Android Botnet Detection Using Risk Assessment

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

[1]Faculty of Science and Technology (FST),

[2]Cybersecurity and System Research Unit, Islamic Science Institute (ISI),

Universiti Sains Islam Malaysia, 71800 Nilai, Negeri Sembilan, MALAYSIA.

Abstract

The increasing popularity of Android mobile phones in recent years has attracted the attention of malware developers. Android applications (apps) pose many risks/threats to the user's privacy and system integrity. Currently, permission-based models are used in the Android systems to detect the dangerous apps that possess several weaknesses. In this paper, a new risk assessment method is proposed to evaluate the amount of risk associated with every app in terms of privacy risk, financial risk, and smartphones system risk. It focused on the GPS exploitation for Android botnet detection. The assessment was based on static analysis that used features set permission and API calls. The quantitative calculation model was used as a method to differentiate between the benign and botnet apps. Every app was assessed for risk based on five categories such as Very High, High, Medium, Low and Very Low. Two datasets with 2694 Android botnet samples from Drebin and 774 benign apps from Google Play were used to evaluate the effectiveness of this method. The obtained results demonstrate that the proposed method is good in differentiating the Android botnet and benign apps based on the risk level. This will give a promising impression to the users during apps installation.

Keywords: Android, Android Botnet, Risk Assessment, Threat Level

Introduction

In recent years, smartphones or mobile devices become the most important gadgets of modern life due to their offered capabilities. Smartphones make people's daily life activities very easy such as bill payment via online banking, obtaining important information via web browsing, connecting people via social networking, and entertainment via online gaming (Yusof, Mohd Saudi, & Ridzuan, 2017). According to Gartner (Gartner, 2018), 1.5 billion smartphones were sold worldwide in the year 2017, which was an increase of 2.7% from the year 2016. Gartner also released an operating system (OS) market shares report, which revealed that Android dominated the market

with 86% of the total market share in 2017, which was an increase of 1.1% from the last year.

The popularity and adoption of mobile devices due to the offered capabilities and rich features attracted the malware writers to target the vulnerable, especially on Android (Felt, Finifter, Chin, Hanna, & Wagner, 2011; La Polla, Martinelli, & Sgandurra, 2012). Smartphones provide lots of the facilities of traditional personal computers (PCs). People also prefer to use the smartphones for financial activities and store sensitive data rather than in PCs (Yusof, M. Saudi, & Ridzuan, 2017). The authors in (Felt, Finifter, et al., 2011) concluded that the most common malware activities were collecting user information (61 %) and sending premium-rate SMS messages (52 %). The main purpose of malware is because of the novelty or amusement, credential theft, SMS spam, search engine optimization fraud, and ransom. According to Symantec internet threat report 2018 (Symantec, 2018), the number of new mobile malware variants was increased by 54 % in 2017 compared to 2016 as shown in Figure 1. They found that an average of 24,000 malicious mobile applications were blocked each day.

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 (Pieterse, 2014). 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 (Geng et al., 2012). The infected mobile devices then can be used by botmaster for cyber-crimes or cyberattacks, such as sending spam messages, interruption, denial of services (DoS) and collecting sensitive information which can be exploited for illegal purposes. Mobile botnets are presently posing serious threats for both end users and cellular networks (Xiang, Binxing, Lihua, Xiaoyi, & Tianning, 2011).

There are three types of installation techniques of malicious apps to infect the device namely Repackaging, Update Attack and Drive-by Download (Zhou & Jiang, 2012). One of the most common techniques adopted by malware developers is Repackaging that installs a malicious application on the Android Platform. It involves the downloading of a popular app, disassembling them, adding malicious code, repacking the code and uploading again. Users may be vulnerable and enticed to download and install these infected apps. Update Attack involves the addition of the malicious code in an update component that will download the malware at run time instead of directly including it in the application. The third technique Drive-by Download is very effective for malware penetration where it tricks the users to download the malicious apps through the online site contaminated with malicious content when during their visit to those sites.

Nowadays, apps for mobile devices are distributed through online marketplaces, such as Google Play or App Store. In contrast to iPhone users, Android users can download an app not only from the official Google Play Store, but they can also download them from any third-party apps markets, torrents or direct downloads from the Internet. Study from Lindorfer et al. (Lindorfer, Volanis, et al., 2014) showed that 5 to 8 % malicious apps were available in the third-party market. They studied and analyzed eight thirdparty marketplaces. They found that a large number of ad-aggressive apps, malicious apps, and some markets even allowed the authors to publish known malicious apps without prompt action (Lindorfer, Volanis, et al., 2014). In the third-party marketplace, apps developer can openly publish any malicious or benign applications. Normally, these third-party or unofficial markets provide free, non-paying apps or cheaper apps as compared to Google Play Store. This attracts more users to use the third-party market and thus, it can possibly expose the smartphone users to download and install malicious apps from this market (Yusof, Saudi, & Ridzuan, 2017b). The risk assessment method is needed before the installation for the existing apps or while downloading the apps from third-party markets, to evaluate the reliability of the apps.

Although many researches were done on mobile malware, however research on mobile malware, especially on mobile botnet for risk assessment is still very less. Hence, this research aimed to fill the gap of the existing studies by developing a risk assessment method to evaluate the risk level of Android apps in terms of privacy, financial and system by focusing on the GPS exploitation. GPS is among the top surveillance functions that is always manipulated by attackers to launch an attack to the smartphone users. Attackers will use botnet threats because botnets are closely related to locations and GPS functions. This research proposed a risk assessment via mathematical analysis to evaluate the app with the given threat score value and risk level as Very Low, Low, Moderate, High and Very High as constructed in [13-14]. The detailed discussion is in Section 3.

The rest of this paper is organised as follows: Section 2 presents the related works. Section 3 introduces with the used methodology. Section 4 discusses the obtained results and general findings of the analysis. Lastly, Section 5 concludes this study and also provides idea for future works.

Related Work

Android malware detection approaches can be classified as static, dynamic and hybrid analysis (Feizollah, Anuar, Salleh, & A Wahab, 2015). Static analysis is a technique where an application is analysed without running it. The source code was examined by using the reverse engineering techniques to extract certain features from the applications.

Static analysis technique was implemented in several research works for mobile malware detection (Arp, Spreitzenbarth, Malte, Gascon, & Rieck, 2014; Sanz, Santos, Laorden, Ugarte-Pedrero, Nieves, et al., 2013; Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012; Yerima, Sezer, McWilliams, & Muttik, 2013). Meanwhile, dynamic analysis involves execution of the application to observe its behavior and it relies on runtime behaviors to judge whether the sample is malicious (Burguera, Zurutuza, & Nadjm-Tehrani, 2011; Dini, Martinelli, Saracino, & Sgandurra, 2012; Rastogi, Chen, & Enck, 2013). The analysis by third type is a hybrid analysis that combines the features of static and dynamic techniques.

Permission features for static analysis are the most popular malware detection approaches among the researchers due to the lightweight technique that consumes less system resources (Aung & Zaw, 2013; Sahs & Khan, 2012; Sanz, Santos, Laorden, Ugarte-Pedrero, Bringas, et al., 2013; Sanz, Santos, Laorden, Ugarte-Pedrero, Nieves, et al., 2013). Ye et. al (Ye, Wu, Hong, & Huang, 2017) analyzed more than 30,000 samples and they found that the feature like permission, APIs, and intent are not effective to distinguish malware from benign apps. There were only few differences in between the used malware and benign app for which the study was ended up with high false positives and false negatives rate.

To overcome the disadvantages of existing static analysis based approaches in which the used features are not effective to distinguish malware from benign Android applications, some researchers attempted to detect Android malware using application risk classification. DroidRisk (Wang, Zheng, Sun, & Mukkamala, 2013) is a framework of security risk assessment for both Android permissions and apps based on permission request patterns. One of the design goals of DroidRisk is to improve the user attention about the risks of Android permissions and apps. Matthew et al. (Mathew & Joy, 2015) proposed static and dynamic analysis to analyse the risk for apps permission. The Learner and Analyzer Systems were introduced in static analysis to extract the permission; and show the permission usage, detailed risk summary reports, along with risk ratings on the scale of 5, percentage and permission details. Users can review the apps based on this risk score and malware pattern description. Unfortunately, the authors do not compare and evaluate the proposed system to show the effectiveness of the system.

Sarma et al. (Sarma, Li, Gates, Potharaju, & Nita-rotaru, 2012) proposed Rare Critical Permissions (RCP) and Rare Pairs of Critical Permissions (RPCP) to calculate the risk of an app. They identified Android malware based on the observations and it revealed that the apps in the same category requested similar permissions. They concluded that an Android app might be suspicious as malware if it requested permissions that matched the

RCP list to trigger the RPCP. Peng et al. (Peng et al., 2012) proposed Probabilistic Generative Model of risk scoring and risk ranking for Android apps. They applied several Bayes models to calculate the generative probability of an app through the probabilities of each requested permission. Then generative probability was converted to a risk score by the risk scoring function. The obtained result of risk score was put in order to measure the risk of an app. They concluded that the probabilistic general models significantly outperformed the earlier approaches, and Naive Bayes model provided a promising risk scoring approach.

Dini et al. (Dini et al., 2016) introduced MAETROID (Multi-criteria App Evaluator of TRust for AndrOID) which is a framework to evaluate the trustworthiness of Android apps targeting the end users during the time of deployment and before its installation. It started from a set of criteria based on the permission features. When these criteria are combined, it is called Analytic Hierarchy Process (AHP). After the evaluation, MAETROID labelled the app as either trusted, medium risk or high risk. They also introduced the global threat score, which was based on a weighted and the normalized sum of a threat level. The evaluation and the calculation of the threat level was based on three security parameters such as privacy, financial and system. The authors also added this framework acts at deploy-time on the user devices and this app did not require the code to be decompiled and analyzed.

Ye et al. (Ye et al., 2017) proposed a risk classification based approach for Android malware detection. They classified the application risks into the following five specific categories: money, privacy, sensitive program behavior, network connection and highly dangerous permission. Afterwards, a quantitative calculation model was proposed to compute the comprehensive risk based on these categories. The result indicated that the risk classification approach achieved a high accuracy with a low false positive rate using the Random Forest algorithm. Both of the researches were carried out based on the permission feature using risk assessment and classification to rank the risk apps for mobile malware detection. Both used mathematical calculation model to differentiate the malware and benign apps. According to Ye et al. (Ye et al., 2017), the differences between benign apps and malware for the requested permissions were very less. Therefore, it is evident that the previous approaches have a weakness for which other methods are needed to improve. This study considered API calls as a selected feature besides the permission features to solve the weakness in the previous works.

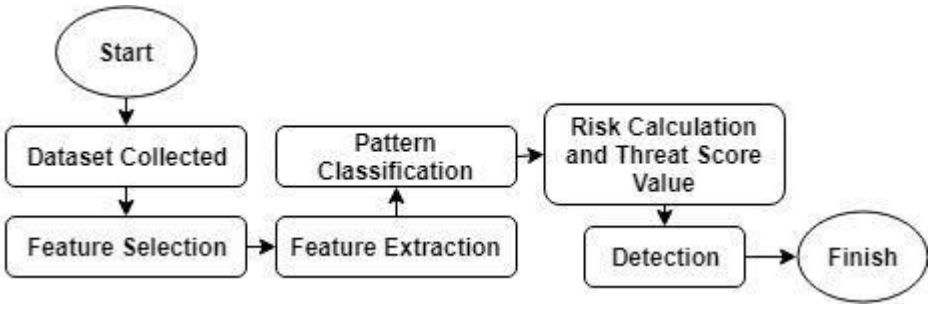
The most relevant research was conducted by Abdullah and M. Saudi (Abdullah & Mohd Saudi, 2017) and it was a study on Android botnet detection by using risk assessment method based on permission and API features. They applied the best 20

permission and API calls to compute with the risk level score based on privacy, financial and system to differentiate between malware and benign. In contrast, this research aimed to extend the research by Abdullah and M. Saudi to propose a new risk assessment method by focusing on GPS and location related features to detect the mobile botnet apps.

Methodology

This research proposed risk assessment method using risk score level based on privacy, financial and system to detect Android botnet. Static analysis was used to extract all the Android botnet samples from two types of dataset. The mobile botnet dataset is from Drebin (Arp et al., 2014) and the benign samples were downloaded from Google Play Store. The main part of this research is to detect all the apps of both datasets, whether benign or botnet, by using risk assessment method based on the permission and API call features. The overview process of the analysis is shown in Figure 1.

Figure 1: Risk Assessment Flow Process



Dataset

Two types of datasets (benign and malware) were used in this study. Drebin [28] dataset that consists of 5,560 malwares from 179 different families were used in this research as malware dataset. Based on the investigation of Android botnet features and earlier works from other researchers (Abdul Kadir, Stakhanova, & Ghorbani, 2015; Jiang, 2011b, 2011c, 2011a; Karim, Salleh, & Shah, 2015; Pieterse & Burke, 2015; Pieterse & Olivier, 2012; Spreitzenbarth, n.d.; Zhou & Jiang, 2011, 2012), this study collected 2694 botnet samples belonged to 44 different botnet families from Drebin dataset. Nowadays, Drebin dataset is considered as the largest publicly available dataset used in most of mobile malware research as was used in studies (Lindorfer, Neugschwandtner, et al., 2014; Talha, Alper, & Aydin, 2015; Yusof, Saudi, & Ridzuan, 2017a). The list of botnet

family is tabulated in Table 1. While for the testing and benign dataset, 774 apps of random categories were downloaded from the Google Play Store.

Table 1: Android Botnet Families

Family	Samples	Family	Samples
DroidKungFu	653	BeanBot	8
Plankton	474	Coogos	8
GingerMaster	332	Ksapp	6
BaseBridge	314	Fjcon	5
Kmin	144	Gamex	5
Iconosys	133	Nickspy	4
Geinimi	80	PdaSpy	4
Adrd	78	RootSmart	4
DroidDream	78	Stiniter	4
MobileTx	68	Biige	3
GoldDream	66	Luckycat	3
Imlog	43	SeaWeth	3
YZHC	36	Ackposts	2
Dougalek	17	Fidall	2
Fatakr	17	JSmsHider	2
Steek	14	Loozfon	2
Zitmo	14	Mobilespy	2
Vdloader	13	Saiva	2
FakeTimer	12	Acnetdoor	1
Nandrobox	12	Booster	1
Placms	12	PJApps	1
Cosha	11	Updtbot	1
Total			2694

Feature Selection

According to Feizollah et al. [10], feature selection plays an important part for Android malware and botnets detection. It reduces the noisy and irrelevant data from datasets to produce higher accuracy results for machine learning algorithms [10]. Although this study is not directly related to machine learning, but the accuracy rate depends on the feature selection to separate the botnet and benign samples.

As mentioned earlier, permission alone is not enough to differentiate between malware and benign apps and another feature like API calls is required to contribute in the detection accuracy. There are 147 permissions in the Android manifest file for the latest API level 26 [31]. However, not all of these permissions were requested by Android application in the dataset. To choose the most relevant feature with GPS related function,

this study made a comparison on certain criteria such as repackaging an apps, receiving command, messaging, stealing information, third party apps market, downloading additional content and modifying the Android Manifest file (Abdul Kadir et al., 2015; Jiang, 2011b, 2011c, 2011a; Karim, Shah, et al., 2015; Karim, Salleh, et al., 2015; Nigam, 2015; Pieterse, 2014; Pieterse & Burke, 2015; Pieterse & Olivier, 2012; Zhou & Jiang, 2011, 2012). Features from permission and API calls which having related with GPS exploitation and the impact to the privacy, financial and system categories were selected during the analysis. Yusof et al. listed out the top 20 permission and API calls features that were related with Android botnet and GPS exploitation (Yusof et al., 2017; Yusof, Saudi, et al., 2017b). As a result, from the comprehensive review and analysis, this research selected 12 features for permissions and 30 features for API calls. These are the most important and related features with Android botnet, GPS, and location function having the impact of risk factor as shown in Table 2.

Table 2: List of Permission and API Calls Features Selection

Code	Permissions	Code	API Calls
p1	ACCESS_COARSE_LOCATION	a1	startService()
p2	ACCESS_FINE_LOCATION	a2	getBestProvider()
p3	ACCESS_NETWORK_STATE	a3	getLastKnownLocation()
p4	ACCESS_WIFI_STATE	a4	isProviderEnabled()
p5	INTERNET	a5	requestLocationUpdates()
p6	READ_PHONE_STATE	a6	getActiveNetworkInfo()
p7	READ_SMS	a7	getAllNetworkInfo()
p8	RECEIVE_BOOT_COMPLETED	a8	getNetworkInfo()
p9	RECEIVE_SMS	a9	getConnectionInfo()
p10	SEND_SMS	a10	getWifiState()
p11	WRITE_EXTERNAL_STORAGE	a11	isWifiEnabled()
p12	WRITE_SMS	a12	setWifiEnabled()
		a13	android/telephony/gsm/SmsManager;->sendTextMessage()

a14android/telephony/SmsManager;->sendTextMessage()
a15getCellLocation()
a16getDeviceId()
a17getLine1Number()
a18getSimSerialNumber()
a19getSubscriberId()
a20getSystemService
a21HttpPost
a22Exec()
a23java/net/HttpURLConnection;->connect()
a24getContent()
a25openConnection()
a26openStream()
a27java/net/URLConnection;->connect
a28getInputStream()
a29Execute()
a30sendSMS

Feature Extraction

This study applied static analysis technique for Android botnet feature extraction. This involved permission and API calls that were also used by previous researchers (Chan & Song, 2014; Peiravian & Zhu, 2013; Wu, Mao, Wei, Lee, & Wu, 2012). For static analysis, the features of permissions and API calls were extracted by using a reverse engineering tool like Apktool ("A tool for reverse engineering Android apk files," n.d.) and dex2jar (Pxb1988, n.d.). Permissions features were extracted from Manifest.xml file and API calls were extracted from .dex class file.

Then, a macro script was applied with the string similarity method to extract permissions and API calls. For each sample, when the requested permission or API call matched with the Android permission and API call, it represented as 1 to indicate its presence in the sample and 0 indicated its absence in the sample. R as considered a vector containing a set of 147 Android permissions. For every i_{th} application in the Android application dataset (botnet and benign), $R_i = \{r_1, r_2, r_3, \ldots r_j\}$ and

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

After that, the extraction process was applied on the Drebin (malicious) and benign datasets to extract permissions and API calls features from all the samples. Out of 5,560 malware samples of the Drebin dataset, only 2694 botnet samples were selected. The researchers performed reverse engineering and applied macro script with the string similarity method to compare 3468 samples (malicious and benign) with 12 permissions and 30 API calls feature sets. The result from this extraction process was transformed into vector in comma separated value (CSV) format file. The value of vector starts either with the value 1 or 0, which indicated the presence or absence of the samples respectively for the feature set. Top 25 mostly used features by Android botnet are illustrated in Table 3.

Table 3: Top 25 Features with the Frequency

Features	Frequency (%)
INTERNET	98.26
READ_PHONE_STATE	95.69
getSystemService	90.87
getDeviceId()	90.53
openConnectio()	89.68
ACCESS_NETWORK_STATE	81.03
startService()	80.96
getActiveNetworkInfo()	76.24
WRITE_EXTERNAL_STORAGE	71.16
java/net/HttpURLConnection;->connect	68.89
HttpPost	68.19
getSubscriberId()	65.14
getLine1Number()	61.84
ACCESS_WIFI_STATE	61.32

getNetworkInfo()	58.24
RECEIVE_BOOT_COMPLETED	55.75
execute()	50.78
ACCESS_COARSE_LOCATION	50.45
getInputStream()	47.48
ACCESS_FINE_LOCATION	46.84
getSimSerialNumber()	45.36
exec()	44.99
java/net/URLConnection;->connect()	43.95
getLastKnownLocation()	43.58

Feature Classification

In this phase, the obtained results of features selection and feature extraction were transformed into a feature vector table and accordingly classification pattern was generated. As a result, 792 new pattern classifications for Android botnet with GPS exploitation based on permissions and API calls were developed. The patterns were used to calculate the risk score and risk level for each app.

Threat Score

To develop a threat score, every feature evaluated the level of threat with respect to three security parameters such as user privacy, users' financial loss and smartphone system downgrading. According to Ye et al. (Ye et al., 2017), privacy risk refers to potential leakage of user's confidential and private data when features (permission and API calls) are granted and executed. Financial refers to the potential financial losses of the victim such as user's mobile credit. Meanwhile, the system refers to the potential degradation of smartphone system's performance and unauthorized modification of its storage and files (Ye et al., 2017). Every feature was given the threat level indication, threat score value and risk score as was done in (NIST, 2012b), (Dini et al., 2016; Ye et al., 2017). The details and explanation about threat value are explained in Table 4. The risk classification and threat level are presented in Table 5.

Table 4: Threat Level, Threat Score and Risk Score

Threat Level	Threat Score	Risk Score	Explanation
Very High	1	81-100	Highly related to GPS and seriously affecting the privacy, financial and system of the device's owner and of other users connected to that owner. The exposed can cause serious damage to the users.
High	0.8	61-80	Related with GPS and affecting the privacy, financial and smartphone system of the user devices and of other users connected to that devices. The exposed can cause damage to the users.
Medium	0.6	41-60	Related with GPS and may definitely affect the user's privacy, financial and system with the combination with other features.
Low	0.4	21-40	Least related to the GPS and does not strongly affect the user's privacy, financial and system.
Very Low	0.2	0-20	Not related to the GPS but can affect user privacy, financial and system.

Table 5: Risk Classification, Threat Level and Threat Score

Features	Risk Classification	Threat Level	Threat Score
ACCESS_COARSE_LOCATION	Privacy	Very High	1
ACCESS_FINE_LOCATION	Privacy	Very High	1
READ_PHONE_STATE	Privacy	Very High	1
READ_SMS	Privacy	High	0.8
WRITE_SMS	Privacy	Medium	0.6
ACCESS_WIFI_STATE	Privacy	Low	0.4
RECEIVE_SMS	Privacy	Low	0.4
getDeviceId()	Privacy	Very High	1
getLine1Number()	Privacy	Very High	1
getSimSerialNumber()	Privacy	Very High	1
getSubscriberId()	Privacy	Very High	1
getLastKnownLocation()	Privacy	Medium	0.6
requestLocationUpdates()	Privacy	Medium	0.6
getBestProvider()	Privacy	Low	0.4

isProviderEnabled()	Privacy	Low	0.4
getWifiState()	Privacy	Low	0.4
setWifiEnabled()	Privacy	Low	0.4
getCellLocation()	Privacy	Low	0.4
getContent()	Privacy	Low	0.4
getConnectionInfo()	Privacy	Very Low	0.2
isWifiEnabled()	Privacy	Very Low	0.2
SEND_SMS	Financial	Medium	0.6
ACCESS_NETWORK_STATE	Financial	Very Low	0.2
INTERNET	Financial	Very Low	0.2
getActiveNetworkInfo ()	Financial	Medium	0.6
getNetworkInfo()	Financial	Low	0.4
android/telephony/gsm/SmsManager;->sendTextMessage	Financial	Low	0.4
android/telephony/SmsManager;->sendTextMessage	Financial	Low	0.4
sendSMS()	Financial	Low	0.4
getAllNetworkInfo()	Financial	Very Low	0.2
RECEIVE_BOOT_COMPLETED	System	High	0.8
WRITE_EXTERNAL_STORAGE	System	Very Low	0.2
startService()	System	High	0.8
HttpPost	System	High	0.8
openConnection()	System	High	0.8
java/net/URLConnection;->connect	System	High	0.8
execute()	System	High	0.8
getSystemService	System	Medium	0.6
exec()	System	Medium	0.6
getInputStream()	System	Medium	0.6
java/net/HttpURLConnection;->connect	System	Low	0.4
openStream()	System	Very Low	0.2

Risk Calculation

All apps are then computed with the selected features and threat value score based on Equation (2).

$$RSP = \frac{fp.tsp}{F_{fp}}$$

$$RSF = \frac{ff.tsf}{F_{ff}}$$

$$RSS = \frac{sf.tss}{F_{fs}} \quad (2)$$

where: RSP = Risk score for privacy
RSF = Risk score for finance
RSS = Risk score for system
fp = privacy related features
ff = financial related features
fs = system related features
tsp = threat score for privacy
tsf = threat score for financial
tss = threat score for system
F = Frequency of features in an app

Based on Formula (2) then the study produces Total Risk Score (TRS) which is a normalization of risk score of every apps based on privacy, financial and system. The formula is shown in Equation (3).

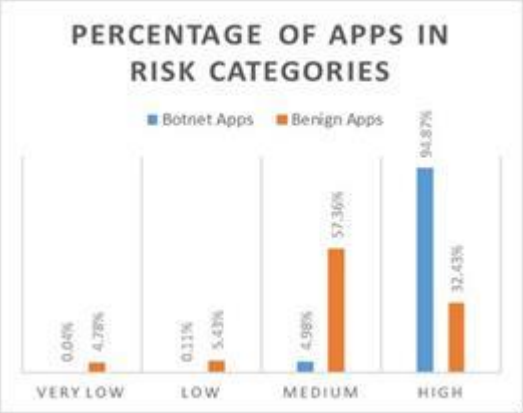
$$TRS = \frac{RSP + RSF + RSS}{1 + (RSP + RSF + RSS)} x\ 100 \quad (3)$$

Results and Discussion

As discussed in the previous section, two datasets were used to evaluate this proposed method. All the 3468 apps (botnet and benign) were evaluated and calculated by using Total Risk Score (TRS) and this research approach focused more on GPS exploitation by using all GPS or location related features. TRS was formulated as per Equation (3). The result is summarized in Figure 2. Out of 2694 Android botnet samples, it was found that 94.87 % was categorized as High Risk, 4.98 % samples were at Medium Risk and 0.11 % samples were at Low Risk. Meanwhile, from 774 Android benign samples, it was found that 32.43 % was categorized as High Risk, 57.36 % samples were at Medium

Risk, 5.43 % samples were at Low Risk and 4.78 % were at Very Low Risk. Therefore, it is evident from the result that the assessment method has successfully differentiated botnet apps and benign apps.

Figure 2: Android Apps Risk Categories:



Most of the benign samples that were collected from communications and utilities categories, used the GPS or location related features for which the number of high risk categories in these samples were high. Another reason behind this higher number was the use of over-privileged permissions features by many app developers during the development of the apps (Wang et al., 2013). It was observed that that 44 % of 50 randomly selected apps from Google play store were over-privileged. This is supported by Fetl et al. (Felt, Chin, Hanna, Song, & Wagner, 2011) who investigated 940 Android apps using Stowaway tool and found that around one-third of those apps were overprivileged. The result shows that this experiment is successful in differentiating botnet and benign samples.

Conclusion and future work

This study proposed a risk assessment method based on privacy risk, financial risk and system risk focusing on the GPS exploitation for Android botnet detection. All samples were classified, computed and calculated by using Total Risk Score formula. This method has successfully differentiated between Android botnet apps and benign apps. The results show the capability of this method in detecting the Android malware at a satisfying accuracy rate. In addition, this research planned to integrate the dynamic analysis for system call extraction along with the better detection and accuracy rate. Since in this research the number of High Risk detection from benign dataset was high, hence for future work it is suggested to select the benign dataset with equal number of data in

every category such as utilities, games, entertainment, news, productivity and social networking so that the outcome can be improved. This paper is part of a larger project to develop an automated Android botnet detection and response model.

References

A tool for reverse engineering Android apk files. (n.d.). Retrieved May 24, 2017, from https://ibotpeaches.github.io/Apktool/

Abdul Kadir, A. F., Stakhanova, N., & Ghorbani, A. A. (2015). Android Botnets: What URLs are Telling Us. In *9th International Conference, Network and System Security 2015 New York, USA* (Vol. 9408, pp. 78–91).

Abdullah, Z., & Mohd Saudi, M. (2017). RAPID - Risk Assessment of Android Permission and Application Programming Interface (API) Call for Android Botnet. In *In Proceeding of the 2017 International Conference on Information Systems & Security (ICoISS2017)*.

Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., & Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. *Symposium on Network and Distributed System Security (NDSS)*, 23–26. http://doi.org/10.14722/ndss.2014.23247

Aung, Z., & Zaw, W. (2013). Permission-Based Android Malware Detection. *International Journal of Scientific & Technology Research*, *2*(3), 228–234.

Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-Based Malware Detection System for Android. *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '11*, 15. http://doi.org/10.1145/2046614.2046619

Chan, P. P. K., & Song, W. (2014). Static Detection of Android Malware by Using Permissions and API Calls. In *2014 International Conference on Machine Learning and Cybernetics* (pp. 82–87).

Dini, G., Martinelli, F., Matteucci, I., Petrocchi, M., Saracino, A., & Sgandurra, D. (2016). Risk Analysis of Android Applications: A User-Centric Solution. *Future Generation Computer Systems*, *80*, 505–518. http://doi.org/10.1016/j.future.2016.05.035

Dini, G., Martinelli, F., Saracino, A., & Sgandurra, D. (2012). MADAM: A multi-level anomaly detector for android malware. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7531 LNCS*, 240–253. http://doi.org/10.1007/978-3-642-33704-821

Feizollah, A., Anuar, N. B., Salleh, R., & A Wahab, A. W. (2015). A review on feature selection in mobile malware detection. *Digital Investigation*, *13*, 22–37. http://doi.org/10.1016/j.diin.2015.02.001

Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). Android permissions demystified. *Proceedings of the 18th ACM Conference on Computer and Communications Security - CCS '11*, 627. http://doi.org/10.1145/2046707.2046779

Felt, A. P., Finifter, M., Chin, E., Hanna, S., & Wagner, D. (2011). A survey of mobile malware in the wild. *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '11*, 3. http://doi.org/10.1145/2046614.2046618

Gartner. (2018). Gartner Says Worldwide Sales of Smartphones Recorded First Ever Decline During the Fourth Quarter of 2017. Retrieved April 27, 2018, from https://www.gartner.com/newsroom/id/3859963

Geng, G., Xu, G., Zhang, M., Guo, Y., Yang, G., & Cui, W. (2012). The design of SMS based heterogeneous mobile botnet. *Journal of Computers*, *7*(1), 235–243. http://doi.org/10.4304/jcp.7.1.235-243

Jiang, X. (2011a). Security Alert: New Android SMS Trojan -- YZHCSMS -- Found in Official Android Market and Alternative M. Retrieved May 2, 2017, from www.csc2.ncsu.edu/faculty/xjiang4/YZHCSMS/

Jiang, X. (2011b). Security Alert: New Sophisticated Android Malware DroidKungFu Found in Alternative Chinese App Markets. Retrieved May 2, 2017, from www.csc2.ncsu.edu/faculty/xjiang4/DroidKungFu.html

Jiang, X. (2011c). Security Alert: New Stealthy Android Spyware -- Plankton -- Found in Official Android Market. Retrieved May 2, 2017, from www.csc2.ncsu.edu/faculty/xjiang4/Plankton/

Karim, A., Salleh, R., & Shah, S. A. A. (2015). DeDroid: A Mobile Botnet Detection Approach Based on Static Analysis. *EEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 1327–1332. http://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.240

Karim, A., Shah, S. A. A., Salleh, R. Bin, Arif, M., Md Noor, R., Shamshirband, S., … Noor, R. (2015). Mobile botnet attacks - an emerging threat: Classification, review and open issues. *KSII Transactions on Internet and Information Systems*, *9*(4), 1471–1492. http://doi.org/10.3837/tiis.2015.04.012

La Polla, M., Martinelli, F., & Sgandurra, D. (2012). A Survey on Security for Mobile Devices. *IEEE Communications Surveys & Tutorials*, *15*(1), 446–471. http://doi.org/10.1109/SURV.2012.013012.00028

Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., Venn, V. van der, & Platzer, C. (2014). ANDRUBIS - 1,000,000 Apps Later: A View on Current Android Malware Behaviors. *Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 3–17. http://doi.org/10.1109/BADGERS.2014.7

Lindorfer, M., Volanis, S., Sisto, A., Neugschwandtner, M., Athanasopoulos, E., Maggi, F., … Ioannidis, S. (2014). AndRadar: Fast Discovery of Android Applications in Alternative Markets. In S. Dietrich (Ed.), *Detection of Intrusions and Malware, and Vulnerability Assessment: 11th International Conference, DIMVA 2014, Egham, UK, July 10-11, 2014. Proceedings* (pp. 51–71). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-08509-8_4

Mathew, J. J., & Joy, M. T. (2015). Efficient Risk Analysis for Android Applications. In *Proceedings of the 2015 IEEE Recent Advances in Intelligent Computational Systems, RAICS 2015* (pp. 382–387). http://doi.org/10.1109/RAICS.2015.7488446

Nigam, R. R. (2015). A Timeline Of Mobile Botnets. *Spring*, (March), 1–8.

NIST. (2012a). Guide for Conducting Risk Assessments. *NIST Special Publication*, (September). http://doi.org/10.6028/NIST.SP.800-30r1

NIST. (2012b). Guide for Conducting Risk Assessments. *NIST Special Publication*, (September). http://doi.org/10.6028/NIST.SP.800-30r1

Peiravian, N., & Zhu, X. (2013). Machine learning for Android malware detection using permission and API calls. In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*. http://doi.org/10.1109/ICTAI.2013.53

Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., … Molloy, I. (2012). Using Probabilistic Generative Models for Ranking Risks of Android Apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security - CCS '12* (pp. 241–252). http://doi.org/10.1145/2382196.2382224

Pieterse, H. (2014). *Design of a Hybrid Command and Control Mobile Botnet*. University of Pretoria. Retrieved from http://researchspace.csir.co.za/dspace/handle/10204/6784

Pieterse, H., & Burke, I. (2015). Evolution Study of Android Botnets. In *Proceedings of the 10Th International Conference on Cyber Warfare and Security (ICCWS2015)* (pp. 232–240).

Pieterse, H., & Olivier, M. S. (2012). Android botnets on the rise: Trends and characteristics. *2012 Information Security for South Africa - Proceedings of the ISSA 2012 Conference*. http://doi.org/10.1109/ISSA.2012.6320432

Pxb1988. (n.d.). dex2jar - Tools to work with android .dex and java .class files. Retrieved from https://sourceforge.net/p/dex2jar/wiki/Home/

Rastogi, V., Chen, Y., & Enck, W. (2013). AppsPlayground : Automatic Security Analysis of Smartphone Applications. *CODASPY '13 (3rd ACM Conference on Data and Application Security and Privac)*, 209–220. http://doi.org/10.1145/2435349.2435379

Sahs, J., & Khan, L. (2012). A Machine Learning Approach to Android Malware Detection. *Intelligence and Security Informatics Conference*, 141–147. http://doi.org/10.1109/EISIC.2012.34

Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P. G., & Alvarez, G. (2013). PUMA: Permission usage to detect malware in android. *Advances in Intelligent Systems and Computing*, *189 AISC*, 289–298. http://doi.org/10.1007/978-3-642-33018-6_30

Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Nieves, J., Bringas, P. G., & Álvarez Marañón, G. (2013). Mama: Manifest Analysis for Malware Detection in Android. *Cybernetics and Systems*, *44*(6–7), 469–488. http://doi.org/10.1080/01969722.2013.803889

Sarma, B., Li, N., Gates, C., Potharaju, R., & Nita-rotaru, C. (2012). Android Permissions: A Perspective Combining Risks and Benefits. In *Proceedings of the 2012 ACM Conference on Symposium on Access Control Models and Technologies (SACMAT)* (pp. 13–22). http://doi.org/10.1145/2295136.2295141

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": A behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, *38*(1), 161–190. http://doi.org/10.1007/s10844010-0148-x

Spreitzenbarth, M. (n.d.). Forensic Blog. Retrieved March 1, 2018, from https://forensics.spreitzenbarth.de/android-malware/

Symantec. (2018). *Internet Security Threat Report 2018*. *ISTR*. Retrieved from https://academic.oup.com/jicru/article-lookup/doi/10.1093/jicru/ndl025

Talha, K. A., Alper, D. I., & Aydin, C. (2015). APK Auditor: Permission-based Android malware detection system. *Digital Investigation*, *13*, 1–14. http://doi.org/10.1016/j.diin.2015.01.001

Wang, Y., Zheng, J., Sun, C., & Mukkamala, S. (2013). Quantitative Security Risk Assessment of Android Permissions and Applications. In *27th Data and Applications Security and Privacy (DBSec)* (pp. 226–241).

Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., & Wu, K.-P. (2012). DroidMat: Android malware detection through manifest and API calls tracing. *Proceedings of the 7th Asia Joint Conference on Information Security, Asia JCIS*, 62–69. http://doi.org/10.1109/AsiaJCIS.2012.18

Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., & Tianning, Z. (2011). Andbot: towards advanced mobile botnets. *LEET'11 Proceedings of the 4th USENIX Conference on Large-Scale Exploits and Emergent Threats*, 11. Retrieved from http://dl.acm.org/citation.cfm?id=1972441.1972456

Ye, Y., Wu, L., Hong, Z., & Huang, K. (2017). A risk classification based approach for Android malware detection. *KSII Transactions on Internet and Information Systems*, *11*(2), 959–981. http://doi.org/10.3837/tiis.2017.02.018

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

Yusof, M., M. Saudi, M., & Ridzuan, F. (2017). A Systematic Review Analysis of of Mobile Botnet Detection for GPS Exploitation. *Advanced Science Letters*, *23*(5), 4696–4700. http://doi.org/10.1007/978-3-319-07674-4

Yusof, M., Mohd Saudi, M., & Ridzuan, F. (2017). A New Mobile Botnet Classification based on Permission and API Calls. In *Seventh International Conference on Emerging Security Technologies (EST)* (pp. 122–127).

Yusof, M., Saudi, M. M., & Ridzuan, F. (2017a). A New Android Botnet Classi fi cation for GPS Exploitation Based on Permission and API Calls. *Recent Advances in Electrical Engineering and Related Sciences: Theory and Application. Lecture Notes in Electrical Engineering. AETA 2017.*, *465*, 27–37. http://doi.org/10.1007/978-3-319-69814-4_3

Yusof, M., Saudi, M. M., & Ridzuan, F. (2017b). Mobile Botnet Classification By Using Hybrid Analysis. In *The 1st International Conference on Information Systems & Security (ICoISS2017)*.

Zhou, Y., & Jiang, X. (2011). An Analysis of the AnserverBot Trojan. *Security*, 1–17.

Zhou, Y., & Jiang, X. (2012). Dissecting Android Malware : Characterization and Evolution. *2012 IEEE Symposium on Security and Privacy*, (4), 95–109. http://doi.org/10.1109/SP.2012.16