Computer Systems Science & Engineering
DOI: 10.32604/csse.2022.023652
Article

A Learning Model to Detect Android C&C Applications Using Hybrid Analysis

Attia Qammar[1], Ahmad Karim[1,*], Yasser Alharbi[2], Mohammad Alsaffar[2] and Abdullah Alharbi[2]

[1]Department of Information Technology, Bahauddin Zakariya University, Multan, 60000, Pakistan
[2]College of Computer Science and Engineering, University of Hail, Ha'il, 81451, Saudi Arabia
*Corresponding Author: Ahmad Karim. Email: ahmadkarim@bzu.edu.pk
Received: 15 September 2021; Accepted: 25 November 2021

Abstract: Smartphone devices particularly Android devices are in use by billions of people everywhere in the world. Similarly, this increasing rate attracts mobile botnet attacks which is a network of interconnected nodes operated through the command and control (C&C) method to expand malicious activities. At present, mobile botnet attacks launched the Distributed denial of services (DDoS) that causes to steal of sensitive data, remote access, and spam generation, etc. Consequently, various approaches are defined in the literature to detect mobile botnet attacks using static or dynamic analysis. In this paper, a novel hybrid model, the combination of static and dynamic methods that relies on machine learning to detect android botnet applications is proposed. Furthermore, results are evaluated using machine learning classifiers. The Random Forest (RF) classifier outperform as compared to other ML techniques i.e., Naïve Bayes (NB), Support Vector Machine (SVM), and Simple Logistic (SL). Our proposed framework achieved 97.48% accuracy in the detection of botnet applications. Finally, some future research directions are highlighted regarding botnet attacks detection 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 lives by holding every kind of information. Smartphone devices, especially Android mobiles are very popular due to their affordability, rich user environment, and appealing applications such as maps, weather forecasting, GPS function, and many more. According to StatCounter Globalstats report [1], the Android operating system has a 42.26% worldwide market share which is larger than compared to others as shown in Fig. 1. Furthermore, Statista [2], reported that Android is the first leading store with 3.8 million applications in the first quarter of 2018.

As a result, the emerging rate of an open-source Android OS was not ignored by malware writers and mobile botnet attacks grows faster. A mobile botnet consists of a network of bots operated through the command and control (C&C) method by the botmaster. A botnet attack launched the Distributed Denial of service (DDoS) attack, steals personal information, gain illegal access to services, send emails, send

SMS on premium-rate numbers, and infects multiple mobile devices. The first botnet *SmsHOw.U* was discovered in September 2010 and till now several mobile botnets are available i.e., Geinimi, DroidKungFu, GoldDream, NotCompatible, Fakeplay, and Anseverbot, etc [3]. A RottenSystem botnet [4], was activated in 2016 and till now affected ca. 5 million Android devices in the first quarter of 2018. Fig. 2 shows the McAfee mobile threat report [5], in the year 2018.

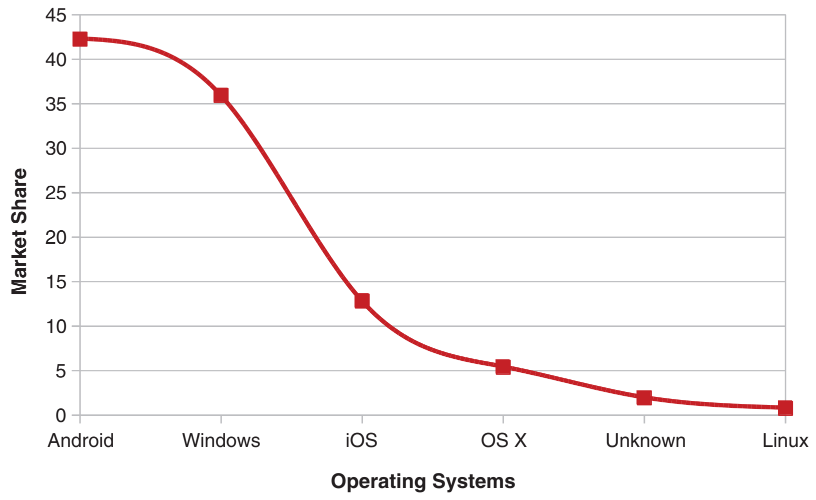

Figure 1: Operating system market share worldwide from January 2018 to July 2018 [1]

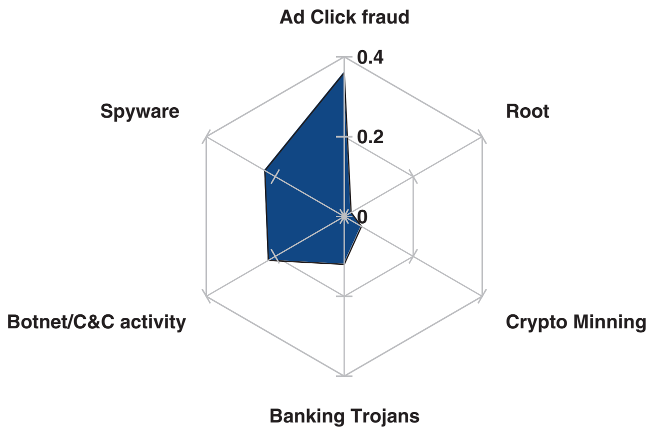

Figure 2: Malware attacks growth rate in the first quarter of 2018 [2]

In literature, many methods are used to detect Android malware attacks. These methods are divided into two types (a) static and (b) dynamic analysis. Static analysis refers to the code analysis while in dynamic method ensures to analyze the behavior of an application at runtime. Kirubavathi et al. [6], proposed an Android botnet detection model based on structural analysis with the features of permissions and API calls. The authors, take a large dataset to perform detection operations and achieved the highest accuracy with 99% via Support Vector Machine (SVM) classifier. However, dynamic features of android applications were not discussed. Another researcher [7], introduced the dynamic detection model such as SMARTbot for android botnet detection. The rich dynamic features such as network traces, DNS queries, and file activities, etc. are considered. There are some limitations in both types of analysis methods. Static analysis are incompetent to detect obfuscated code that's why researches used the dynamic analysis.

Correspondingly, dynamic analysis are not able to deal with native code and do not provide full code coverage. Hence, in this paper hybrid analysis method is used, that combines static and dynamic analysis to deal with these challenges effectively.

In short, the main contributions in this paper are as follows:

1. A hybrid analysis model with machine learning techniques to detect android botnet applications having C&C capabilities is proposed.
2. The most prominent static and dynamic features in the perspective of malware writers that used to target android applications are investigated. Specifically, C&C methods are used in malicious code to generate an attack. Furthermore, static and dynamic features such as permissions, API calls, file operations, and Network traces are extracted, respectively. Besides, frequency analysis graphs are presented to distinguish botnet and benign applications.
3. To validate results with existing approaches, machine learning (ML) techniques are implemented to classify C&C-specific applications from benign. Among heterogeneous ML classifiers, Random Forest (RF) proved as the best classifier with 97.48% accuracy.
4. The proposed hybrid model compares well with other existing approaches that include static and dynamic techniques of mobile botnet detection [8,9].
5. Finally, some guidelines are provided to avoid threatening hazards.

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

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 methods but they did not consider the hybrid method. In this section, a few existing approaches are discussed.

Yang et al. [10], introduced the multilevel features extraction dynamic method that contains (a) basic level features (b) traffic level features, and (c) content level features to detect mobile botnet from applications package kit (APK) files. Together with these features, network traffic that relates to hypertext transfer protocol (HTTP) is collected from Anubis [11], CopperDroid [12], and Sandroid platforms. For the experiment, datasets are collected from Baidu app market [13] and Android Drebin project [14] for normal and malicious applications, respectively. Furthermore, each application is executed for ten to fifteen minutes to assemble the network traffic. Additionally, a machine learning classifier such as RF is used to evaluate results and it provides a 93% true positive rate (TPR) in the detection of botnet applications.

Similarly, Girei et al. [15], adopted a dynamic analysis method and proposed a mobile botnet detection approach named "Logdog" by analyzing the log files. Logdog requires root permissions to start, collect logs, stop, and then maintain text files of log messages. In addition, for analysis purposes expressions are already saved in logs that notify the botnet activity if they matched with an existing malicious log. The authors performed an experiment on actual Android devices. Furthermore, the Android botnet application "dendroid" is used to communicate with the C&C server via HTTP to send logs.

In the work of [16], the authors performed network behavior analysis to detect HTTP-based mobile botnet attacks. A light application "Tpacketcapture" [17] was installed on mobile devices to capture the network traffic and collect data stored in a 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] are collected from the Genome Malware project [19]. Finally, results are validated via machine learning classifiers such as rule-based and J48. Hence, the combination of rule-based classifiers and proposed periodic metrics shows 98.60% accuracy in mobile botnet detection. However, rich botnet application datasets having C&C ability with state-of-the-art mobile botnet applications samples are considered in our paper.

Another study by Al-Dayil et al. [20], presented an Artificial immune system (AIS) with the combination of user activity to detect Android botnet from social networking sites like Twitter. Furthermore, provide discrimination between user-generated and bot tweets. The method comprises 4 steps (a) capture tweets from Twitter (b) relate Twitter activity with user's activity (c) create a signature for Twitter activity and (d) match with the existing signatures. In case, if it does not match then assume it is a legitimate tweet otherwise considered as bots-generated tweets. For the experiment, datasets are collected from Twitter stream API. Consequently, the AIS detector gives 95% detection accuracy.

Moreover, Hijawi et al. [21], proposed a static analysis framework by considering permissions feature to detect android botnet. The seven levels of permissions protection scheme are elaborated such as normal, signature-based, dangerous, and others permissions, etc. Moreover, they used botnet and benign datasets from ISCX android bot [22], and Google play store [23], conversely for the experiment. Eventually, a Machine learning classifier (i.e., Random Forest, Naïve Bayes, J48, and Multilayer Perceptron) are used to verify the proposed method. Between these classifier models, RF shows the best accuracy with 97.3% by achieving the highest botnet detection rate.

3 Methodology

In this section, a learning-based hybrid detection model is presented that discriminates between benign and botnet applications of C&C capability. In Fig. 3, a workflow of the proposed model is illustrated. There are four major modules in this model such as (1) data collection (2) analysis (3) features extraction and selection and (4) Machine learning module. These modules are briefly discussed in the following subsections.

3.1 Data Collection Module

The data collection module includes benign and botnet datasets of Android applications which are collected from different resources i.e., Google play store [23], Virus total [24], Malgenome project [2], and malware security blog [25]. The botnet dataset comprises 1321 applications from fifteen different families with C&C ability. In Tab. 1, a summary of the collected datasets is characterized with train and test datasets. Moreover, in Tab. 2 botnet families with their C&C types, infection vector, samples ratio, and market source are presented. The botnet variants from the year 2011 to 2018 are described including Geinimi, Zitmo, and state-of-the-art Android botnet malware such as wireX [26], and Rottensys [27].

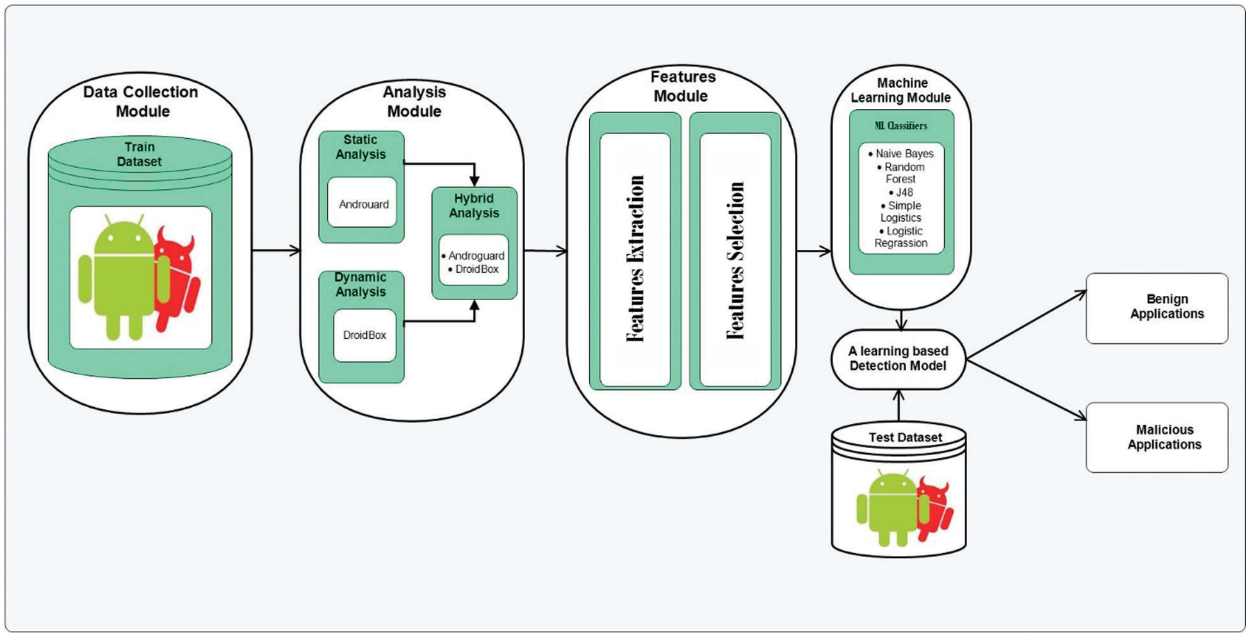

Figure 3: Proposed hybrid learning-based detection model

Table 1: Summary of datasets

Datasets	Category	No. of samples	Source
Train dataset	Benign	575	VirusTotal [25], Malgenome Project [19], and malware security blog [28]
	Botnet	1321	
Test dataset	Benign	575	
	Botnet	610	

3.2 Analysis Module

In this module, applications analysis in a hybrid manner (static and dynamic) analysis is performed to classify botnet from benign applications. The analysis module is divided into three types (a) static analysis (b) dynamic analysis and (c) hybrid analysis. For static analysis, different tools are used such as Androguard [29], APK inspector [30], to extract static features of Android applications. To disassemble the source code of apks into the readable format, Androguard [29], is used. Similarly, the Androguard tool can analyze any type of application whether it is benign or malware by decompiling them. Furthermore, AndroidManifest.xml and classes.dex files are extracted to obtain permissions, intents, and native code. Correspondingly, dynamic analysis inspects the behavior of applications at runtime in a protected environment. Dynamic tools such as i.e., DroidScope [31], Droidbox [32], APK Analyzer [33], are available to test applications. Correspondingly, Andrubis [34], for hybrid analysis is used in this paper that provides a rich environment for both static and dynamic analysis.

Table 2: 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 collects information such as IMEI no. device model and installs other applications.	323
Geinimi	2011	✓					✓	It receives commands server to control the phone and steal data.	208
PJapps	2011	✓					✓	It is capable to install applications, sending messages through C&C.	58
Nickyspy	2011		✓				✓	It steals information such as GPS and Wi-Fi state.	90
TigerBot	2012		✓			✓		Operate through SMS to record calls and GPS location	32
Rootsmart	2012	✓					✓	Silently install other malware and collect the device information	26
Zitmo	2012		✓				✓	Steal banking credentials details of the users and send it to through SMS	65
MisoSMS	2013				✓		✓	Uses up to 450 unique email addresses to steal SMS messages	77
Sandroid	2014		✓				✓	Steal bank account detail and demand ransom	22
Pletor	2014	✓	✓				✓	It can hack device resources and demand ransom to release them	83
NotCompatible. C	2015			✓			✓	Send spam messages without user knowledge, can launch spam campaigns and brute force attacks.	74
Android/ Twitoor	2016		✓				✓	It hides, tracks Twitter accounts, discloses information, and installs malicious applications.	3

(Continued)

Table 2 (continued)

Botnet families	Introduced year	C&C type			Market source			Infection vector	No. of sample
		HTTP	SMS	DNS	Email	Official android market	Unofficial android market		
WireX	2017					✓		It causes a Denial of Service attack and is found in more than 300 applications.	16
Rottensys	2018						✓	Causes to drain battery life, earned the US $115,000 in just 10 days, and maliciously install other applications.	31

3.3 Features Module

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

As a result, from the mined set of static and dynamic features, we have divided these features into further categories. In Tab. 3 static feature set such as (a) permissions (b) metadata (c) intents (d) reflection and (e) cryptographic functions are discussed. Whereas, in Tab. 4, dynamic features that include (a) network traces (b) system calls (c) API calls (d) cryptographic operations, and (e) data operations are presented. In this work, a hybrid features set is considered.

Table 3: 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,	Owner, Issuer, serial_number, validity, application_name, fingerprints	intent_filter, broadcast_receivers	uses_crypo	uses_reflection, uses_dynamic_coe, uses_native_code, java_packages	valid_manifest, valid_zipfile, valid_androguard_zipfile call

Table 4: 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

3.4 Machine Learning Module

Machine learning techniques are used to classify malware binaries from benign and to conduct results with the lowest error rate. In this study, five ML classifiers i.e., Random Forest, Naïve Bayes, Multi-layer perceptron, J48, and Simple Logistics. During classification, results are evaluated by 10 fold cross-validation method that has numerous metrics including area under the curve (AUC), receiver operating characteristic (ROC), recall, and precision. The ROC curve comparatively analyzes the True Positive Rate (TPR) and False Positive Rate (FPR) while on the other hand, AUC conforms to the highest detected value.

In binary classification, labeled data $\{Mi|Fi\}\ Ntr\ i = 1$, is defined. In this case, M is a variable relates to malware family and $Mi \in \{0, 1\}$ and Fi is an array that contains values of features or P predictors as, $Fi = (fi1, \ldots, fiP)$. However, Fi have 82 hybrid features that contain permissions, API calls, Data operations, cryptographic operations, and network features. Additionally, a machine learning tool WEKA [35] is applied to rank features that help us in the feature selection process. The types of famous attribute selection algorithms include such as correlation-based, information gain, learner-based, and a principal component. However, in this study, InfoGainAttributeEval [36], is used to calculate the information gain score of extracted features. Furthermore, the information gain score evaluates the worth of each attribute as it is called entropy. For an experiment, it helps us to select features from the extracted ones.

4 Results and Discussions

This section describes experimental results from the analysis of benign and malicious datasets. In the first instance, features and their trends among botnet and benign applications are compared by considering static and dynamic features, individually. The datasets are divided into (a) train and (b) test datasets. In which a test dataset helps to validate the proposed hybrid analysis model. Furthermore, test results are validated through a machine learning tool such as WEKA [35].

4.1 Observed Static Features among Botnet and Benign Applications

In this subsection, the frequently used static features are discussed. The usage rate of the below-mentioned features are surprisingly high in botnet applications as compared to benign.

4.1.1 Permissions

In Fig. 4, permissions that are frequently in the practice of botnet attackers are compared with benign applications. The permissions used by botnet applications are INTERNET, ACCESS_NETWORK_STATE, and READ_PHONE_STATE shows 100%, 88%, and 86% rates, respectively. In contrast, these permissions in benign applications are utilized at a fewer rate with 85%, 53%, and 22%, exclusively. These permissions required a network connection to launch an attack through C&C. In addition, the attacker tries to detect the current phone state that helps to start conciliation with cell phones. Besides, other permissions such as READ_LOGS, READ_SMS, SEND_SMS, RECEIVE_SMS, and READ_CONTACTS are also observed. Botnets having C&C mechanism are used in SEND_SMS permission to send messages at premium-rate numbers with 61%. In comparison, only 1.3% of SEND_SMS permission is used in benign applications. Similarly, another permission READ_CONTACTS indicates 67% usage in botnet whereas only 8% in benign applications. Furthermore, INTSALL_SHORTCUTS and ACCESS_COARSE_LOCATION permissions are used at a very low rate in botnet as well as in benign applications with 5%, 23%, and 1%, 17%, individually.

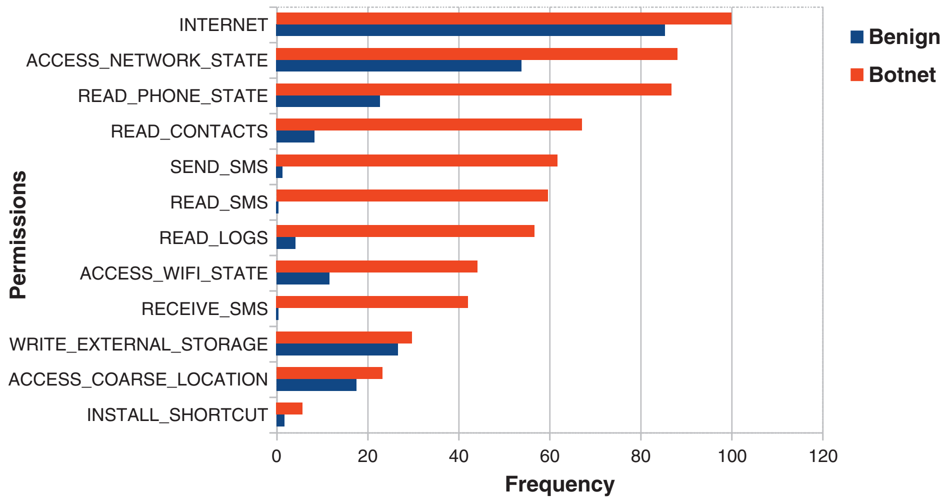

Figure 4: Usage of permissions analysis in a botnet and benign applications

4.1.2 API Calls Graph

In Fig. 5, API calls are analyzed, 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% ratios, separately in malicious applications. Conversely, benign applications utilize these API calls with only 10%, 1%, and 31% ratios, individually. This information is required to get identification and the current network state of devices and send it to a remote server. Moreover, *getCellLocation ()* and *getDefault ()* are used to track users' active locations to exploit them. The above-mentioned API calls are in the practice of botnet applications with 16% and 17% while benign applications only used them with 0.3% and 0.6%, respectively. The API calls such as *getLine1Number ()* and *getSimSerialnumber ()* are equally used by a botnet and benign application with 3% ratio.

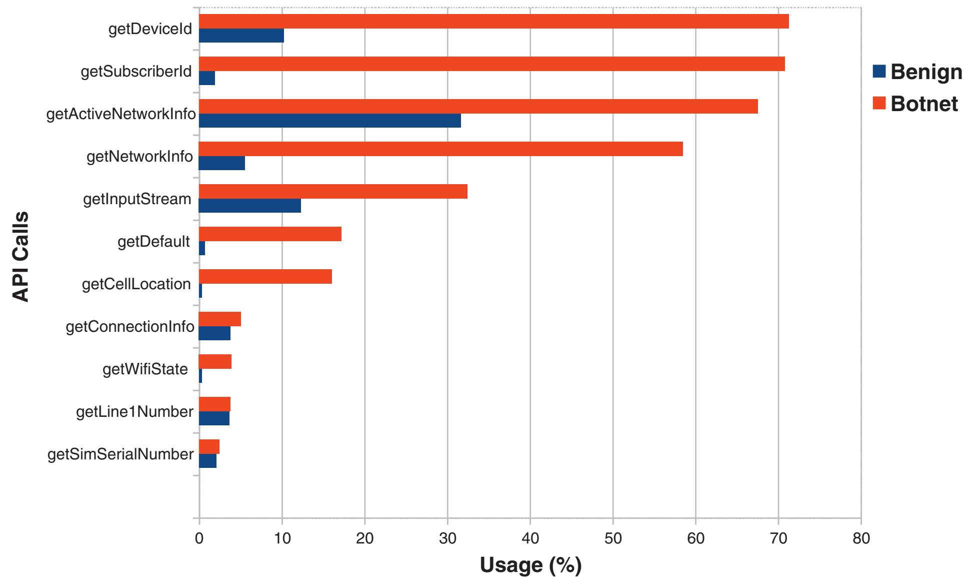

Figure 5: Usage of API calls analysis in a botnet and benign applications

4.2 Observed Dynamic Features among Botnet and Benign Applications

In dynamic features selection, results show the rising trend of botnet applications with the features array such as network traces and file operations.

4.2.1 Network Traces

In Fig. 6, the most commonly used network traces features in botnet applications are detected. The *open_conn* and *con_attempt* features 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 a 4628 and 383 times usage rate, respectively. Similarly, botnet having a C&C mechanism causes *network_leaks* with the frequency of 862, conversely, in benign applications, its occurrence is 164 times. In addition, *http_conversations* that create a connection between browser and web servers are observed. It contains information about data, source and destination address, and time stamp. The *http_consversations* ratio in benign and botnet applications has an average of 5.6% and 5.3%, individually.

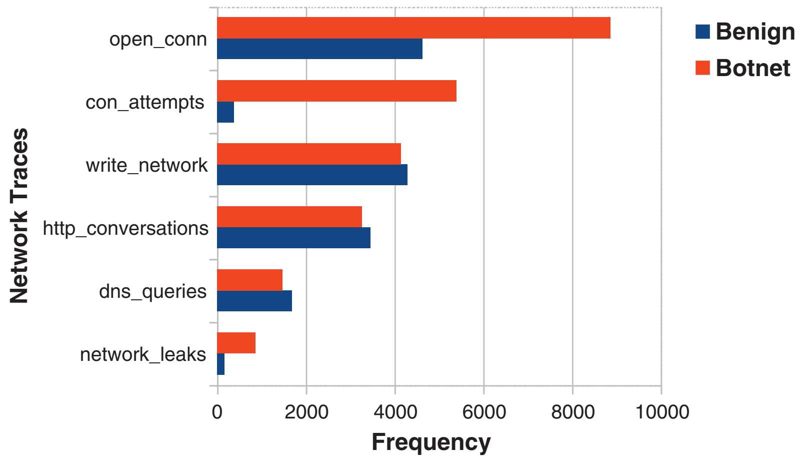

Figure 6: Network traces frequency analysis

4.2.2 File Operations Graph

Figs. 7–9 depict the results of file operations. In which *file_write*, *file_read*, and *file_leaks* are extensively are in the practice of malware authors.

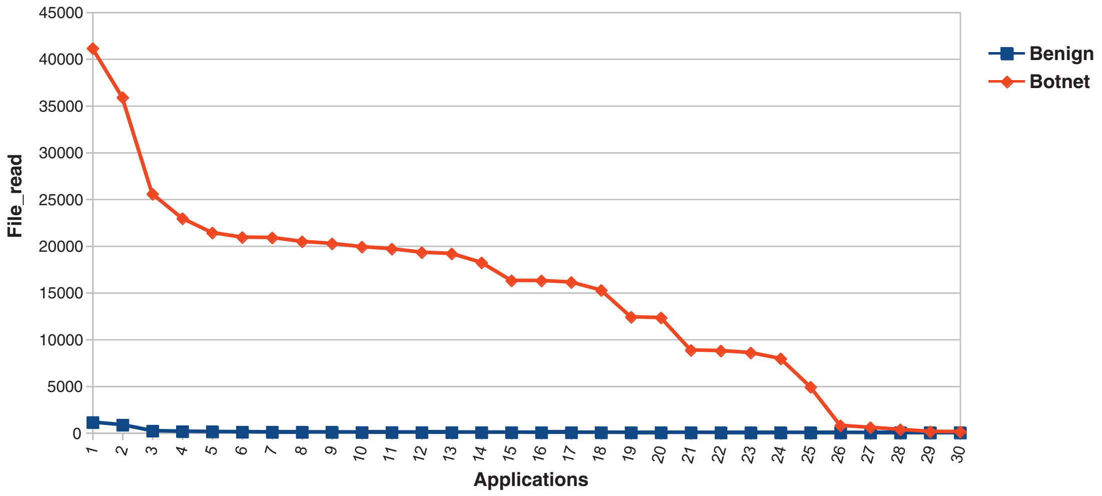

Figure 7: File read analysis in a botnet and benign applications

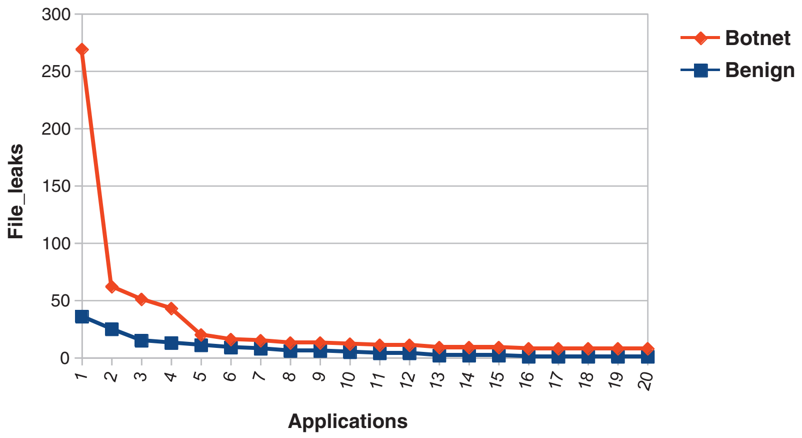

Figure 8: File leaks analysis in a botnet and benign applications

In Fig. 7, the top applications in the context of *file_read* operations are observed. Botnet applications cause to read files and steal sensitive data [37]. Furthermore, *file_read* in botnet applications is surprisingly high in the opposite of benign with an average of 766 and 11 times, identically.

Fig. 8, depicts *file_leaks* operation in the top 20 botnet and benign applications. Botnet applications cause *file_leaks* in which sensitive data such as device information, account login details are stolen by botnet authors. The average rate of *file_leaks* in botnet and benign applications are 244 and 44 times, individually. In the *File_write* operation botnet writers additionally add some malicious content in files and automatically write multiple files in data storage that causes to use of excessive memory.

In Fig. 9, the top 131 applications used the aforementioned features extensively to perform harmful operations. The usage rate of *file_write* operations in a botnet and benign applications are more than 1600 and 1200 times, individually. The average rate of *file_write* in benign and botnet applications is

281 and 59 times, respectively. Moreover, *file_write* in some applications are above 1000 and 200 times, in botnet and benign applications, separately. In file operations, botnet applications use the *file_write* to write malicious binaries in external storage.

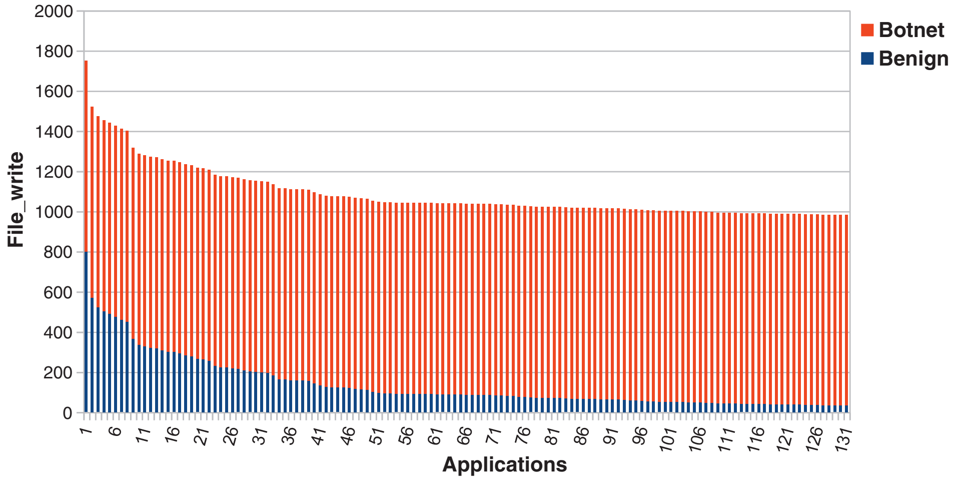

Figure 9: File write analysis in a botnet and benign applications

4.2.3 *Crypto Operations Statistics*

In Cryptographic operations, the encrypted code is used to secure communication between users and 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 to undetectable from applications. Likewise, malware writers used encrypted malicious code to evade from detection mechanisms. In Fig. 10, crypto_operations in a botnet and benign applications are observed. It shows that crypto_operations are in more practice of botnet applications as compared to benign applications. Botnet applications initiate twenty-two crypto operations, while benign applications used it only fifteen times. The average rate of cryptographic operations in botnet and benign applications are 2011 and 276 times, respectively.

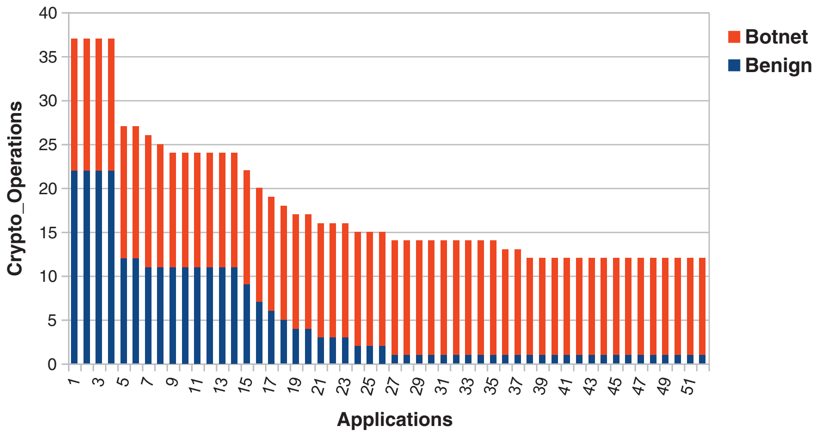

Figure 10: Crypto operations analysis in a botnet and benign applications

4.3 Effectiveness & Performance

The performance of machine learning classifiers is measured in terms of true positive rate (TPR), false-positive rate (FPR), Recall, Precision, F-Measures, and Accuracy, whereas for effective testing, 10 fold cross-validation method is selected. The empirical results are presented in Tabs. 5–7.

Table 5: Static analysis results with 10 fold cross validation

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%

Table 6: Dynamic analysis results with 10 fold cross validation

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 7: Hybrid analysis results with 10 fold cross validation

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%

In Tab. 5, the static analysis results of different machine learning classifiers with their evaluation parameters are described. Among the aforementioned classifiers, Naïve Bayes gives the lowest accuracy at 89.78%, while RF proves as the best classifier with an accuracy of 89.78%.

Tab. 6, demonstrates the dynamic detection rate of android botnet applications with various machine learning classifiers. The RF provides maximum accuracy with 95.12% in comparison with other classifiers.

The result of the hybrid analysis is shown in Tab. 7, with a 10-fold cross-validation method. The RF classifier outperforms, achieved the highest detection rate with 97.48% accuracy and the lowest false positive rate with 0.034. Moreover, other classifiers such as simple logistic, J48, and multilayer perceptron have an accuracy of 96%.

4.4 Comparison with Existing Approaches

In accordance with, our proposed botnet detection model is compared with different existing approaches. In literature, most researchers only adopted static or dynamic analysis to perform their experiments. In Fig. 11, the accuracy of our proposed hybrid model is compared with others. For instance, Da Costa et al. [9], detected botnet applications by implementing a dynamic analysis method, and wherefore it gives only 92.9% accuracy. Additionally, another researcher [8], follows a static analysis method with three noticeable static features like permissions, broadcast receivers, and background services, and provides 95.1% accuracy. However, in this work hybrid method is applied to detect android botnet applications having C&C ability. In this way, our proposed model shows 97.48% detection accuracy with prominent features.

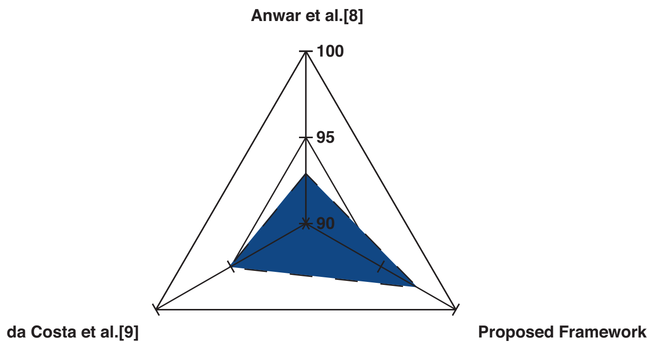

Figure 11: Comparison with existing approaches

5 Conclusion and Future Directions

In this paper, a hybrid learning-based mobile botnet detection model is proposed. The model is divided into four modules (1) data collection (2) analysis module (3) features module and (4) machine learning module. In the hybrid analysis, prominent static and dynamic features such as permissions, API calls, network traces, data operations, etc. respectively, are considered. Furthermore, frequency analysis graphs indicate that botnet attacks are in the practice of malware writers and make illicit use of multiple features.

Consequently, heterogeneous machine learning classifiers are applied. The Random Forest (RF) achieved the best accuracy with 97.48% and minimum false positive rate (FPR) with 0.034 in the hybrid analysis. In future directions, it is recommended to download applications from the trusted app store as Google play store [23], install appropriate mobile security tools, educate people, and government should make strict policies about cybersecurity.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] StatcounterGlobalstats, “Operating system market share worldwide,” [Online]. Available: http://gs.statcounter.com/os-market-share#monthly-201701-201805-bar (accessed May 29, 2018).

[2] Statista, “Number of apps available in leading app stores as of 1st quarter,” [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ (accessed May 29, 2018).

[3] R. Nigam, “A timeline of mobile botnets,” [Online]. Available: https://www.virusbulletin.com/virusbulletin/2015/03/timeline-mobile-botnets (accessed May. 29, 2018).

[4] C. Cimpanu, “Chinese crooks assembling massive botnet of nearly 5 million Android devices,” [Online]. Available: https://www.bleepingcomputer.com/news/security/chinese-crooks-assembling-massive-botnet-of-nearly-5-million-android-devices/ (accessed May 29, 2018).

[5] McAfee, “Mobile Threat Report Q1, 2018,” [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf (accessed May 29, 2018).

[6] G. Kirubavathi and R. Anitha, “Structural analysis and detection of android botnets using machine learning techniques,” *International Journal of Information Security*, vol. 17, no. 2, pp. 153–167, 2012.

[7] A. Karim, R. Salleh and M. K. Khan, “SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications,” *PloS One*, vol. 11, no. 3, pp. e0150077, 2016.

[8] S. Anwar, J. M. Zain, Z. Inayat, R. U. Haq, A. Karim *et al.,* “A static approach towards mobile botnet detection,” in *3rd Int. Conf. on Electronic Design (ICED)*, Phuket, Thailand, pp. 563–567, 2016.

[9] V. G. T. da Costa, S. Barbon, R. S. Miani, J. J. P. C. Rodrigues and B. B. Zarpelão, “Detecting mobile botnets through machine learning and system calls analysis,” in *IEEE Int. Conf. on Communications (ICC)*, Paris, France, pp. 1–6, 2017.

[10] M. Yang and Q. Wen, “A multi-level feature extraction technique to detect mobile botnet,” in *2nd IEEE Int. Conf. on Computer and Communications (ICCC)*, Chengdu, China, pp. 2495–2498, 2016.

[11] Pearltrees, “Anubis-malware analysis for unknown binaries,” [Online]. Available: http://www.pearltrees.com/u/4051585-malware-analysis-binaries (accessed May 29, 2018).

[12] CopperDroid, [Online]. Available: http://copperdroid.isg.rhul.ac.uk/copperdroid/ (accessed May 29, 2018).

[13] B. A. Store, “One stop store downloading & managing PC apps,” [Online]. Available: http://pcappstore.baidu.com/en/index.php (accessed May. 29, 2018).

[14] The Drebin Dataset, [Online]. Available: https://www.sec.cs.tu-bs.de/~danarp/drebin/ (accessed May 29, 2018).

[15] D. A. Girei, M. Ali Shah and M. B. Shahid, “An enhanced botnet detection technique for mobile devices using log analysis,” in *22nd Int. Conf. on Automation and Computing (ICAC)*, Colchester, UK, pp. 450–455, 2016.

[16] M. Eslahi, M. Yousefi, M. V. Naseri, Y. M. Yussof, N. M. Tahir *et al.,* “Cooperative network behaviour analysis model for mobile Botnet detection,” *IEEE Symp. on Computer Applications & Industrial Electronics (ISCAIE)*, Penang, Malaysia, pp. 107–112, 2016.

[17] S. Bojjagani and V. N. Sastry, “Stamba: Security testing for Android mobile banking apps,” in *Advances in Signal Processing and Intelligent Recognition Systems*. Cham: Springer, pp. 671–683, 2016.

[18] T. Strazzere and T. Wyatt, “Geinimi trojan technical teardown,” [Online]. Available: https://androidcommunity.com/wp-content/uploads/2011/01/Geinimi_Trojan_Teardown.pdf (accessed June 2, 2018).

[19] MalGenomeProject, “Android malware genome project,” [Online]. Available: http://www.malgenomeproject.org/ (accessed May 29, 2018).

[20] R. A. Al-Dayil and M. H. Dahshan, “Detecting social media mobile botnets using user activity correlation and artificial immune system,” in *7th Int. Conf. on Information and Communication Systems (ICICS)*, Irbid, Jordan, pp. 109–114, 2016.

[21] W. Hijawi, J. Alqatawna and H. Faris, “Toward a detection framework for Android botnet,” in *Int. Conf. on New Trends in Computing Sciences (ICTCS)*, Amman, Jordan, pp. 197–202, 2017.

[22] Cybersecurity, “Canadian institute for cybersecurity,” [Online]. Available: http://www.unb.ca/cic/datasets/index.html (accessed May 29, 2018).

[23] G. Play, “Google play,” [Online]. Available: https://play.google.com/store?hl=en (accessed May 29, 2018).

[24] VirusTotal, [Online]. Available: https://www.virustotal.com/#/home/upload (accessed May 29, 2018).

[25] Malware Security blog, [Online]. Available: http://artemonsecurity.blogspot.com/ (accessed May 29, 2018).

[26] NJCCIC, “WireX,” [Online]. Available: https://www.cyber.nj.gov/threat-profiles/botnet-variants/wirex (accessed May 29, 2018).

[27] NJCCIC, “RottenSys,” [Online]. Available: https://www.cyber.nj.gov/threat-profiles/android-malware-variants/rottensys (accessed May 29, 2018).

[28] Contagio Mobile, [Online]. Available: http://contagiominidump.blogspot.com/ (accessed May 29, 2018).

[29] GitHub, "androguard," [Online]. Available: https://github.com/androguard/androguard/ (accessed May 29, 2018).

[30] GitHub, "apkinspector," [Online]. Available: https://github.com/honeynet/apkinspector/ (accessed May 29, 2018).

[31] L. K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis," in *21st USENIX Security Symp. USENIX Security 12*, Bellevue, WA, pp. 569–584, 2012.

[32] GitHub, "DroidBox," [Online]. Available: https://github.com/pjlantz/droidbox (accessed May 29, 2018).

[33] Android, "APK Analyzer," [Online]. Available: https://developer.android.com/studio/build/apk-analyzer (accessed May 29, 2018).

[34] K.c. Andrubis, "Scan & analyze Android apks," [Online]. Available: http://hackpla.net/anubis-scan-android-apks/ (accessed May 29, 2018).

[35] Weka, "Weka 3: Data mining software in Java," [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/ (accessed May 29, 2018).

[36] S. Y. Yerima, S. Sezer and G. McWilliams, "Analysis of Bayesian classification-based approaches for Android malware detection," *IET Information Security*, vol. 8, no. 1, pp. 25–36, 2013.

[37] S. Anwar, M. F. Zolkipli, Z. Inayat, J. Odili, M. Ali *et al.,* "Android botnets: A serious threat to Android devices," *Pertanika Journal of Science & Technology*, vol. 26, no. 1, pp. 37–70, 2018.