowdroid is a behaviour-based malware analysis and detection approach [16]. This framework accumulates the system calls by using the Strace tool and detects the presence of malware. Multi-level Anomaly Detector for Android Malware (MADAM) uses a machine learning K-Nearest Neighbors (KNN) classifier for mobile malware detection [17]. It uses 13 features to detect malware at both kernel and user level. MADAM has successfully achieved 93% of the accuracy rate. Although this approach produces a good result, it is incapable of detecting malware that avoids the system calls with root permission.

Feature selection is seen as a promising way to extract mobile botnet features as it has considerable effects on experimental results. The most comprehensive study on feature selection for mobile malware is done by [10]. They highlighted all the feature selection methods for mobile malware detection and categorised all available features into four groups, which are static features, dynamic features, hybrid features and application metadata.

In [18] proposed Droid-Sec, a combination of static and dynamic analyses which extracted more than 200 features by using deep learning classifier to detect android malware. This study extracted the features based on permission (120), sensitive API (64) and dynamic behaviour (18). Although the result was promising with a high accuracy rate of 96.5%, the data set was considered too small with 300 samples for malicious apps and 200 samples for benign apps. Mobile-Sandbox is an automatically Android apps analyser which uses the combination of static and dynamic analysis [19]. The results from a static analysis are then used by the dynamic analysis to execute the apps. This automatic web-based analyser is combined with machine learning techniques to produce better detection results.

Many researchers have studied on specific trends, characteristics, architectures, impact, type of attacks, detection approach, code, structure and behaviour relating to mobile botnet behaviour such as [20-21]. In [22] analysed on mobile botnets C&C and the URLs and produced a new mobile botnet dataset based on samples of 14 botnet families. In [23] presented detail information on mobile botnet characteristics and behaviour based on 20 mobile botnet families. Meanwhile, in [24] were among the early researchers who focused on mobile botnet detection. They proposed a mobile botnet detection by inspecting abnormal network flow through a virtual private network (VPN) with a 94.6% detection rate. In [25] developed Android Botnet Classification (ABC) classification on Android mobile botnet based only on permission with 94.6% detection rate. A Prototype of Android Botnet Identification System (ABIS) extracts feature set permissions and API calls from Android apps to identify the botnet and its family [26]. They were implemented machine learning algorithm to classify the Android botnet family with a high detection rate. DeDroid used static analysis based on permissions and API calls feature to examine botnet-specific properties that can be used to detect mobile botnets [27]. They found that 35% of malware on Drebin dataset was considered as botnets [28]. Hybrid analysis will solve the problem for malware packing and obfuscation techniques but there is a lack of existing research that were focused on hybrid analysis. Therefore, this research intends to develop a new classification by using hybrid analysis based on feature set permissions, API calls and system calls for mobile botnet classification. To the best of knowledge, there is no research on mobile botnet classification and detection based on hybrid analysis.

3. Methodology

This research proposed a hybrid analysis for mobile botnet classification. The paper is an extension of the research that used feature set permissions and API calls [29]. 1,527 botnets samples from 14 botnets families from Drebin [28] dataset were used to collect all information for botnets features. Mobile botnets samples were studied and analysed to select the most related mobile botnet feature set of permission, API calls and system calls. All features from 1,527 botnets samples were extracted by using reverse engineering for static analysis and were executed in a sandbox environment for dynamics analysis. The list of Android mobile botnet family and the number of samples are listed in Table 1. The overview of process analysis is shown in Figure 1.

Table 1: Android Mobile Botnet Family

Botnet Family	Year	Number of Samples	Motivation
Geinimi	2010	218	Data stealing
DroidDream	2011	297	Data stealing
AnserverBot	2011	203	Propagation of possible malware
PjApps	2011	194	Financial and data stealing
NickySpy	2011	128	Spying/data stealing
TigerBot	2012	68	Financial, spying/data stealing
RootSmart	2012	27	SMS/mobile Transaction Authentication Number (mTAN) stealing
Zitmo	2012	65	Financial

Bmaster	2012	4	Financial, SMS stealing
MisoSMS	2013	72	Proxy
NotCompatible	2014	59	Data stealing
Sandroid	2014	36	Financial, mobile banking attack
Pletor	2014	64	Ransomware
Wroba	2014	82	Financial, mobile banking attack

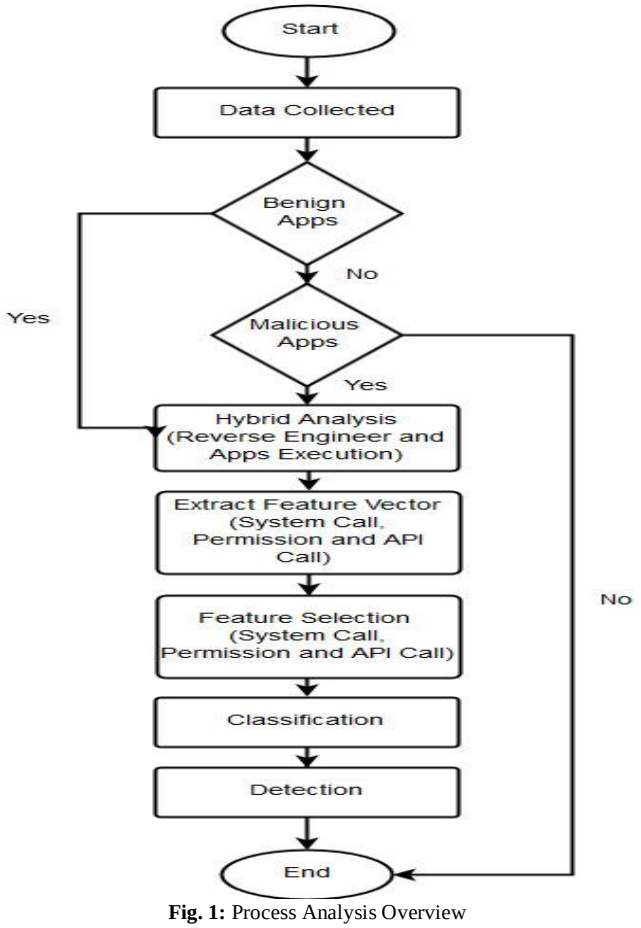

Fig. 1: Process Analysis Overview

3.1. Dataset

Drebin dataset [28] which consists of 5,560 malware from 179 different families, were used for training. Drebin dataset is currently one of the largest freely available dataset on the Internet which are used in mobile malware research [29-32]. Meanwhile, for testing, 800 apps datasets of random categories were downloaded from Google Play Store and were tested with Virustotal [33] to ensure the apps were free from malicious.

3.2. Feature Selection

The feature selection plays an important part for mobile malware and botnets detection. According to [10], feature selection can reduce the noise and irrelevant data from datasets to produce higher accuracy results of machine learning algorithms. It can also reduce the runtime of the machine learning algorithms during the training phase. In this research, the feature selection was classified from the permission, API calls and system calls. Both static and dynamic analyses techniques were used to select the most related features from those three features selection. Both analyses have their advantages and disadvantages. For example, static analysis can run faster but time consuming, especially for the large dataset. Meanwhile dynamic analysis can investigate the obfuscation code, but static analysis cannot.

3.3. Feature Extraction

For static analysis, the features of permissions and API calls were extracted by using a reverse engineering tool like Apktool [34] and dex2jar [35]. Permissions features were extracted from Manifest.xml file and API calls were extracted from .dex class file. Meanwhile the features of system calls were extracted from strace command with a sandbox environment by using dynamic analysis. Then, this experiment applied a macro script with the string similarity method to extract permissions, API calls and system calls. For each sample, if the requested permissions, API calls or system calls match the Android permissions, API calls and system calls, it was represented as 1 to indicate its presence in the sample, while 0 indicated the absence. Let R be a vector containing a set of 147 Android permissions [36]. For every ith application in the Android application dataset (botnet and benign), $R_i = \{r_1, r_2, r_3, \ldots r_j\}$ and

$$r_j = \begin{cases} 1, & \text{if } j_{th} \text{ permission exist} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

As a result, this research summarised the top 20% features used in 1,527 samples from 14 botnet families, as shown in Table 2.
After feature set was selected and summarised, it was applied to the Drebin dataset (malicious) for training purpose and was evaluated with benign apps from Google Play Store. Only 5,482 apps were selected from Drebin dataset out of 5,560 apps due to the constraint to execute and reverse engineer in both static and dynamic analyses. Static and dynamics analyses were carried out and then macro script was applied to the string similarity method to compare 6,282 samples (malicious and benign) with 20 permissions, 38 API calls and 44 system calls feature set. The result of this extraction process was transformed into vector in comma separated value (CSV) format file. The value of vector started either with the value of 1 or 0, which indicated whether the samples were present or absent for the feature set. Figure 2 shows an example of permission vectors. The file starts with the hash function.

005b46f64c266bc14ced91703b28c2411f2f4e4e19cae9f3883
d385cb27d7642,1,0,1,0,0,0,1,1,0,1,0,……,1

Fig. 2: Example of Permission Vector

3.4. Classification

$$True\ positive\ rate, TPR = \frac{TP}{TP + FN} \quad (2)$$

$$False\ positive\ rate, FPR = \frac{FP}{TN + FP} \quad (3)$$

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$Accurancy, ACC = \frac{TP + TN}{TP + FP + TN + FN} \quad (6)$$

Five machine learning classifiers were used in the dataset, which are Naïve Bayes (NB), K-Nearest Neighbors (KNN) Decision Tree (J48), Random Forest (RF) and Support Vector Machine (SVM) with SMO. These five classifiers were chosen due to their common use in other similar works [26, 37-39]. The experiment was conducted by using Waikato Environment for Knowledge Analysis (WEKA), which is a software tool that was widely employed for implementing the feature selection method and the classification algorithm [36]. All the classifiers were run based on default setting. The experiment also uses a standard 10-fold cross validation technique to determine classification accuracies of all algorithms.

Table 2: Top 20% Features of Permission, API Calls and System Calls

Permission	Occurrence (%)	API Calls	Occurrence (%)	System Calls	Occurrence (%)
INTERNET	98.10	getSystemService()	97.90	Read	99.61
READ_PHONE_STATE	97.38	startService()	96.53	Write	99.35
ACCESS_NETWORK_STATE	88.79	openConnection()	95.54	clock_gettime	99.35
ACCESS_WIFI_STATE	82.24	getDeviceId()	95.09	mprotect	99.28
WRITE_EXTERNAL_STORAGE	70.05	getActiveNetworkInfo()	87.22	writev	99.21
RECEIVE_BOOT_COMPLETED	65.01	android/content/Context;->startActivity	85.32	getpid	99.08
ACCESS_COARSE_LOCATION	64.48	java/net/HttpURLConnection;->connect	78.05	Ioctl	99.02
ACCESS_FINE_LOCATION	57.54	notify()	76.15	Close	98.69
VIBRATE	55.37	getPackageInfo	70.90	Stat	95.48
READ_SMS	46.40	getNetworkInfo()	70.25	Poll	91.94
WAKE_LOCK	45.28	HttpPost	68.74	epoll_wait	91.94
INSTALL_SHORTCUT	36.04	query()	68.61	gettid	91.88
READ_CONTACTS	30.21	execute()	63.96	getuid	91.81
CHANGE_WIFI_STATE	29.29	getInputStream()	63.04	Open	91.68
SEND_SMS	28.31	getSubscriberId()	61.99	mmap	91.42
UNINSTALL_SHORTCUT	28.31	getLastKnownLocation()	61.93	munmap	91.36
CALL_PHONE	27.33	getLine1Number()	61.07	recvfrom	90.83
WRITE_CONTACTS	25.95	android/app/Activity;->startActivity	60.94	Brk	89.91
ACCESS_LOCATION_EXTRA_COMMANDS	25.03	java/net/URLConnection;->connect	60.88	sendto	89.46
RECEIVE_SMS	21.49	CryptoCipher	56.09	madvise	89.19
		exec()	51.51	fstat	89.06
		sendBroadcast()	51.11	futex	89.06
		requestLocationUpdates()	50.92	dup	88.74
		getConnectionInfo()	49.67	epoll_ctl	88.02
		Cipher(AES)	48.95	fcntl	87.49
		getSimSerialNumber()	44.36	pread64	87.03
		getBestProvider()	34.73	gettimeofday	87.03
		openStream()	32.83	clone	82.58
		start()	30.60	access	63.52
		system/bin/su	30.14	sync	49.84
		acquire()	29.29	chmod	48.53
		isProviderEnabled()	29.23	lstat	43.81
		release()	29.16	fsync	43.75
		setWifiEnabled()	29.10	lseek	42.24
		getWifiState()	28.90	link	39.69
		getCellLocation()	24.18	unlink	39.69
		stop()	23.13	nanosleep	39.16
		getContent()	20.90	rename	38.44
				sched_yield	36.48
				chown	31.63
				fchown	31.63
				umask	31.63
				pwrite64	26.85
				fdatasync	26.00

4. Results and Discussion

This section presents the experimental results and the performance of the proposed feature set. The results were compared with single and other combination feature sets containing the permissions, API calls and system calls. The performance was evaluated by using True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall and Accuracy (ACC) which are defined as the following:

where True Positive (TP) is the number of malicious applications classified correctly as a botnet. False Positive (FP) is the number of benign applications that were classified as malicious. True Negative (TN) is the number of benign applications that were classified as benign; and False Negative (FN) is the number of malicious applications that were classified as benign. Precision refers to the probability that an application is classified as a botnet application correctly. Recall or detection rate is defined as the portion of the total malicious applications that were classified as botnet. Accuracy (ACC) is defined as correctly classified samples in the category divided by the number of all samples from malicious and benign.

The single feature from system calls, permissions and API calls were classified and the results are shown in Table 3. The accuracy rate of most features ranges from 87% to 93%. Permission and API calls were the only feature that achieved the best accuracy rate between 93% and 93.9%. The best accuracy rate was achieved by the Random Forest classifier for only permission feature with 93.9%. Meanwhile, the combination two only feature and the classifier results are displayed in Table 4. Feature set permissions and API calls achieved the best accuracy rate from about 96% to 97%. The best performance, 97.3% was achieved by feature set permission and API calls by using Random Forest classifier. The result showed that the combination of feature sets, such as permissions and API calls produced the best result as compared to other feature set combination. The combination feature sets system calls with permissions and system calls with API calls just produced the accuracy rate of 93.5% and 93.8%, respectively. The results of TPR and FPR were slightly similar to previous study [29].

Table 3: Results With Only One Feature

Classifiers	System Calls					Permission					API Calls				
	Measure Metrics (%)					Measure Metrics (%)					Measure Metrics (%)				
	TPR	FPR	PRE	REC	ACC	TPR	FPR	PRE	REC	ACC	TPR	FPR	PRE	REC	ACC
NB	42.3	25.2	92	42.3	46.4	88	26.5	95.8	88	86.2	76.4	21.1	96.1	76.4	76.7
KNN	98.7	90.5	88.2	98.7	87.4	97.7	32.6	95.4	97.7	93.8	99.5	45.2	93.8	99.5	93.8
J48	99.4	92.1	88.1	99.4	87.8	97.6	36.2	94.9	97.6	93.3	99.2	49.1	93.3	99.2	93.0
RF	98.5	89	88.4	98.5	87.4	97.4	30.2	95.7	97.4	93.9	99.6	45.9	93.7	99.6	93.8
SVM	99.8	98.4	87.5	99.8	87.3	97.6	54.3	92.5	97.6	91.0	98.6	50.8	93	98.6	92.4

Table 4: Results With Combination Only Two Features

Classifiers	System Calls + Permission					System Calls + API Calls					Permission + API Calls				
	Measure Metrics (%)					Measure Metrics (%)					Measure Metrics (%)				
	TPR	FPR	PRE	REC	ACC	TPR	FPR	PRE	REC	ACC	TPR	FPR	PRE	REC	ACC
NB	74.9	20.4	96.2	74.9	75.5	71.2	19.8	96.1	71.2	72.3	83.9	19.4	96.7	83.9	83.5
KNN	97.4	45.9	93.6	97.4	91.9	97.5	39.4	94.4	97.5	92.8	98.7	19.7	97.2	98.7	96.4
J48	97.7	35.1	95	97.7	93.5	98.1	35.6	95	98.1	93.8	98.4	14.7	97.9	98.4	96.8
RF	98.2	41	94.3	98.2	93.2	98.3	36.8	94.8	98.3	93.8	99.4	16.7	97.6	99.4	97.3
SVM	98.4	49.3	93.2	98.4	92.3	98.5	46.6	93.6	98.5	92.7	98.6	16.6	97.6	98.6	96.7

However, the results of the proposed feature set, which included the feature set from system calls, permissions and API calls are shown in Table 5. The best accuracy rate is 97.9%, which is achieved by the Random Forest algorithm. As mentioned earlier in Section 3, the objective of this paper is to determine whether the combination of features set, which consist of system calls, permissions and API calls can provide additional knowledge in characterising the characteristics and behaviours of botnet applications. This paper has proven that the combination of feature sets of system calls, permissions and API calls with the selection of relevant and suitable features could increase the mobile botnet detection accuracy rate.

Table 5: Results With Proposed Features

Classifiers	System Calls + Permission + API Calls				
	Measure Metrics (%)				
	TPR	FPR	PRE	REC	ACC
Naïve Bayes	89.1	19.4	96.9	89.1	88.1
K-Nearest Neighbors	99.1	10.9	98.4	99.1	97.8
Decision Tree	98.5	13.2	98.1	98.5	97.0
Random Forest	99.4	12.4	98.2	99.4	**97.9**
SVM with SMO	98.3	15.6	97.7	98.3	96.5

As mentioned earlier, there was a lack of research carried out on mobile botnet detection by using hybrid analysis and applied machine learning classifiers. In order to highlight the significance of this research result, a comparison is made with previous similar research. Table 6 shows the comparison between this research and Droid-Sec [18]. Although Droid-Sec investigated more into mobile malware with different dataset, the characteristics and behaviours of mobile malware can be applied to the mobile botnet too. Both approaches had used hybrid analysis with the combination of three different features. Both approaches also used five types of machine learning classifier although only two were of the same classifier. More classifiers expedite a more comprehensive analysis.

Based on Table 6, it is shown that the results of this study are superior to Droid-Sec. A 97.9% accuracy rate was achieved in this study by using the Random Forest classifier as compared to 96.5% in the other work by using the Deep Learning classifier.

Table 6: Comparison of Proposed Work With Other Approach

Related Works	Droid-Sec[18]	Proposed Work
Numbers of Features	202	102
Number of Samples (Malicious + Benign)	500	6282
Naïve Bayes (%)	79	88.1
K-Nearest Neighbors (%)	-	97.8
Decision Tree (%)	-	97.0
Random Forest (%)	-	97.9
SVM with SMO (%)	80	96.5
Deep Learning (%)	96.5	-

5. Conclusion

In this study, hybrid analysis and applied machine learning classifiers were used to classify Android mobile botnet. The combination of feature set system calls, permissions and API calls by selecting the most related and suitable features will increase the accuracy rate. Five algorithms, which are Naïve Bayes, K-Nearest Neighbors, Decision Tree, Random Forest and SVM with SMO were used to classify an app as a mobile botnet or benign. The experimental results have shown that the combination feature set of system calls, permissions and API calls with applied Random Forest classifier has achieved the highest accuracy rate of 97.9%. The experimental result indicates that the proposed feature set system calls, permissions and API calls provided more useful features than the other one or two combination features. This paper has achieved its objective to produce the best mobile botnet classification and an accuracy rate with the combination of feature set system calls, permissions and API calls. For future work, more significant approach, such as by using different feature selection like intent, string and system component as well as to combine static and dynamics will be further explored for a better accuracy rate.

Acknowledgement

The authors would like to express their gratitude to the Ministry of Higher Education (MOHE), Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This work was supported by grant: [USIM/FRGS/FST/32/50114].

References

[1] Poonguzhali, P., Dhanokar, P., Chaithanya, M. K., & Patil, M. U. (2016). Secure storage of data on android based devices. Int. Journal Eng. Technology, 8(3), 177–182.

[2] Gartner Newsroom. (2017). Gartner says worldwide sales of smartphones grew 9 percent in first quarter of 2017, http://www.gartner.com/newsroom/id/3725117.

[3] Alcatel-Lucent. (2015). Mobile malware: A network view. https://www.blackhat.com/docs/ldn-15/materials/london-15-McNamee-Mobile-Malware-A-Network-View-wp.pdf.

[4] Lindorfer, M., Volanis, S., Sisto, A., Neugschwandtner, M., Athanasopoulos, E., Maggi, F., Platzer, C., Zanero, S., & Ioannidis, S. (2014). AndRadar: Fast discovery of android applications in alternative markets. Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 51-71.
[5] Lockheimer, H. (2012). Android and security, http://googlemobile.blogspot.my/2012/02/android-and-security.html.
[6] Oberheide, J. (2012). Dissecting the android bouncer, https://jon.oberheide.org/blog/2012/06/21/dissecting-the-android-bouncer/.
[7] Percoco, J. & Schulte, S. Adventures in BouncerLand,://media.blackhat.com/bh-us.
[8] Pieterse, H., & Olivier, M. (2013). Design of a hybrid command and control mobile botnet. Proceedings of the 8th International Conference on Information Warfare and Security, pp. 1-13.
[9] Geng, G., Xu, G., Zhang, M., Guo, Y., Yang, G., & Cui, W. (2012). The design of SMS based heterogeneous mobile botnet. J. Comput., 7(1), 235–243.
[10] Feizollah, A., Anuar, N. B., Salleh, R., & Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. Digit. Investig., 13. 22–37.
[11] Baskaran, B., & Ralescu, A. (2016). A study of android malware detection techniques and machine learning. Proceedings of the Modern Artificial Intelligence and Cognitive Science Conference, pp. 15–23.
[12] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv., 44(2), 1–42.
[13] Enck, W., Ongtang, M., & McDaniel, P. (2009). On lightweight mobile phone application certification. Proceedings of the 16th ACM Conf. Comput. Commun. Secur., pp. 235–245.
[14] Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik I. (2013). A new android malware detection approach using Bayesian classification. Proceedings of the IEEE 27th Int. Conf. Adv. Inf. Netw. Appl., pp. 121–128.
[15] Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). Andromaly': A behavioral malware detection framework for android devices. J. Intell. Inf. Syst., 38(1), 161–190.
[16] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-based malware detection system for android. Proceedings of the 1st ACM Work. Secur. Priv. smartphones Mob. Devices, pp. 1-11.
[17] Dini, G., Martinelli, F., Saracino, A., & Sgandurra, D. (2012). MADAM: A multi-level anomaly detector for android malware. Lect. Notes Comp. Sc. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 7531, 240–253.
[18] Yuan, Z., Lu, Y., Wang, Z., & Xue, Y. (2014). Droid-Sec: Deep learning in android malware detection. Proceedings of the ACM conference on SIGCOMM, pp. 371–372.
[19] Spreitzenbarth, M., Schreck, T., Echtler, F., Arp, D., & Hoffmann, J. (2014). Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques. Int. J. Inf. Secur., 14(2), 141–153.
[20] Pieterse, H. & Olivier, M. S. (2012). Android botnets on the rise: Trends and characteristics. Proceedings of the IEEE Information Security for South Africa, pp. 1-5.
[21] Karim, A., Ali Shah, S. A., & Salleh, R. (2014). Mobile botnet attacks: A thematic taxonomy. In A. Rocha, A. Correia, F. Tan, & K. Stroetmann (Eds.), New Perspectives in Information Systems and Technologies. Cham: Springer, pp. 153–164.
[22] Kadir, F. A., Stakhanova, N., & Ghorbani, A. A. (2015). Android Botnets: What URLs are telling us. Proceedings of the 9th International Conference, Network and System Security, pp. 78–91.
[23] Pieterse, H. & Burke, I. (2015). Evolution study of android botnets. Proceedings of the 10th International Conference on Cyber Warfare and Security, pp. 232–240.
[24] Choi, B., Choi, S. K., & Cho, K. (2013). Detection of mobile botnet using VPN. Proceedings of the 7th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput., pp. 142–148.
[25] Abdullah, Z., Saudi, M. M., & Nor Badrul, A. (2017). ABC: Android botnet classification using feature selection and classification algorithms. Adv. Science Letter, 23(5), 4717–4720.
[26] Tansettanakorn, C., Thongprasit, S., Thamkongka, S., & Visoottiviseth, V. (2016). ABIS: A prototype of Android Botnet Identification System. Proceedings of the 5th ICT Int. Student Proj. Conf. ICT, pp. 1–5
[27] Karim, A., Salleh, R., & Shah, S. A. A. (2015). DeDroid: A mobile botnet detection approach based on static analysis. Proceedings of the IEEE 12th Intl Conf Ubiquitous Intell. Comput., pp. 1327–1332.
[28] Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., & Rieck, K. (2014). Drebin: Effective and explainable detection of android malware in your pocket. Proceedings of the Symp. Netw. Distrib. System Security, pp. 23–26.
[29] Yusof, M., Saudi, M. M., & Ridzuan, F. (2017). A New Mobile Botnet Classification based on Permission and API Calls. Seventh International Conference on Emerging Security Technologies, pp. 122–127.
[30] Li, Z., Sun, L., Yan, Q., Srisa-an, W., & Chen, Z. (2017). DroidClassifier: Efficient adaptive mining of application-layer header for classifying android malware. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 198, 597–616.
[31] Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y. van der Venn, V., & Platzer, C. (2014). ANDRUBIS - 1,000,000 apps later: A view on current android malware behaviors. Proceedings of the 3rd Int. Work. Build. Anal. Datasets Gather. Exp. Returns Secur., pp. 3–17.
[32] Talha, K. A., Alper, D. I., & Aydin, C. (2015). APK auditor: Permission-based Android malware detection system. Digit. Investig., 13, 1–14.
[33] VirusTotal. Free online virus, malware and URL scanner, https://www.virustotal.com/.
[34] ApkTool. (n.d.). A tool for reverse engineering Android apk files, https://ibotpeaches.github.io/Apktool/.
[35] Pxb1988. dex2jar - Tools to work with android .dex and java .class files, https://sourceforge.net/p/dex2jar/wiki/Home/.
[36] Google. Manifest.permission | Android Developers, https://developer.android.com/reference/android/Manifest.permission.html.
[37] Chan, P. P. K., & Song, W. (2014). Static detection of android malware by using permissions and API calls. Proceedings of the International Conference on Machine Learning and Cybernetics, pp. 82–87.
[38] Feizollah, A., Anuar, N. B., Salleh, R., Amalina, F., Ma'arof, R. R., & Shamshirband, S. (2013). A study of machine learning classifiers for anomaly-based mobile botnet detection. Malaysian J. Comput. Sci., 26(4), 251–265.
[39] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., & Bringas, P. G. (2012). On the automatic categorisation of android applications. Proceedings of the IEEE Consum. Commun. Netw., pp. 149–153.