Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

A LEARNING BASED FRAMEWORK FOR DETECTION OF ANDROID C&C ENABLED APPLICATIONS USING HYBRID ANALYSIS

Attia Qamar[1], Ahmad Karim[2], Shahaboddin Shamshirband[3,4]

[1]National College of Business Administration (NCBA), Pakistan
[2]Department of Information Technology, Bahauddin Zakariya University, Pakistan
[3]Department for Management of Science and Technology Development, Ton Duc Thang University, Ho Chi Minh City, Vietnam
4Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

attiaaqamar@gmail.com, ahmadkarim@bzu.edu.pk,
shahaboddin.shamshirband@tdtu.edu.vn

Abstract

Smartphone devices, particularly android devices used by billions of people everywhere in the world. Similarly, this increasing rate attracts mobile botnet attackers that is a network of interconnected nodes operated by command and control (C&C) method to expand malicious activities. At present, mobile botnet attacks carried Distributed denial of services (DDoS) that causes to steal sensitive data, remote access, spam generation, etc. Consequently, various approaches are defined in the literature to detect mobile botnet using static or dynamic analysis. In this paper, we have proposed a novel hybrid model, which is a combination of static and dynamic method that relies on machine learning to identify android botnet applications having C&C capability. The results evaluated through machine learning classifiers in which Random forest classifier outperform other ML techniques, i.e. Naïve Bayes, Support Vector Machine, and Simple logistics. Our proposed framework can achieve 97.48% accuracy in detecting such harmful applications. Furthermore, we highlight some research directions and possible solutions regarding botnet attacks for the entire community.

Keywords: Android botnet, Botnet detection, Hybrid analysis, Machine learning classifiers, Mobile malware

1. Introduction

Currently, smartphone devices have become an inseparable part of human's lives by holding every kind of information from banking to its network usage. Smartphone devices, especially android mobiles, are very popular due to its affordability, rich user environment, and its appealing applications such as maps, weather forecasting, GPS function, and many more. According to statcounter Globalstats [1] android operating system is having 42.26 % worldwide market share, which is more significant than as compare to other OS shown in figure 1. Furthermore, Statista [2] reported that android is the first leading store by having 3.8 million applications in the first quarter of 2018.

As a result, emerging rate of an open source android operating system is not ignored by malware writers as well as mobile botnet attacks have grown faster. Mobile botnet is a network of bots, which operated through command and control (C&C) method by the botmaster. A botnet attack is responsible for Distributed Denial of service (DDoS) attack, steal personal information, gain illegal services access, send emails, send SMS on premium rate number, and infect multiple mobile devices. First botnet *SmsHOw.U* was discovered in September 2010, and now several mobile botnets are available i.e. Geinimi, DroidKungFu ,GoldDream, NotCompatible, Fakeplay, Anseverbot etc [3]. A RottenSystem [4] botnet that is activated from 2016 to till now and affected ca. 5 million android devices in first quarter of 2018. Besides, MAcafee mobile threat report [5] shows malware attacks in 2018.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

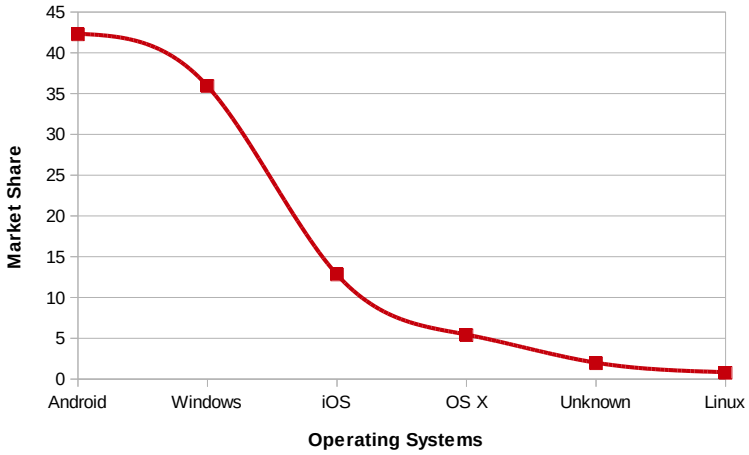

Figure 1. Operating system market share worldwide from January 2018 to July 2018[1]

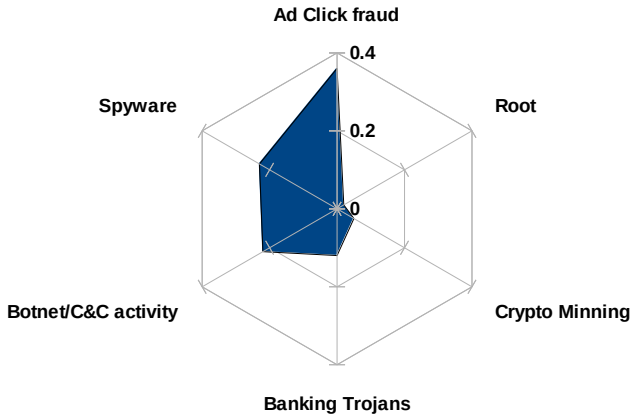

Figure 2. Malware attacks Growth Rate in the first quarter of 2018 [2]

In literature, many methods are used to detect android malware as they divided into two significant groups (a) static and (b) dynamic analysis. Static analysis refers to the code analysis while in dynamic method, analyze the behavior of an application at runtime. Kirubavathi and Anitha [6] proposed an android botnet detection model based on structural analysis having the feature of permissions and API calls. They take extensive data set to perform detection operation and achieved the highest accuracy with 99% using a classifier named as support vector machine. However, they did not cover dynamic features of android applications. Another researcher [7], introduced a dynamic detection model as SMARTbot for android botnet detection. They considered rich dynamic features such as network traces, DNS queries, file activities, etc. However, some limitations existed in both analysis methods. Static analysis is incompetent to detect obfuscated code that's why researches used dynamic analysis. Correspondingly, dynamic analysis is not able to deal with native code and did not provide full code coverage. Hence, in this paper, we used a hybrid analysis method that is a combination of static and dynamic analysis, which can cope with these challenges effectively.

In short, the aim of this paper is given by making the following contributions:

1. We propose a hybrid analysis framework with machine learning techniques to detect android botnet applications having C&C capabilities.
2. In this work, we investigate the most prominent static and dynamic features in perspective of malware authors used to target android applications. Specifically, we study C&C methods used in malicious code to generate an attack. Furthermore, we extract static and dynamic features such as permissions, API calls, file operations, and Network traces, respectively, and frequency analysis graphs are presented to distinguish botnet and benign applications.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

3. In order to validate results with existing approaches, we have used machine learning techniques to classify C&C specific application with a normal one. Among heterogeneous ML classifiers, random forest proved as the best classifier with 97.48% accuracy.
4. This proposed hybrid model compares well with other existing approaches that included static and dynamic techniques of mobile botnet detection[8],[9].
5. Finally, we have provided some guidelines to avoid such threatening hazard.

The rest of the paper divided into the following parts: section 2 gives the related work, briefly describe existing approaches that are used by researchers to detect botnet applications. Section 3 explains a proposed learning based hybrid detection model, features extraction, features selection, and datasets acquisition. Section 4 discusses results and compare it with existing botnet detection models. In the end, section 5 provides the overall conclusion of this paper.

2. Related Work

Mobile botnets are the new emerging threat for smartphone users. To avoid this hazard, researchers used static and dynamic analysis with the combination of machine learning algorithms to evaluate results. However, several studies have been carried out in the detection of mobile botnet attacks using static and dynamic analysis method, but they did not consider a hybrid method. In this section, we will discuss a few existing approaches that relate to this paper.

Yang et al. [10] introduced the multilevel feature extraction a dynamic method that contains (a) basic level features (b) traffic level features and (c) content level features to detect mobile botnet from APK files. Together with these features, network traffic that relates to HTTP collected from Anubis [11], CopperDroid [12], and Sandroid [] platforms. For the experiment, data is collected from Baidu app market [13] for normal while android Drebin project datasets for malicious apks [14]. Furthermore, each application is executed for 10 to 15 minutes to assemble the network traffic. Additionally, a machine learning classifier random forest is used to evaluate results, and it provides 93% true positive rate in the detection of botnet applications. In this study, we proposed a hybrid analysis framework for detection of C&C applications.

Similarly, Girei et al. [15] adopted dynamic analysis method and proposed a mobile botnet detection approach, which is named as “Logdog” by analyzing the log files. Logdog required root permissions to start, collect logs, stop, and then maintain a text file of log messages. Also, for analysis purpose expressions are already saved in logs that notify the botnet activity if they matched with the current malicious log. The authors experimented with actual android devices and android botnet application dendroid is used to communicate with C&C server via HTTP to send logs.

Moreover, in the work of [16] they carry out network behavior analysis in detection of HTTP based mobile botnet. A light application Tpacketcapture [17] is installed on mobile devices to capture the network traffic; collected data stored in a spontaneously connected data repository. Besides, network traffic features include domain name, source IP address, Destination IP address, and URL. Malicious data samples of botnet families such as Geinimi, AnseverBot, DroidDream, DroidkungFU [18] is collected from Genome Malware project [19]. Finally, results validate by using machine learning classifier Rule-Based and J48. Hence, a combination of rule-based classifiers and proposed periodic metrics shows 98.60% accuracy in mobile botnet detection. However, we take rich botnet application dataset having C&C ability with state-of-the-art mobile botnet applications sample.

Another study by al-detail et al. [20] presents Artificial immune system (AIS) with the combination of user activity to detect android botnet from social networking sites as Twitter and discriminate between user-generated tweets or bots caused. Besides, the proposed method worked in 4 steps: (a) capture tweets from twitter (b) relate twitter activity with user’s activity (c) create a signature for twitter activity (d) match with existing signatures if it does not match then assumed it legitimate tweet otherwise considers as bots generated tweet. For the experiment, datasets collected from Twitter stream API. As a consequence, AIS detectors gives 95% detection accuracy.

Anwer et al. [8] discussed the static method to detect those applications that have signs of a botnet. In this detection process, 1400 mobile botnet applications collected from ISCX android botnet datasets [21], and roguard used to extract static features. Several machine learning classifiers used to validate results. Hence, J-48 gives 94.1 % accuracy for broadcast services, which is better than other classifiers and SVM, and RF classifiers conducted an equal percentage with 93.4 for background services. However, in our proposed hybrid framework, we achieved 96.56% accuracy from static analysis with state-of-the-art mobile botnet samples.

Correspondingly, Yusof et al. [22] presented an inventive mobile botnet detection approach based on the static analysis by using permissions and API calls parameters. Most essential and conspicuous 31 API calls and 16 permissions extracted in the perception of mobile botnet authors used. To perform an experiment, they used reverse engineering techniques to extract features and make CSV file of them to a readable format. Finally, a random forest algorithm directed the highest accuracy, with 99.4% in the detection of mobile botnet applications. They did not consider any dynamic feature for analysis while on the other hand, we give a learning-based hybrid framework with a rich feature set.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

Moreover, Hijawi et al. [23] proposed a static analysis framework by considering permissions feature to detect android botnet. They elaborate the seven-level of permissions protection few of these are normal permissions, signature-based permissions, other permissions, and dangerous permissions. As well, intend to perform analysis they used botnet and benign dataset from ISCX android bot [21] and google play store [24] conversely. Eventually, they used Machine learning classifier (i.e., Random Forest, Naïve Bayes, J48, and Multilayer Perceptron) to verify the proposed method. Between these classifier models, Random Forest shows the best accuracy 97.3% by achieving highest botnet detection results. However, in this paper, we used a hybrid method to detect botnet applications based on machine learning algorithms, and it gives an advantage as compared to perform any single analysis approach, i.e. static or dynamic. In table 1 we provide a summary of existing approaches that are discussed in this section.

Table: 1 Summary of Related Work

References/Approaches	Year	Analysis Type	Features	Dataset Source	ML Classifiers/Clustering	Results
Multilevel feature extraction techniques [10]	2016	Dynamic analysis	Network Traffic (HTTP related)	Baidu app market for normal applications dataset & Android Drebin Project for malicious apks. Anubis, CopperDroid, SandDroid are a platform that is used to collect network traffic from apks.	Random Forest	0.93 TPR
[15]	2016	Dynamic analysis	Collect log files and analyze them	Logdog and Dendroid applications are used to collect log files		
[8]	2016	Static analysis	Permissions, broadcast Receiver, Background Services, MD5	UNB ISCX android botnet	SVM, Random Forest, K-NN,J-48	RM 93.4% accuracy for background services RM 92.7% accuracy for the broadcast receiver RF 93.4% for permissions
[16]	2016	Dynamic analysis	Network Traffic (HTTP)	Network Traffic collected by using Tpacketcapture for normal Traffic Genome Malware Project	J48, Rule-Based	The rule-based result shows 98.60% accuracy with a low rate of false positive
User Activity + Artificial Immune System [20]	2016	Dynamic analysis	Capture Tweets from Twitter	Twitter Public Stream API	AIS Detector	95% detection accuracy
[22]	2017	Static analysis	Permissions & API Calls	Drebin & Google Play Store	Random Forest	99.4% accuracy
[23]	2017	Static analysis	Permissions	ISCX Android botnet dataset, Google play store	J48,RF,MLP,NB	RF shows 97.3% accuracy

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

3. Methodology

In this section, we presented a learning-based hybrid detection framework that discriminates between benign and botnet applications having C &C capability. In fig 1 we illustrate the overall work flow of the proposed framework. There are four major modules in this study (a) Data Collection (b) Analysis(c) Features and (d) Machine learning module. We discuss these modules briefly in the following sections.

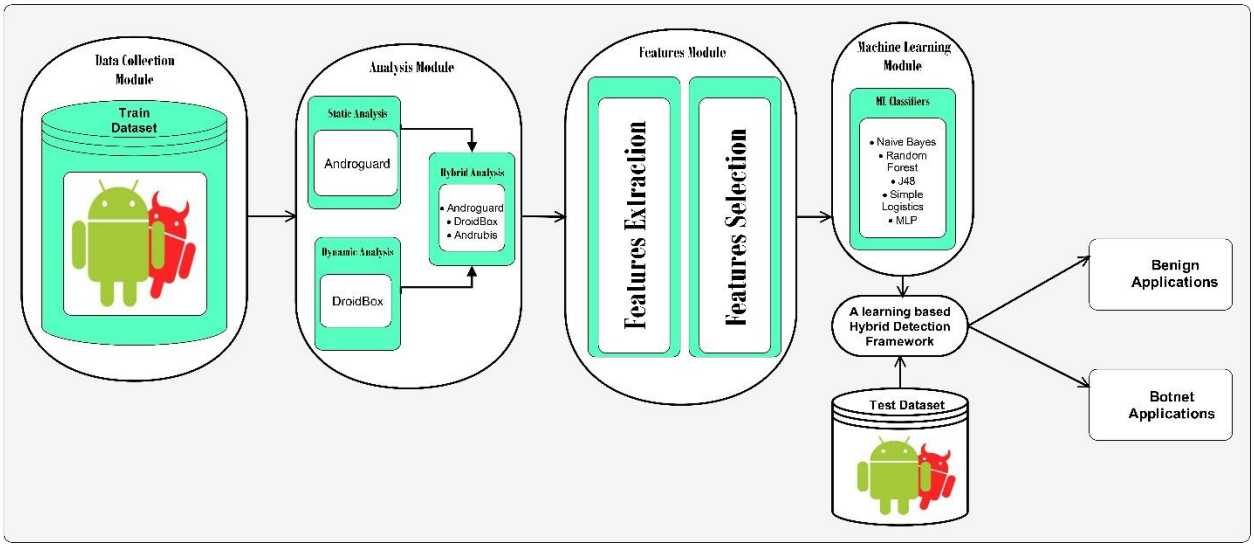

Figure 3. Proposed Hybrid Learning based Detection Framework

3.1 Data Collection Module

For experimental purpose, we collected benign as well as botnet dataset of android applications from different resources i.e (a) Google play store (b) Virus total [25] and (c) malgenome project [2] and (d) malware security blog[26] . We consider botnet dataset that comprises 1321 applications from 15 different families that having C&C ability. In table 2, a summary of a collected dataset is characterized in two types of train a dataset and test dataset. While on the other hand, in table 3 we presented botnet families with their C & C types, infection vector, number of samples and market source. We take botnet variants from the year 2011 to 2018 that included Geinimi, Zitmo, Sandroid, and state-of-the-art android botnet malware such as wireX [27] and Rottensys [28].

Table 2. Summary of Datasets

Datasets	Category	No. of Samples	Source
Train Dataset	Benign	575	VirusTotal [25], Malgenome Project [19] and malware security blog [29]
	Botnet	1321	
Test Dataset	Benign	575	
	Botnet	610	

N Table 3. Summary of Selected Malware Families

Botnet Families	Introduced year	C&C Type				Market source		Infection Vector	No. of sample
		HTTP	SMS	DNS	Email	Official android market	Unofficial android market		
Anserverbot	**2011**	✓	✓			✓		Install payloads and send SMS to users to activate itself	213
DroidDream	**2011**	✓				✓		It collect information such as IMEI no. device model and install other applications.	323

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

Geinimi	**2011**	✓					✓	It receive commands server to control phone and steal data.	208
PJapps	**2011**	✓					✓	it is capable to install applications, send messages through C&C.	58
Nickyspy	**2011**		✓				✓	It steal information such as GPS and Wi-Fi state.	90
TigerBot	**2012**		✓			✓		Operate through SMS to record calls and GPS location	32
Rootsmart	**2012**	✓					✓	It silently install other malwares and collect device information	26
Zitmo	**2012**		✓				✓	It steal banking credentials details of user and send it to through SMS	65
MisoSMS	**2013**				✓		✓	Uses up to 450 unique email address to steal SMS messages	77
Sandroid	**2014**		✓				✓	Steal bank account detail and demand ransom	22
Pletor	**2014**	✓	✓				✓	It has ability to hack device resources and demand ransom to release	83
NotCompatible.C	**2015**			✓			✓	It has an ability to send spam messages without user knowledge, it also launches spam campaigns and brute force attack.	74
Android/Twitoor	**2016**		✓				✓	It hide themselves track twitter accounts, disclose information, and install malicious applications.	3
WireX	**2017**					✓		It causes Denial of Service of attack and found in over 300 applications.	16
Rottensys	**2018**						✓	It has 316 variants, drain battery life, earned US $115,000 in just 10 days and maliciously install other applications.	31

3.2 Analysis Module

In this module, we examine applications in a hybrid manner which relates to static and dynamic analysis to classify botnet apks from benign. The analysis module divided into three types (a) static analysis (b) dynamic analysis and (c) hybrid analysis. For static analysis, different tools are used such as androguard [30], APK inspector [31] to extract static features of android applications. In order to disassemble the source code of apks (Applications Package kit) into the readable format, we used Andro-guard [30] which is available as an open source project. Similarly, Androguard tool can analyze any application whether it is good ware or malware by decompiling them. Furthermore, AndroidManifest.xml and classes.dex files are mined in which permissions, intents and native code is accessible. Correspondingly, dynamic analysis inspects the behavior of applications at runtime in a protected environment. The various dynamic tools are available i.e. DroidScope [32], Droidbox [33] APK Analyzer [34] to test applications. Droidbox [33] conduct results in the form of graphics or text by examining the behavior of applications in emulator.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

Correspondingly, we also used Andrubis [35] for hybrid analysis that provides productive environment for both static and dynamic analysis.

3.3 Features Module

The features module plays a vital role in learning-based system. Hence, we divided it into two parts (a) features extraction and (b) features selection. We applied a similar method to extract features from benign and botnet applications through reverse engineering. We applied python script on all extracted .xml and .jason files that contain several static and dynamic features. All extracted files is converted into comma-separated value (CSV) file that contains numerous features. In CSV file all values show in binary format in which "1" denotes that features are activated whereas "0" shows deactivated features.

As a result, from a mined set of static and dynamic features, we divided these features into further categories. In table 4, we discussed a static feature set that having (a) Permissions (b) metadata (c) intents (d) Reflection and (e) Cryptographic functions. Whereas, in table 5, we identically described dynamic features that comprise (a) Network traces (b) system calls(c) API calls (d) cryptographic operations and (e) Data operations. In this work, we considered a hybrid feature set that is frequently in the practice of malware writers.

Table 4. Extracted Static Features

Permissions	Metadata	Intents	Cryptographic	Reflection/Obfuscation	Others
INTERNET, ACCESS_FINE_LOCATION, ACCESS_NETWORK_STATE, SEND_SMS,RECIVE_SMS, ACCESS_WIFI_STATE, READ_PHONE_STATE, WRITE_EXTERNAL_STORAGE, READ_SMS, ACCESS_COARSE_LOCATION, INSTALL_SHORTCUT, READ_LOGS, req_per, used_per,	Owner, Issuer, serial_number, validity, application_name, fingerprints	intent_filter, broadcast_receivers	uses_crypto	uses_reflection, uses_dynamic_code, uses_native_code, java_packages	valid_manifest, valid_zipfile, valid_androguard_zipfile call

Table 5. Extracted Dynamic Features

Network	API Calls	Cryptographic Operations	Data Operations	Others
open_conn, Host, Path, Port, Socket, Connect, Read_network, Write_netwotk, Network_leaks, dns_queries, http_conversations, unknown_tcp_conversations, unknown_udp_conversations	getCellLocation, getContent, getWifiState, getConnectionInfo, getActiveNetworkInfo, getLastKnownLocation, getLine1Number, getInputStream, getSimSerialNumber, getSubscriberId, getDeviceSoftwareVersion, getDefault,	Crpto_op, Cp_traffic	File_read, File_write, File_leaks,	sendTextMessage, sent_sms, received_sms, started_services, exec, execute, TAINT_IMEI TAINT_IMSI

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

3.4 Machine Learning Module

Machine learning techniques are used to classify malware binaries from benign to conduct results with the lowest error rate. In this study, we consider four renowned Ml classifiers i.e. Random Forest, Naïve Bayes, Multi-layer perceptron, J48, and Simple logistics. During classification, results evaluated by 10 fold cross validation method that has various metrics: Area under Curve, Receiver Operating Characteristic (AUC and ROC), recall, and precision. The ROC curve comparative analyzes True Positive Rate and False Positive Rate while on the other hand, AUC conforms the highest detected value. Furthermore, TPR, FPR, TNR, FNR, precision, and recall defined in Table 6.

Table 6. Evaluation parameters of Machine Learning

Evaluation Parameters	Formula	Description
True Positive Ratio, Recall (TPR)	$\frac{TP}{TP+FN}$	It is a ratio that truly classified android botnet applications from the total number of botnet apps in the dataset.
False Positive Rate(FPR)	$\frac{FP}{TN+FP}$	It is a ratio that incorrectly classified android benign applications from the total number of benign apps in the dataset.
True Negative Rate (TNR)	$\frac{TN}{TN+FP}$	It is a ratio that truly classified android benign applications from the total number of benign apps in the dataset.
False Negative Rate(FNR)	$\frac{FN}{TP+FN}$	It is a ratio that incorrectly classified android botnet applications from the total number of botnet apps in the dataset.
Precision	$\frac{TP}{TP+FP}$	It is a ratio that truly classified botnet applications to the total truly classified botnet apps.
F-Measures	$\frac{2*precision*Recall}{Precision+Recall}$	It validates the test accuracy and also called F score that carries the balance among precision and recall.
Accuracy (ACC)	$\frac{TP+TN}{TP+TN+FP+FN}$	It shows the total number of truly classified android botnet and benign applications which is divided by total number of occurrences.

In binary classification, we show defined labeled data $\{Mi|Fi\}\ Ntr\ i=1,$ in this case M is a variable relates to malware family and $Mi\ \in\{0,1\}$ and Fi is an array that contain values of features or P predictors as ,$Fi=(fi1,\ldots,fiP)$. However, Fi conforms 82 hybrid features that contain permissions, API calls, Data operations, cryptographic operations and network features. Additionally, we also applied machine learning tool a WEKA [36] to rank features that help us in selection process. Most of famous attribute selection algorithms are used such as correlation based, information gain, learner based and principle component. In this study, we choose InfoGainAttributeEval [37] that calculates information gain score of extracted features. Furthermore, information gain score evaluates worth of each attribute as it is called entropy. It support us to select exact feature for experiment as it is described in below table 7 and 8 with diverse features as permissions, API calls and Network traces separately.

Table 7. Features (permissions, API calls) ranking Using InfoGainAttributeEval

Permissions	Frequency of Benign	Frequency of Botnet	Total	Info gain Score	API Calls	Frequency of Benign	Frequency of Botnet	Total	Info gain Score
READ_SMS	3	364	367	0.36921	**getSubcriberId()**	11	432	443	0.4389
SEND_SMS	8	377	385	0.36444	**getDeviceId()**	59	435	494	0.3033
READ_PHONE_STATE	131	530	661	0.3261	**getNetworkInfo**	32	357	389	0.2588
READ_CONTACTS	48	410	458	0.29162	**getActive NetworkInfo**	182	412	594	0.1088
READ_LOGS	24	346	370	0.26629	**getCellLocation**	2	98	100	0.0739
RECEIVE_SMS	3	257	260	0.23072	**getDefault**	4	105	109	0.0729
ACCESS_NETWORK_STATE	310	538	848	0.1087	**getInputStream**	71	198	269	0.043

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

ACCESS_WIFI_STATE	67	270	337	0.09952	**getWifiState**	2	24	26	0.0128
INTERNET	493	610	1103	0.076144					
ACCESS_FINE_LOCATION	178	122	300	0.011488					
INSTALL_SHORTCUT	10	35	45	0.008338					

Table 7. Ranking of Network Traces Feature Using InfoGainAttributeEval

Network Traces	Frequency of Benign	Frequency of Botnet	Total	Info gain Score
Network_leaks	164	862	1026	0.1031
openConnection	246	437	683	0.0634
Connect	241	418	659	0.053
Open_conn	4628	8865	13493	0.0501
http_conversations	3455	3261	6716	0.0228
Con_attempts	383	5397	5780	0.018

4. Results and Discussions

This section describes experimental results from the analysis on separate datasets. In the first instance, we compare exciting features and their trends among botnet and benign applications by considering static and dynamic features individually. As we take two datasets (a) train dataset and (b) test dataset. In which a test dataset help us to validate our proposed hybrid analysis framework. Furthermore, test results are validated by a machine learning tool WEKA [36]. In the end, results are determined and identified the best classifier for detection model.

4.1 Observed Static Features among Botnet and Benign

In this subsection, we discussed frequently used static features by botnet authors. We seen usage rate of below-mentioned features is surprisingly high in botnet applications as compare to benign.

4.1.1 Permissions

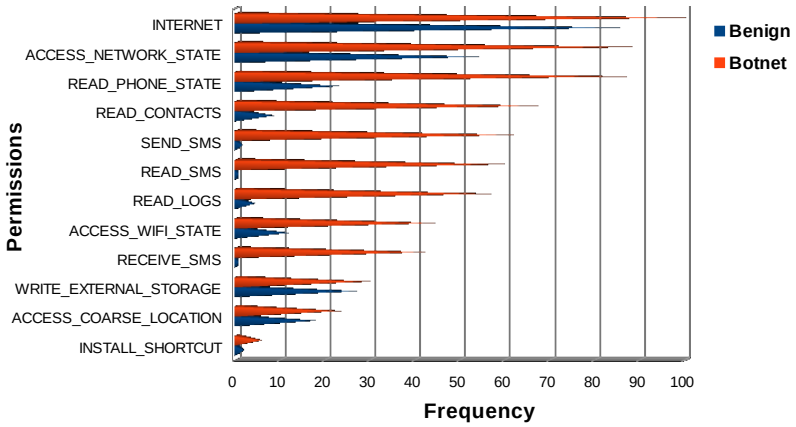

Figure 4. Usage of Permissions Analysis in a botnet and benign applications

In figure 4, we observed all those prominent permissions that are frequently in the practice of botnet as compare to benign applications. The permissions used by botnet applications are INTERNET, ACCESS_NETWORK_STATE and READ_PHONE_STATE with 100%, 88% and 86% respectively. In contrast, these permissions in benign applications are utilized at a fewer rate with a percentage of 85, 53 and 22 exclusively. The permissions indicated above are usually required in network connections to launch

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

attacks through C&C. Besides, it tries to detect current phone state that will help bot master to start conciliation with cell phones. Besides, we observed READ_LOGS, READ_SMS, SEND_SMS, RECEIVE_SMS, and READ_CONTACTS permissions that are regularly used by malware authors. Botnets having C&C mechanism is using SEND_SMS permission to send messages at premium rate numbers routinely with a percentage of 61%. In comparison, only 1.3% of SEND_SMS used in benign applications. Similarly, another permission READ_CONTACTS is indicated 67 % usage in botnet, whereas only 8% in benign applications. Furthermore, INTSALL_SHORTCUTS and ACCESS_COARSE_LOCATION are a type of permissions used at a very low rate in benign as well as botnet applications with 5%, 23% and 1%, 17% individually.

4.1.2 API Calls graph

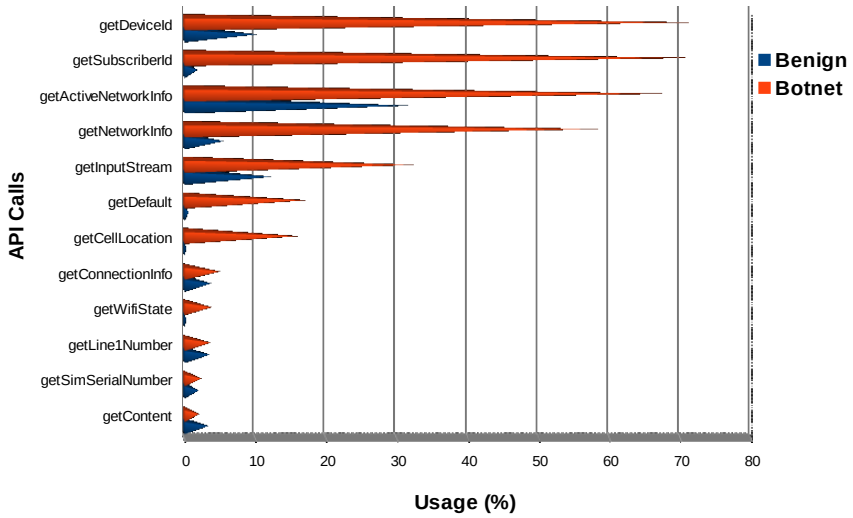

Figure 5. Usage of API Calls Analysis in botnet and benign applications

In figure 5, we analyzed API calls because botnet not only relies on permissions to generate an attack. Most commonly used API calls are *getDeviceId (), getSubcriberId () and getActiveNetworkInfo*, with 71%, 70%, and 67% separately. Conversely, benign applications least utilize these API calls with 10%, 1%, and 31 % individually. This information required to get identification and current network state of devices and send it to a remote server. Moreover, *getCellLocation ()* and *getDefault ()* is used to track users current location to exploit them. The above-mentioned API calls are in practice of botnet applications with 16% and 17% while benign applications only employ it with 0.3% and 0.6% respectively. The API calls as *getLine1Number ()* and *getSimSerialnumber ()* are equally used by a botnet and benign application with 3%. However, *getConnect ()* API call used in benign and botnet application with 3% and 2% distinctly.

4.2 Observed Dynamic Features among Botnet and Benign

In dynamic feature selection, results are noticeably shows rising trend of botnet applications with the nominated features array such as Network Traces, and file operations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

4.2.1 Network Traces

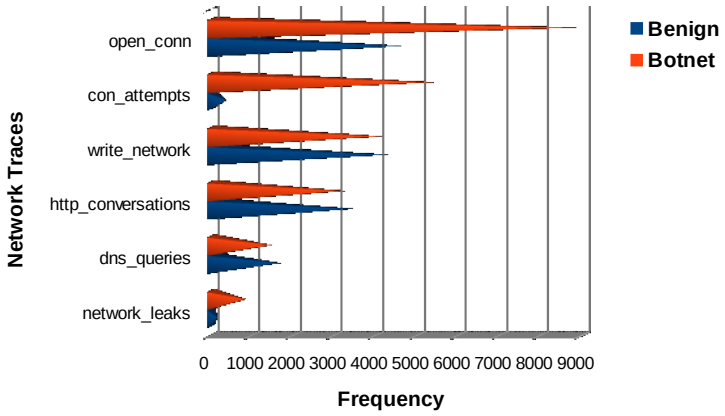

Figure 6. Network Traces Frequency Analysis

In figure 6, we detected network traces features that are most commonly used in botnet applications. *open_conn* and *con_attempt* feature try to establish a connection via TCP and UDP conversations. The observed frequency of described network traces is 8865 and 5397 in botnet applications. In opposite, benign applications have 4628 and 383, respectively. Similarly, botnet having C&C mechanism causes *network_leaks* with the frequency of 862 conversely in benign applications its occurrences is 164 times. Also, we noticed *http_conversations* that create a connection between browser and web servers. It contains information about data, source and destination address, and time stamp. The *http_consversations* used in benign and botnet applications with an average of 5.6 and 5.3 individually.

4.2.2 File operations Graph

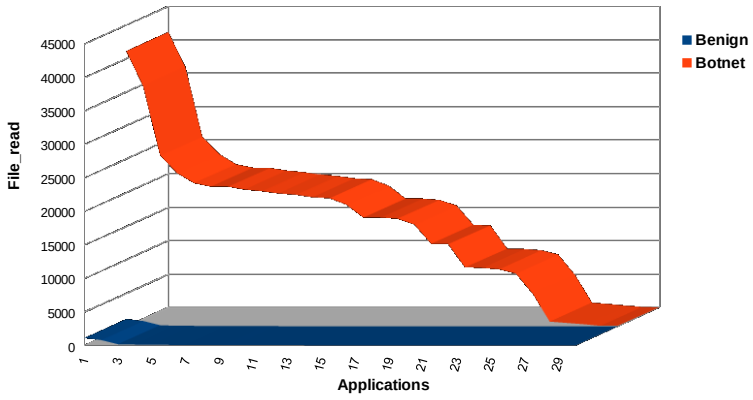

Figure 7. File read Analysis in botnet and benign applications

Below graphs 7, 8, 9 are used to depict the results of file operations in which *file_write*, *file_read* and *file_leaks* are extensively in practice of malware authors. In figure 7, we observed the top most applications in the context of *file_read* operations. Botnet application read files and causes to steal sensitive data [38]. We can see that; file read the feature in top 29 botnet applications in the opposite of benign applications, which is much higher than benign applications. Furthermore, we observed that *file_read* in botnet applications is surprisingly high in opposite of benign with an average of 766 and 11 times identically.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

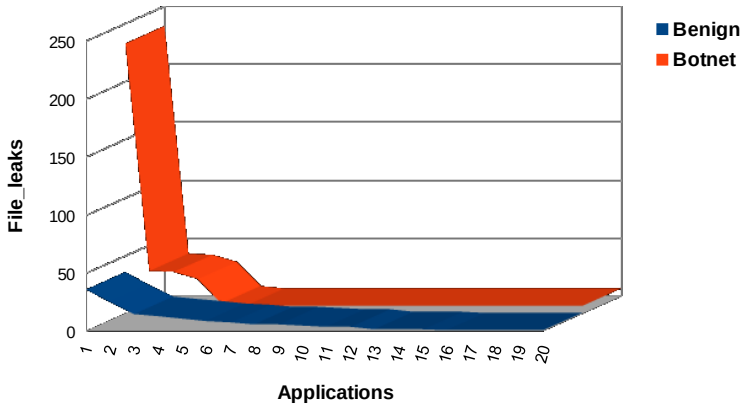

Figure 8. File leaks Analysis in a botnet and benign applications

Figure 8 depicted that *file_leaks* operation in top 20 botnet and benign applications. Botnet applications cause *file_leaks* in which sensitive data such as device information, account login details, steal by botnet authors. The average rate of *file_leaks* in a botnet and benign applications is 244 and 44 individually.

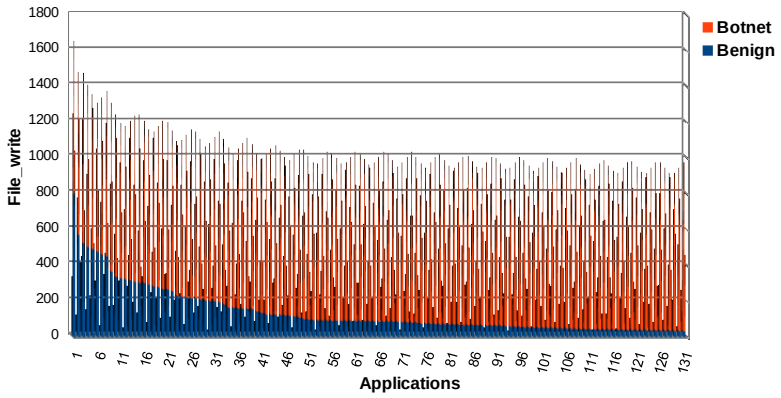

Figure 9. File write Analysis in a botnet and benign applications

File_write, a dynamic feature that is used by android botnet authors in apks. In file write operation, botnet writers additionally add malicious content in files and automatically write multiple files in data storage that causes excessive use of memory. In figure 17, we observed the top 131 applications that use the aforementioned feature extensively to perform harmful operations. The usage rate of *file_write* operations in a botnet and benign application is as above 1600 and 1200 individually. The average rate of *file_write* in benign and botnet is 281 and 59, respectively. Moreover, its use is seen in some applications as above 1000 and 200 in a botnet and benign applications separately. In file operations, botnet applications used *file_write* to write malicious binaries in external storage.

4.2.3 Crypto Operations Statistics

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

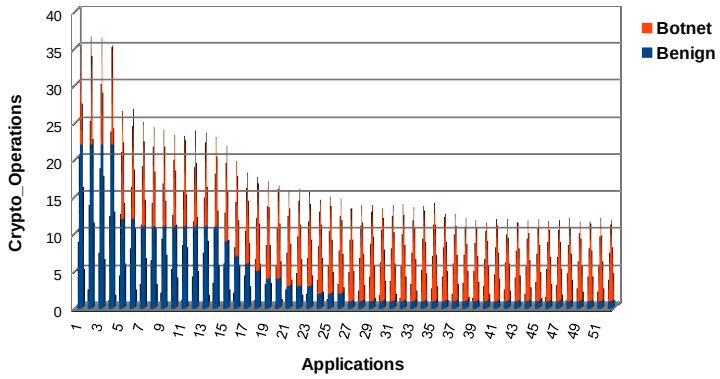

Figure 10. Crypto operations Analysis in a botnet and benign applications

In Cryptographic operations, an encrypted code is used to secure the communication between users other details such as bank account logins. While on the other hand, android botnet authors used cryptographic operations to hide malicious code and make this code undetectable from applications. Likewise, malware writers used malicious encrypted code that helps them to evade from the detection mechanism. In figure 10, we observed crypto_operations in a botnet and benign applications. It shows that crypto operations are in more practice in botnet applications as a comparison with benign applications. Botnet applications initiate 22 crypto operations, while benign applications only 15 times. The average rate of cryptographic operations in a botnet and benign applications is 2011 and 276, respectively.

4.3 Effectiveness & Performance

The performance of machine learning classifiers measured in terms of TPR, FPR, Recall, Precision, F-Measures, and Accuracy, whereas for effective testing, we select ten fold cross-validation method. The better empirical results are presented in table 8, 9, and 10.

Table 8. Static analysis results with ten fold cross Validation

	Evaluation Parameters of ML					
Classifiers	**TPR**	**FPR**	**Recall**	**Precision**	**F-Measures**	**Accuracy**
Naïve Bayes	0.898	0.083	0.898	0.910	0.900	89.78%
Simple Logistic	0.955	0.076	0.955	0.955	0.968	95.48%
Multi-layer Perceptron	0.957	0.068	0.957	0.957	0.957	95.37%
J48	0.954	0.070	0.913	0.954	0.954	95.37%
Random Forest	0.957	0.058	0.957	0.965	0.965	96.56%

In table 8, we described the static analysis results of different machine learning classifiers with their evaluation parameters. Among the aforementioned classifiers, naïve Bayes gives the lowest accuracy on 89.78% while random forest proves as best classifier with an accuracy of 89.78%.

Table 9. Dynamic analysis results with ten fold cross Validation

	Evaluation Parameters of ML					
Classifiers	**TPR**	**FPR**	**Recall**	**Precision**	**F-Measures**	**Accuracy**
Naïve Bayes	0.725	0.157	0.725	0.826	0.737	72.53%
Simple Logistic	0.903	0.142	0.903	0.902	0.903	90.29%
Multi-layer Perceptron	0.904	0.148	0.904	0.903	0.903	90.40%
J48	0.939	0.089	0.939	0.939	0.939	93.89%
Random Forest	0.951	0.075	0.951	0.951	0.951	95.12%

Table described above demonstrates effective detection rate of android botnet applications with different machine learning classifiers. Random forest gives maximum accuracy with 95.12% in comparison with other classifiers.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

Table 10. Hybrid Analysis results with 10 fold cross Validation

	Evaluation Parameters of ML					
Classifiers	**TPR**	**FPR**	**Recall**	**Precision**	**F-Measures**	**Accuracy**
Naïve Bayes	0.854	0.102	0.854	0.883	0.859	85.42%
Simple Logistic	0.963	0.055	0.963	0.963	0.963	96.48%
Multi-layer Perceptron	0.965	0.051	0.965	0.965	0.965	96.45%
J48	0.967	0.046	0.967	0.967	0.967	96.66%
Random Forest	0.975	0.034	0.975	0.975	0.975	97.48%

The result of the hybrid analysis is shown in table 8 with 10-fold cross-validation method. Random Forest classifier outperforms, achieved the highest detection rate with 97.48% accuracy and lowest false positive rate with 0.034 in an experiment. Moreover, other algorithms such as Simple logistic, J48, and Multilayer Perceptron has an accuracy of 96%. Overall the cumulative false positive rate and accuracy of each classifier is presented in below figures 11 and 12 identically.

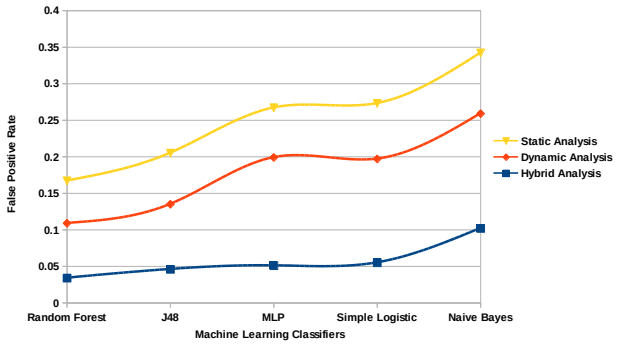

Figure 11. False Positive rate of machine learning Classifiers

The above graph explains the well false positive rate of analysis types with ML classifiers. Consequently, it shows the least false positive rate of 0.034 in the hybrid analysis as compared to static and dynamic analysis.

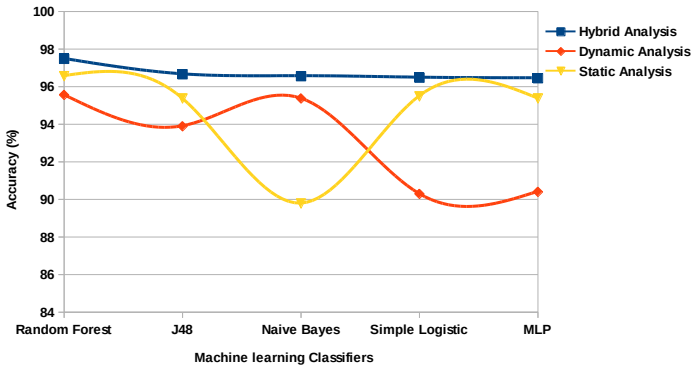

Figure 12. Accuracy of Machine learning Classifiers

In figure 12, we identify the accuracy of different analysis types as static, dynamic, and hybrid. We performed a hybrid method, and it gives excellent outcome with 97.48% accuracy in random forest classifier. Whereas static and dynamic result shows 96.56% and 95.12% accuracy respectively, with random forest classifier. From results, we learn that the hybrid method is more useful other than applying any single method such as static or dynamic.

4.4 Comparison with existing Approaches

By, our proposed malware detection framework, we analyze different existing approaches. The below-mentioned table 10 comparatively elaborate heterogeneous botnet detection techniques with their prominent features and accuracy.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 June 2019 doi:10.20944/preprints201906.0060.v1

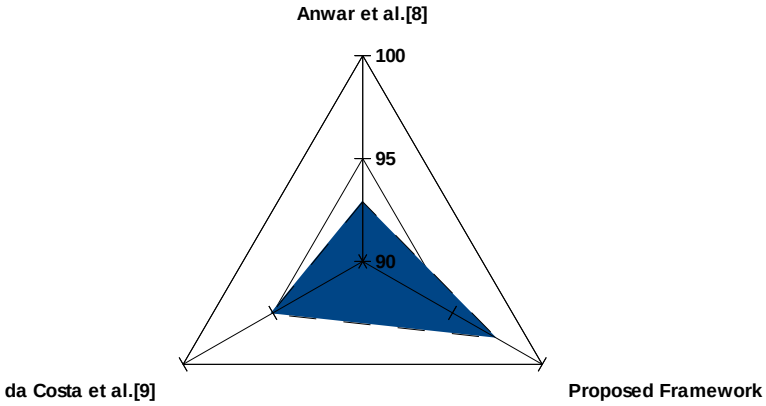

Figure 12. Comparison with existing approaches

In literature, most of the researchers adopted only static or dynamic analysis to perform their experiments. In figure 12, we comparatively show the accuracy of our proposed hybrid framework. For instance, Da costa et al. [9] detect botnet applications by implementing dynamic analysis method and wherefore it gives only 92.9% accuracy. Additionally, another researcher [8] follow a static analysis method by having three noticeable static features like permissions, broadcast receivers, and background services and provide 95.1% accuracy. However, we applied a hybrid method to detect android botnet applications having C&C ability. In this way, our proposed framework shows 97.48% detection accuracy with prominent features that we well-thought-out in our experiment.

5. Future directions

In this section, we discuss a few possible keys and future work directions that help us to evade from botnet application attacks. By following these measures, users can escape from this growing mobile botnet hazard.

- As google play store provide advance security mechanism in terms of “play protect” it checks application on a regular basis to find out malicious behavior. It helps to protect about two billion number of user’s every day around the globe [39]. Make sure that the “scan and improve devices for harmful contents” option from play should be activated for better security.
- Permissions in Android applications are basically firewalled to access the resources of smartphones and how much they have the freedom to use to control device. During the installation of any application, it should be thoroughly analyzed its permissions and then accept it. Most of the dangerous permissions such as read contacts, send SMS automatically, access camera, microphone, personal files, and so forth are used to exploit users. So, pay close attention on permissions before installing any application.
- Install appropriate mobile security applications to keep secure your device and data. Besides this, take a backup of essential data and always connect secured internet connection. In this context internet service providers (ISP) should have to deliver security mechanism.
- It is a crucial fact to educate peoples about cybersecurity as it directly impacts on their lives. It is need to hold seminars, workshops on a regular basis that relates to awareness program of cybersecurity. Peoples need to learn about installation to the development process of applications for their smartphone to aware of security measures.
- The government should make policies, legislation to deal with such harmful attacks and enforce regulations at an international level. As FBI [40] recently announced cybersecurity parameters for toys that connected with the internet and if you suspected that your toy is compromised then you can report them. Such methods devised in mobile security measures.
- Moreover, it is recommended to download applications from the trusted app store as google play store [24], do not allow installation from unknown resources, only install verified applications and check ratings from existing users that indicates about the performance of an application. Similarly, reliable security solutions should be preferred to clean device and applications.

 doi:10.20944/preprints201906.0060.v1

6. Conclusion

In this paper, we proposed a hybrid learning based mobile botnet detection framework which is divided into four modules (a) data collection (b) analysis module (c) features module and (d) machine learning module. In the hybrid analysis, we take prominent static and dynamic features as permissions, API calls, Network Traces, Data operations, etc. Furthermore, we presented frequency analysis graphs that indicate how botnet attack is in practice of malware writers in comparison with benign applications. Consequently, we used different machine learning classifiers like Random Forest proves that best accuracy with 97.48% and least false positive rate with 0.034 in the hybrid analysis.

References

1. statcounterGlobalstats. *Operating System Market Share Worldwide*. 2018 5/27/2018]; Available from: http://gs.statcounter.com/os-market-share#monthly-201701-201805-bar.
2. Statista. *Number of apps available in leading app stores as of 1st quarter 2018*. 2018 5/27/2018]; Available from: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
3. Nigam, R., *A timeline of mobile botnets.* Virus Bulletin, March, 2015.
4. Cimpanu, C. *Chinese Crooks Assembling Massive Botnet of Nearly 5 Million Android Devices*. March 15, 2018 [cited 5/27/2018; Available from: https://www.bleepingcomputer.com/news/security/chinese-crooks-assembling-massive-botnet-of-nearly-5-million-android-devices/.
5. McAfee, *Mobile Threat Report Q1, 2018*, 2018.
6. Kirubavathi, G. and R. Anitha, *Structural analysis and detection of android botnets using machine learning techniques.* International Journal of Information Security, 2018. **17**(2): p. 153-167.
7. Karim, A., R. Salleh, and M.K. Khan, *SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications.* PloS one, 2016. **11**(3): p. e0150077.
8. Anwar, S., et al. *A static approach towards mobile botnet detection*. in *Electronic Design (ICED), 2016 3rd International Conference on*. 2016. IEEE.
9. da Costa, V.G., et al. *Detecting mobile botnets through machine learning and system calls analysis*. in *Communications (ICC), 2017 IEEE International Conference on*. 2017. IEEE.
10. Yang, M. and Q. Wen. *A multi-level feature extraction technique to detect moble botnet*. in *Computer and Communications (ICCC), 2016 2nd IEEE International Conference on*. 2016. IEEE.
11. pearltrees. *Anubis-Malware Analysis for Unknown Binaries*. 5/28/2018]; Available from: http://www.pearltrees.com/u/4051585-malware-analysis-binaries.
12. CopperDroid. 2018 5/28/2018]; Available from: http://copperdroid.isg.rhul.ac.uk/copperdroid/.
13. Store, B.A. *ONE-STOP STORE*

Downloading & Managing PC Apps. 2018; Available from: http://pcappstore.baidu.com/en/index.php.

14. *The Drebin Dataset*. 2018; Available from: https://www.sec.cs.tu-bs.de/~danarp/drebin/.
15. Girei, D.A., M.A. Shah, and M.B. Shahid. *An enhanced botnet detection technique for mobile devices using log analysis*. in *Automation and Computing (ICAC), 2016 22nd International Conference on*. 2016. IEEE.
16. Eslahi, M., et al. *Cooperative network behaviour analysis model for mobile Botnet detection*. in *Computer Applications & Industrial Electronics (ISCAIE), 2016 IEEE Symposium on*. 2016. IEEE.
17. Bojjagani, S. and V. Sastry, *Stamba: Security testing for Android mobile banking apps*, in *Advances in Signal Processing and Intelligent Recognition Systems*. 2016, Springer. p. 671-683.
18. Strazzere, T. and T. Wyatt, *Geinimi trojan technical teardown.* Lookout Mobile Security, 2011.
19. MalGenomeProject. *Android Malware Genome Project*. 5/29/2018]; Available from: http://www.malgenomeproject.org/.

20. Al-Dayil, R.A. and M.H. Dahshan. *Detecting social media mobile botnets using user activity correlation and artificial immune system*. in *Information and Communication Systems (ICICS), 2016 7th International Conference on*. 2016. IEEE.
21. Cybersecurity, C.I.f. *Datasets*. 2018 5/29/2018]; Available from: http://www.unb.ca/cic/datasets/index.html.
22. Yusof, M., M.M. Saudi, and F. Ridzuan. *A new mobile botnet classification based on permission and API calls*. in *Emerging Security Technologies (EST), 2017 Seventh International Conference on*. 2017. IEEE.
23. Faris, H. *Toward a Detection Framework for Android Botnet*. in *New Trends in Computing Sciences (ICTCS), 2017 International Conference on*. 2017. IEEE.
24. Play, G. *Google Play*. 2018; Available from: https://play.google.com/store?hl=en.
25. VirusTotal. 2018; Available from: https://www.virustotal.com/#/home/upload.
26. *Malware Security blog* Available from: http://artemonsecurity.blogspot.com/.
27. NJCCIC. *WireX*. 2017; Available from: https://www.cyber.nj.gov/threat-profiles/botnet-variants/wirex.
28. NJCCIC. *RottenSys*. 2018; Available from: https://www.cyber.nj.gov/threat-profiles/android-malware-variants/rottensys.
29. Mobile, C. *Contagio Mobile*. Available from: http://contagiominidump.blogspot.com/.
30. GitHub. *androguard*. 2018; Available from: https://github.com/androguard/androguard.
31. GitHub. *apkinspector*. 2011; Available from: https://github.com/honeynet/apkinspector/.
32. Yan, L.-K. and H. Yin. *DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis*. in *USENIX security symposium*. 2012.
33. GitHub. *DroidBox*. 2018; Available from: https://github.com/pjlantz/droidbox.
34. Android. *APK Analyzer*. 2018; Available from: https://developer.android.com/studio/build/apk-analyzer.
35. K.c. *Andrubis: Scan & Analyze Android Apks*. Available from: http://hackpla.net/anubis-scan-android-apks/.
36. WEKA. *Weka 3: Data Mining Software in Java*. Available from: https://www.cs.waikato.ac.nz/ml/weka/.
37. Yerima, S.Y., S. Sezer, and G. McWilliams, *Analysis of Bayesian classification-based approaches for Android malware detection.* IET Information Security, 2014. **8**(1): p. 25-36.
38. Anwar, S., et al., *Android Botnets: A Serious Threat to Android Devices.* Pertanika Journal of Science & Technology, 2018. **26**(1).
39. Google. *Google Play Protect: Securing 2 billion users daily*. 2018; Available from: https://www.android.com/play-protect/.
40. FBI. *CONSUMER NOTICE: INTERNET-CONNECTED TOYS COULD PRESENT PRIVACY AND CONTACT CONCERNS FOR CHILDREN*. 2017; Available from: https://www.ic3.gov/media/2017/170717.aspx.