Android Botnet Detection Using Hybrid Analysis

Mamoona Arhsad[1*], and Ahmad Karim[1]
[1] Department of Information Technology, Bahauddin Zakariya University,
Multan, 60000, Pakistan
[e-mail: msmamoona@outlook.com, ahmadkarim@bzu.edu.pk]
[*]Corresponding author: Mamoona Arshad

Received June 24, 2022; revised August 2, 2023; accepted January 1, 2024; published March 31, 2024

Abstract

Botnet pandemics are becoming more prevalent with the growing use of mobile phone technologies. Mobile phone technologies provide a wide range of applications, including entertainment, commerce, education, and finance. In addition, botnet refers to the collection of compromised devices managed by a botmaster and engaging with each other via a command server to initiate an attack including phishing email, ad-click fraud, blockchain, and much more. As the number of botnet attacks rises, detecting harmful activities is becoming more challenging in handheld devices. Therefore, it is crucial to evaluate mobile botnet assaults to find the security vulnerabilities that occur through coordinated command servers causing major financial and ethical harm. For this purpose, we propose a hybrid analysis approach that integrates permissions and API and experiments on the machine-learning classifiers to detect mobile botnet applications. In this paper, the experiment employed benign, botnet, and malware applications for validation of the performance and accuracy of classifiers. The results conclude that a classifier model based on a simple decision tree obtained 99% accuracy with a low 0.003 false-positive rate than other machine learning classifiers for botnet applications detection. As an outcome of this paper, a hybrid approach enhances the accuracy of mobile botnet detection as compared to static and dynamic features when both are taken separately.

Keywords: Machine Learning, Botnet Applications, Malware, Dynamic, Static

This research is not supported by any research grant and there is no conflict of interests among all others

1. Introduction

Nowadays, Android devices have become increasingly significant in modern life because of the capabilities provided by smartphones and the explosive growth in computational capacity. People prefer to do financial transactions on their cell phones or mobile devices and save sensitive data on handheld devices as a substitute for PCs [1]. Because of the nature of its operating system ecosystem, mobile devices have become a specific target for cybercriminals. In addition, smartphone users can get apps from google play stores, third-party and web browsers such as torrents, and direct Internet [2].

The research study [3] demonstrated that the third-party market contains 5-8 percent of malicious apps. In third-party platforms, programmers and developers can lunch any dangerous or clean applications in the market. For this purpose, the Android platform provides various security platforms such as Bouncer as the first security step for all apps [4]. Unfortunately, cybercriminals can still modify the security measures and use malicious app installation methods for the attack [5, 6]. Moreover, cyber attackers publish their harmful softwares on the third parties platforms in the form of malicious codes. In most cases, a platform of Android apps offers apps that are free, non-paying, or less expensive than the Google Play Store.

Android malware is designed to infect mobile phones rather than PCs for gaining access to Android devices without the user's knowledge. Meanwhile, a mobile botnet refers to a network made up of a group of infected mobile devices, controlled by a self-replicating backdoor program [7]. Eventually, it allows hackers to remotely manipulate mobile devices and execute orders to carry out malicious actions with the use of a platform, such as a Command and Control (C&C) server to control and instruct bots. In addition, attackers or hackers are known as botmasters that control their command-and-control channel to send, update, and obtain information about end users. These types of attacks can infect Android smartphones and turn into harmful bots. These malicious bots turn into large botnets. The botnets are usually divided into subgroups such as botmasters, bot clients, and bot servers. The botmaster is in charge of the botnet's controller and operator known as malware management as shown in **Fig. 1**.

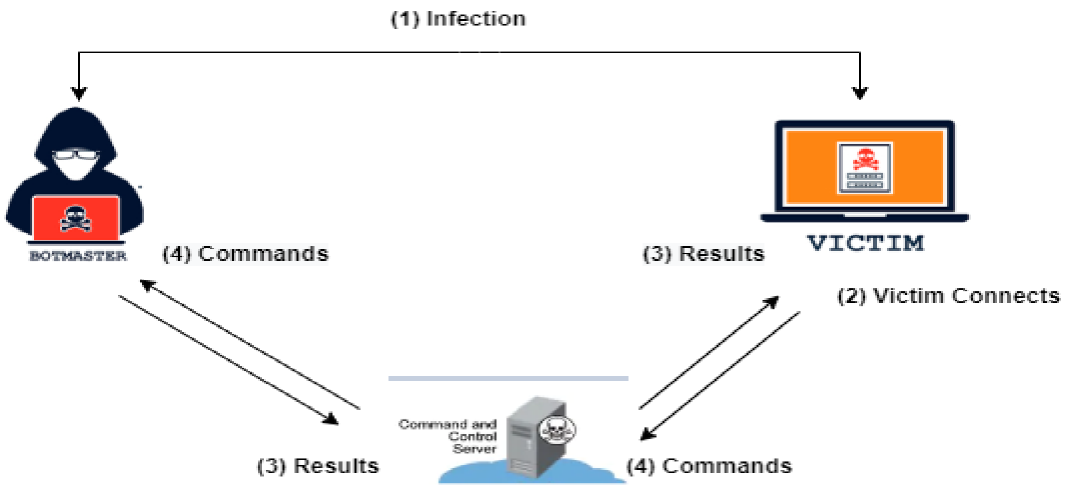

Fig. 1. Workflow of Botnet

The attackers (botmasters) launch an attack systematically. The botmaster's initial action is to infect a victim with a bot. The Bot Clients host the botnet wishes to command and control the botmaster's target device. Using C&C infrastructure, the C&C server accepts commands from the botmaster and controls the issues that are ordered to the bot client through protocols such as HTTP or IRC, used for establishing the connection between the server and the bot client. After establishing the connection, then the C&C server sends commands to the victims, who carry them out and report back to the C&C server. Consequently, the C&C server contains various malicious activities including phishing, spamming, virus, email clicking, advertising, and text messages without the owner's permission. Therefore, the main objective of this paper is the detection of malicious applications in devices through C&C servers that are arranged by botmasters [8].

Many researchers used two common techniques of mobile malware analysis such as code-based and runtime analyses. In static analysis or code based, the research study [9] states that evaluating codes of applications for defects, back doors, or other malicious activities could allow hackers to access sensitive data or client information. It starts with the disassembly of the program using reverse engineering tools and attributes such as byte sequences, Permissions, API calls, string codes, and function calls. In comparison, dynamic analysis [10] extracts features in runtime behavior that must be executed in a safe environment (called a sandbox).

In this paper, we have worked on both static and dynamic analysis approaches to Android applications by comparing them with malware, benign and botnet application. For this purpose, we collected the most popular and comprehensive features such as permissions and API calls used by previous researchers [9] for mobile malware detection in general. To extract and process features, employed the reverse engineering approach. After Feature Extractions, Used hybrid analysis technique and tested it with machine learning classifiers such as SVM with SMO, Naïve Bayesian, Random Forest, decision tree, and Multilayer Perceptron (MLP). Our paper on a hybrid analysis is that it is a more successful approach of detecting mobile botnet applications. The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 describes the proposed methodology with reverse engineering, feature selection and extraction techniques. Section 4 presents the results and discussion with the past papers analysis. Section 5 concludes the paper and Future work.

2. Related Work

In recent years, the field of Android malware analysis has attracted the attention of many researchers. Generally, the majority of existing solutions are designed based on static and dynamic analysis for mobile malware in general rather than mobile bot malware. In contrast, hybrid analysis is a combination of static and dynamic analyses in which relevant features are extracted and the analysis produces a significantly better outcome. For this purpose, we have used a hybrid analysis approach to avoid the limitation of static and dynamic analysis such as code obfuscation technique and virtualized environment.

The motivation of researchers behind malware detection is different e.g., to acquire unauthorized access, obstruct resource use, demand a ransom, use a root vulnerability, spread spam, and create a botnet. Therefore, in this paper, we mainly discuss the Android botnet dilemma, which has gained attraction in recent years due to cybercriminal attacks on smartphones and handheld devices. Moreover, cybercriminals can launch different attacks by creating a single bot network device on mobile devices through Controller (C&C). As a result, we compared our results with mobile malware in general because the Android botnet problem is still not as well-known as the PC-based botnet.

Malicious hackers frequently use hazardous permission sets to exploit devices by taking full advantage of people with a lack of understanding of the complexities associated with the user's permission. In [11], the authors used a static analysis approach, the prominent permissions are first extracted from AndroidManifest.xml of 436 Android applications on their dangerous usage. For the accuracy of results, the researchers used a future pruning method after feature extraction. However, the static analysis approach was not able to extract comprehensive code features and gives greater false positive results.

Choi et al. [12] presented another Android botnet detection approach using virtual private networks (VPN). The authors used VPN to observe the total number of bytes and packets by investigating C&C communication flow. Through the communication flow of the C&C channel, they observed the characteristics of Android botnet applications. On the other hand, another study [13] used a static analysis approach for the detection of zero-day attacks in Android applications.

In [14], the authors proposed a hybrid analysis approach called Droid Ranger. Initially, the application binaries are chosen based on the use of the hazardous permission set. Then, the behavior of malicious binaries is compared with known malware samples based on Android applications manifest, packages, function call graphs, and code execution.

In a dynamic analysis approach called Vet Droid [15] in which the authors selected harmful permissions from applications. Firstly, the authors extracted all permissions from the component VetDroid and make a relationship among the features. Another approach [17] presented a framework of static analysis for the detection of Android malicious binaries. They selected features for analysis such as permissions, functions, and intents. Next, the authors pointed out the significance of short messages (SM) in the detection of malicious applications.

The authors [18] designed an anomaly-based approach working for the detection of Android applications using system calls. They used machine learning classification algorithms on host machines to identify unusual behavior in Android apps using system calls as a feature vector. Furthermore, it uses dynamic aspects of known malware (self-generated) and machine learning to detect botnet behavior. The approach can detect malicious binaries with 92 % accuracy using only dynamic information as input.

In a recent study [19] the authors used the Vennabers predictor, K-nearest neighbor (KNN), and Kernel Density Estimation to study botnet network traffic flow (KDE). They examined botnet families based on HTTP, IRC, and P2P that analyze network events based on the botnet life cycle. In another research study [20], the authors described a hybrid analytical strategy that combines static and dynamic analysis. The evaluation findings demonstrate that hybrid analysis outperforms through different machine learning classifier techniques, including J48, Naive Bayes, Random Forest, and Logistic Regression. This research has a few similarities to our methodology. However, we have used 100 instances of botnet, benign, and malware applications whereas the said approach used 30 instances of Android botnet applications. The authors claimed that random forest outperforms as compared to other classifier models with a classification accuracy of more than 90%. On the other hand, we are addressing hybrid aspects of Android applications utilizing machine learning techniques of SVM with SMO, MLP, Random Forest, Naïve Bayesian, and decision trees to detect botnet applications.

3. Proposed Methodology

In this section, we describe the proposed methodology of Android botnet detection. It is divided into three stages: Data collection, Data preparation, and Classifiers Evaluation. In the first stage, we will collect Android applications from different categories which include benign botnet, and malware set of applications. In the second stage, applications are prepared to carry out analysis, initially reverse-engineered, reconstruct the source code of application and then the apps are examined using a hybrid analysis technique. Usually, apps are passed through a hybrid analysis approach to extract and select static [21] and dynamic features [10] using a self-developed Python script that uses self-regulating tools such as Androguard [16] and Droidbox [22]. In addition, the extracted features are stored in the comma-separated values files. After feature extraction, Classifiers Evaluation is employed to train the various ML classifiers such as SVM, DT, RF, NB, and ANN for detection of the Android botnet application. In the last stage, we will analyze the results of classifier validation produced by using machine learning classifiers. **Fig. 2** shows the workflow of the proposed methodology.

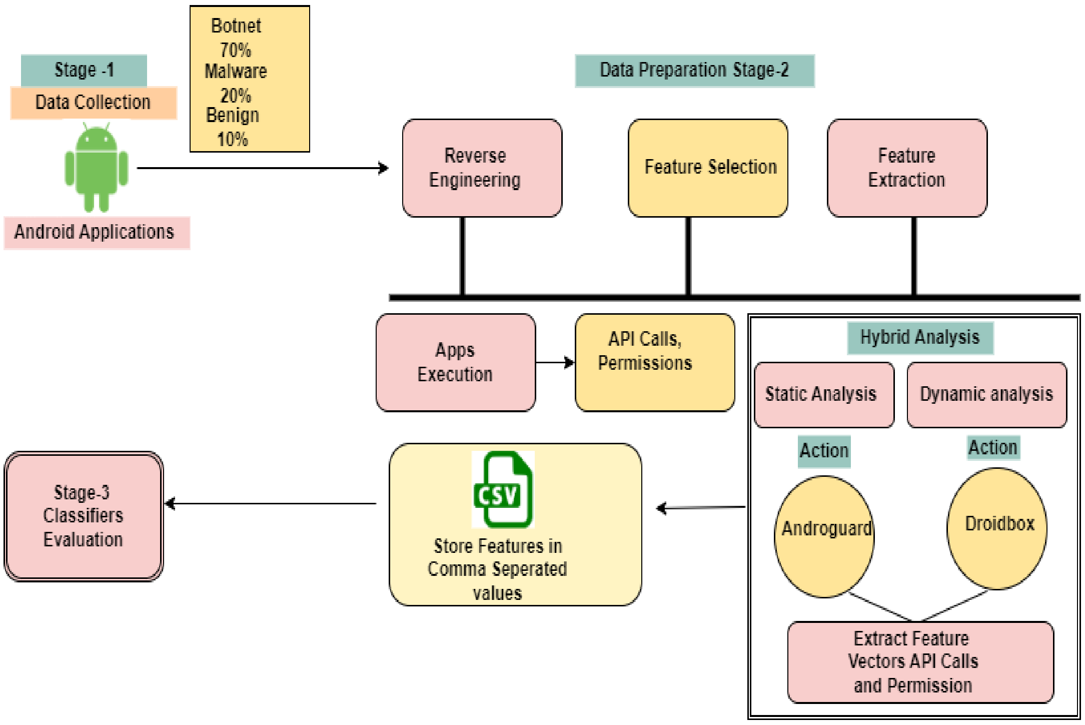

Fig. 2. Workflow of Proposed Methodology

3.1 Data Collection (Stage 1)

The first phase of the proposed methodology is data collection. To complete our analysis work, we have gathered 100 samples of mobile applications from various categories (benign, malware, and botnet) and performed a hybrid analytic strategy that combines static and dynamic techniques. For an experiment, we consider a dataset of real Android botnet and malware [23] which is the largest dataset freely available on the internet. Meanwhile, for benign, 10 samples are obtained from internet repositories, including the Google Play Store. **Table 1** shows the summary of the dataset.

Table 1. Summary of dataset

Samples	Number of apps	Observation	Features
Botnet	70	Static/dynamic	56
Malware	20	Static/dynamic	56
Benign	10	Static/dynamic	56

3.2 Data Preparation (stage 2)

In this stage, we have focused on static and dynamic feature extraction and selection analysis in Android devices for data preparation. For this purpose, we have used a reverse engineering technique which is based on machine code that is called android package kit (APK) and saved as a zip file. To access the contents of the Android package File, we have used the Android asset packaging tool (AAPT) available within the Android SDK. An APK File contains normal Classes.dex, Android Manifest.XML, res, lib, and assets folders.

We have performed our experimentation on a virtual machine of the SANTOKU operating system (a Linux distribution system) that is built primarily for mobile analysis, using hardware with Intel(R) Core (TM) and 16GB RAM. More specifically, the proposed framework was implemented using Android SDK, AAPT, Androguard, and Droid box tools for Data Preparation of Android applications. The basic framework of Data Preparation is illustrated in **Fig. 3**.

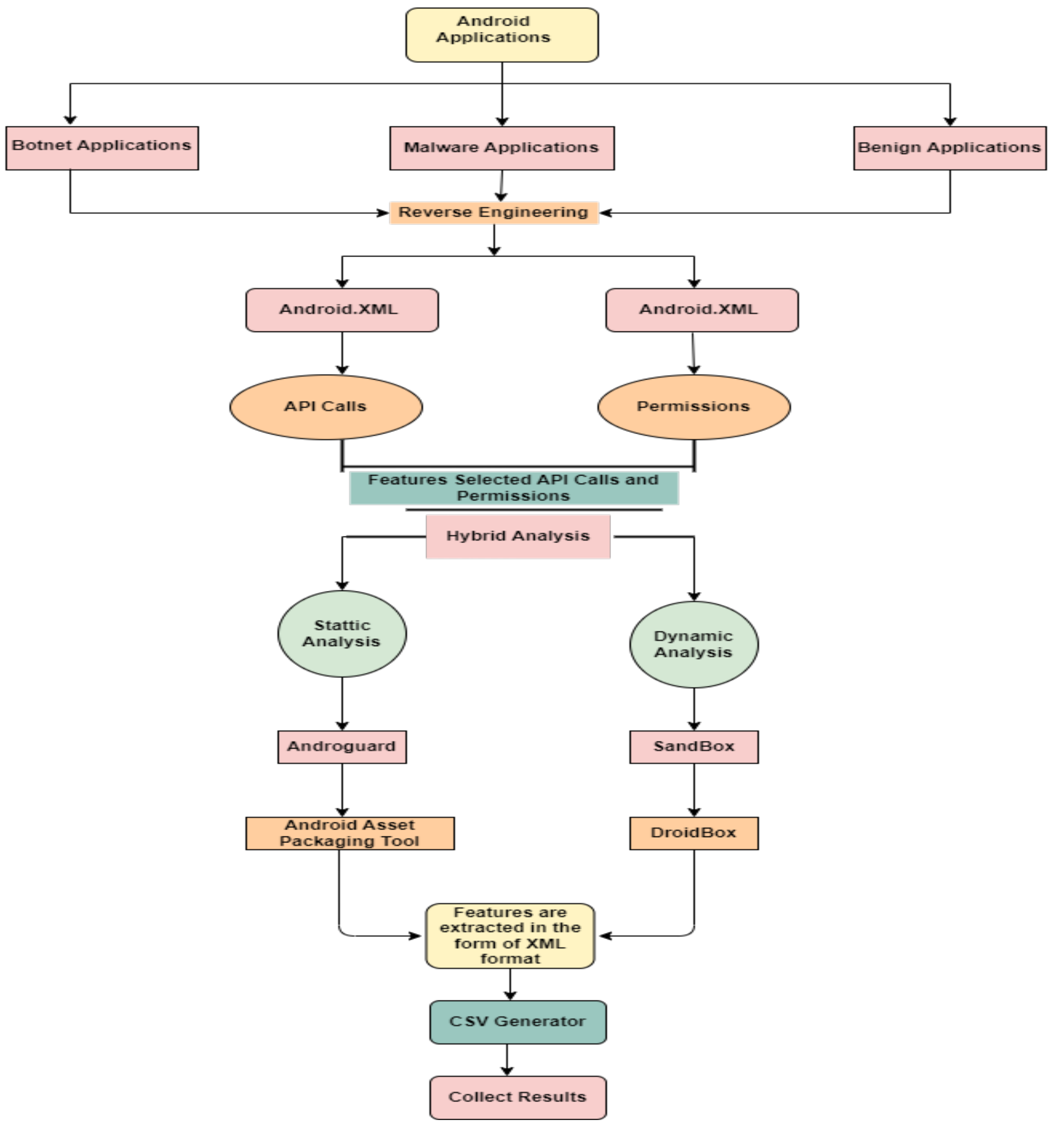

Fig. 3. Dataset Preparation

3.2.1 Features Selection

During the review process of applications, we have selected Classes. dex and android manifest.XML files in our work. In addition, Features are analyzed using static and dynamic analysis approaches. Static analysis is a lightweight approach as compared to the dynamic analysis approach. In the first step, we examine the feature set values associated with botnet, benign, and malware apps, reverse engineering them, and extract Classes. dex and Android Manifest.XML Files. To the best of our knowledge, Classes. dex includes information about API calls, whereas the Android Manifest file contains data about permissions and intents, and the remaining are command strings. For botnet detection, we choose Permissions and API Calls for the analysis of malicious applications.

- **Permission:** The main objective of permissions is to protect the privacy of users. Initially, the apps must seek approval for permission to access sensitive information and system attributes from the user. Sometimes, the system provides approval for permissions on its own or could encourage the users to approve the request. As we already mentioned in the feature selection process, Permission is used in the AndroidManifest. XML. For our analysis, a total of 13 permissions are selected from androidmnifest.xml using a self-developed Python script from the existing literature and Android official sites which indicates that these features are more prevalent in harmful applications.[10, 24]
- **API Calls:** API Calls are used for interacting with the Android device. It contains methods, classes, and packages that developers can use to create apps. The source code of the Android programming language is based on Java and transforms source code into Java bytecode. After the decompilation of Java bytecode, it uses Dalvik Virtual Machine (DVM) to provide information about packages, methods, and classes. For our analysis, we choose aapt tool to extract API Calls from classes.dex files. In addition, a total of 26 API calls are selected using a self-developed Python script and stored in a CSV file for further analysis.

3.2.2 Feature Extraction

In this section, we have analyzed static and dynamic features to identify interesting characteristics of benign, botnet, and malware applications.

3.2.2.1 Static Features

In static analysis, our feature set values contain API calls and permissions having a close relationship with botnets, malware, and benign application. For our work, we have observed the interrelationship phases between features of Permissions, API, and their reasoning for the botnet activities are described in **Table 2**. To extract features automatically, we used a Python script on each Android binary code and recorded all features in a CSV file for further work. The feature values in CSV files are binary numbers, with "1" and "0" denoting applications that enable or disable features. For this purpose, we have observed all enabled attributes to better classify our malicious dataset with clean programs. As a result, for our analysis, the features are saved as "1" if the application has a specific feature enabled and "0" otherwise. Assume that a and b are the number of apps and the set of features, respectively (which includes Permission and API calls). Similarly, the class of application instances in the produced dataset is assumed to be benign, malicious, or botnet which indicates the permission and API.

Let x and y be the number of applications and the set of features (including permissions and API calls, respectively. The feature vector for the application ($a_{i,1,}a_{i,2,}$ $a_{i,3}$………………… $a_{i,j}$) Where:

$$a_{ij} = \{1 \text{ if application x uses feature k otherwise } 0$$

Table 2. Selected Feature Set and Their Reasoning

Permissions	**API Calls**	**Reasoning**
INTERNET	openConnection(), execute(), connect(),openStream(), getInputStream(), Socket(), getContent()	In the first phase of the connection, Cybercriminals can connect to the internet and broadcast to the rest of the world through an open stream, Moreover, they receive information from users.
READ_ PHONE_ STATE	getDeviceId(), getSimSerialNumber(), getSubscriberId(), getLine1Number	After establishing the connection, Virus programmers obtain information about a phone's current state. It is a read-only Permission
ACCESS_ NETWORK_ STATE	getActiveNetworkInfo() getNetworkInfo()	For establishing the connection between the phone state and network state, this permission works like a bridge.
ACCESS COARSE LOCATION	getCellLocation()	For finding locations of network sources, the permission permits the malware to access the information.
SEND_ SMS	getDefault() sendTextMessage()	This permission allows the application to send SMS messages to C&C servers without the need for user participation.
ACCESS_ WIFI_ STATE	getCellLocation()	For getting the information of Wifi, the malware writers use this feature and send it to a remote location
ACCESS_ FINE_ LOCATION	getLastKnownLocation(), isProviderEnabled(), requestLocationUpdates()	This feature enables an application in the C&C server to get an accurate position from GPS, WIFI, or cell towers.
READ_ CONTACTS	openOutputStream() openInputStream() openFileDescriptor()	An application can read a user's contact information using this feature. After that, the information is given to the C&C server, who will carry out the attack
READ_ LOGS	exec()	An application can use this feature to access system log files.

Generally, after selecting features, we divided the operational stages of botnets into phases based on their malevolent actions. These states are characterized as (a) connection phase, (b) communication phase, and (c) status information phase. For instance, one of the common phases is the connection phase where applications have a malicious intention accompanied by INTERNET Permission and bulk of API Calls. For establishing the connection, the negotiations and intersections happen in C &C server mechanism for malicious intentions. For this purpose, cyber criminals use the direction of information and communication protocol. In the second phase, communication protocols are used for pushing and pulling information from users. These communication protocols of botnet applications are HTTP, IRC, and P2PP. Additionally, these protocols often lead to the start of harmful action. As a result, we have highlighted only those features that can lead to communication and assault initiation. For this purpose, we have selected READ_CONTACTS, READ_LOGS, and SEND_SMS Permissions in our paper. On the other hand, the API calls are openInputStream, openOutputStream, openFileDescriptor, exec, getDefault, and sendTextMessage. For getting the status information of the user, the cybercriminals must keep an eye on the device's active status as well as changing network conditions in the third phase of botnet applications.

3.2.2 Dynamic Features Extraction

Behavior or dynamic analysis is shipped with the app itself or loading at runtime to examine the behavior of apps. For this purpose, we have used the dynamic analysis framework of DroidBox which shows the behavior of Android applications of botnet, benign, and malware datasets. For example, network operations are used for establishing a remote connection between opened connections and the read/ write state, the majority of such behavior is caused to initiate a malevolent action in Android applications. Therefore, we have selected those features that cause botnet attacks on our work. In consequence, these features are file activity read, network operations, information leaks, services, SMS, DNS traffic, and HTTP traffic, described in **Table 3** in terms of features, parameters, and rationale.

Table 3. Dynamic Extracted Features

Features	**Parameters**	**Rationale**
File Activity Read	Read, Write	The applications are edited and changed by hacktivists who read the file activity for malevolent action.
Network Operations	Opened Connections, Network Read, Network Write	This feature is used for establishing the remote connection between network operations
Information Leaks	File Leaks, Data Leaks	It monitors the network for information and file leaks.
Services	Started Services	Malicious programs start background services.
SMS	Sent SMS	This capability is critical for identifying SMS-based botnets.
DNS Traffic	DNS Requests	A botnet assault is indicated by frequent DNS requests.
HTTP Traffic	HTTP Conversations, HTTP Connection attempts	This functionality is used by HTTP-based botnets to make TCP-based connections with the outside world.

4. Results and Discussion

In this section, we have examined and discussed the experiment. Initially, we observed code-based and runtime features among existing botnet, malware, and benign applications to emphasize the importance of botnet applications in Android devices. For this reason, we have used evaluation features for API and Permission in Android applications. In addition, we have employed learning-based detection of machine learning algorithms and compared the results with existing studies to strengthen our claim about the existence of malicious behavior in the case of clean applications.

4.1 Evaluations of Features

Android security architecture and applications have diverse representations when it comes to requesting permissions and API Calls. Botnet generally demands more permissions than malware applications or even seeks approval for permission sets on the Android platforms to various system and user resources. On the other hand, users are frequently unaware of the complexities and malicious effects of permissions in Android applications. Therefore, users have additional information to help them make the best decision possible because users install malicious applications unintentionally with a significant risk associated with them.

According to the above explanation, we have used the percentage of permissions requested by botnets, malware, and benign applications in our work. **Fig. 4** clearly shows that the number of requested permissions is larger in botnet applications as compared to malware applications. Simultaneously, it did not imply that botnet programmers would be able to take advantage of all capabilities. The reason for the larger number of permissions is that botnet programmers are attempting to evade detection by invoking capabilities through the code of another program, which reveals the increased number of permissions. To monitor malicious activities, botnet developers require permission attributes to start and build a remote connection to devices. For this reason, we observed those permissions that are utilized by botnet applications in our work. These permissions are INTERNET, READ_LOGS, WRITE_EXTERNAL_STORAGE, ACCESS_WIFI_STATE, and READ_SMS Etc.

As shown in **Fig. 4**. The INTERNET permission is used by botnet applications to maintain the remote connection of the C&C server. Moreover, we have observed that the percentage of Internet permissions is higher than the benign and malware applications. Another noteworthy aspect we have noticed is that the malware and botnet applications have the same malicious properties such as the HTTP-based C&C method. Here, 55.15% of botnet applications employ ACCESS_FINE_LOCATION which is used by C&C server to get an accurate position from GPS, WIFI, or cell towers. In contrast, malware applications employ 25% ACCESS_FINE_LOCATION network connectivity to launch the attacks. In contrast, according to our results, 61.42 % of botnet applications used ACCESS WIFI STATE for getting information about WiFi. Only 10% of malware, on the other contrary, use this privilege. Likewise, botnet programmers must be able to detect the current state of a cell phone That allows them to be aware of the current condition of the mobile device, and if it is active, the botmaster can commence communicating with the cell phone, According to our findings, READ PHONE STATE is used by 100% of botnet applications and 98% of malware applications for nefarious purposes. Only 40% of benign users, on the other hand, take advantage of this permission. Similarly, 82.28% of botnets employ ACCESS NETWORK STATE using ACCESS_NETWORK_STATE to establish the connection between phone state and network state this permission works like a bridge.

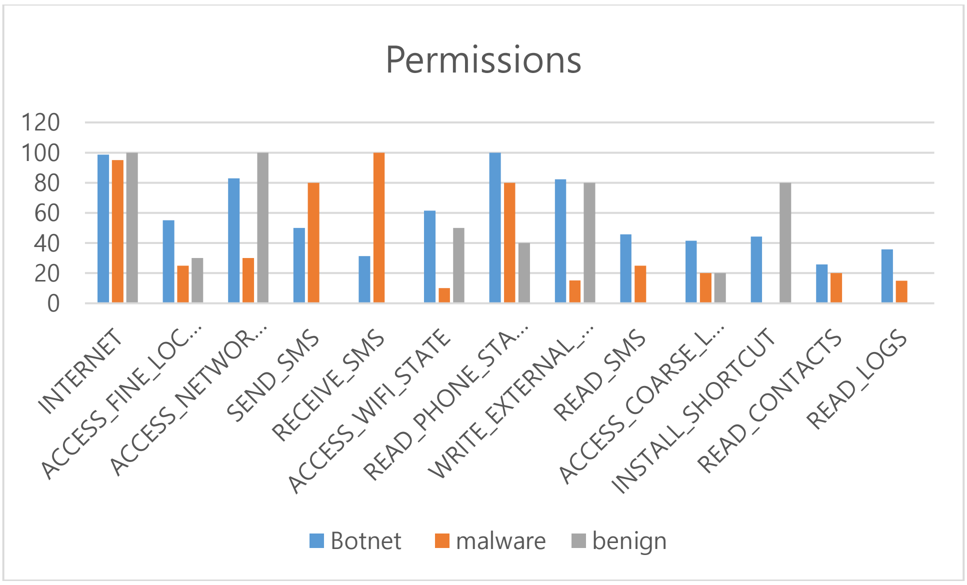

Fig. 4. Frequency Analysis of Permissions

We also looked at the API calls to see if there were any malicious code execution capabilities. The impact of dangerous API calls on malware, botnets, and benign applications is depicted in **Fig. 5**. To establish and disseminate botnet networks, API Calls usually have access to instructions like execute(), connect(), and openConnection(). Similarly, the botnet used the API functions getConnectionInfo(), getNetworkInfor(), getActiveNetworkInfo(),locationListener(), requestLocationUpdates(), getLastKnownLocation(), getLine1Number(), andgetDeviceID() to connect and obtain network information from the devices. On the other hand, for API requests files have the smallest impact on botnets, malware, and benign applications. Furthermore, botnet applications use the getLine1Number 42.13 % of the time, whereas malware applications use it 45%. Another crucial aspect of botnet applications is the ability to obtain and send bot identification information to remote hosts. This can be done through the following API calls: getSimSerialNumber() and getDeviceID(). Botnet applications utilize getSimSerialNumber() and getDeviceID() 31.42% and 42% respectively. Consequently, on average 17% of botnet applications use getLastKnownLocation API Calls. In contrast, on average 5% of malware and 0% of benign applications use getLastKnownLocation API. Similarly, 18% of applications use exec() for remote connection in comparison with 10% and 40% of malware and benign applications respectively.

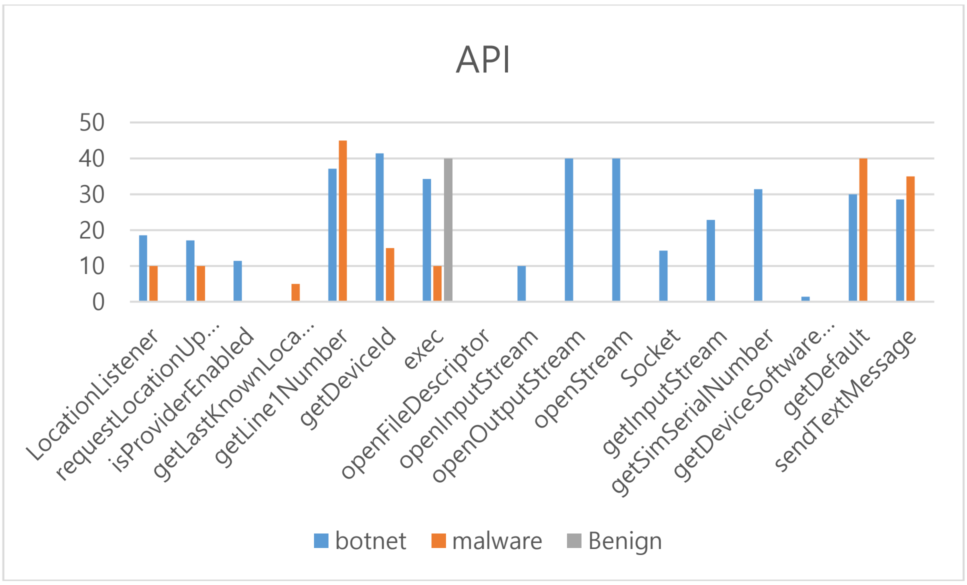

Fig. 5. Frequency Analysis of API Calls

4.2 Classifiers Evaluation (stage- 3)

In this phase, we demonstrate the effectiveness of the suggested explanation to validate our claim that hybrid analysis is an effective and efficient technique for detecting mobile botnet applications with API Calls and Permissions. Therefore, we have employed various machine algorithms such as Naive Bayesian, Random Forest, SVM with SMO, decision tree, and Multilayer Perceptron (MLP) as classification algorithms. To measure the results, we have presented the experimental results and the performance of the models in terms of True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, and Accuracy (ACC).
Table 5 demonstrate the evaluation of machine learning classifiers on benign, malware, and botnet dataset which consists of 100 instances. It observed that decision trees have the highest accuracy of 99%, followed by random forest and Naive Bayesian with 96 % and 94 % accuracy, respectively. Additionally, we indicated that decision trees outperformed other conventional classifiers.

Table 5. Evaluation Results of Machine Learning Classifiers

Classifier	TP Rate	FP Rate	Precision (%)	Recall(%)	F-Measure(%)	Accuracy(%)
Random Forest	0.960	0.071	0.961	0.960	0.960	96
Decision Tree	0.990	0.003	0.990	0.990	0.990	99
SVM with SMO	0.950	0.075	0.950	0.949	0.949	95
Naïve Bayesian	0.940	0.055	0.942	0.940	0.940	94
MLP	0.950	0.053	0.950	0.949	0.950	95

4.3 Comparison with other existing studies

In this section, we evaluate the hybrid analysis framework with previous research studies to emphasize the importance of our work. As previously stated, there seems to be a lack of studies on mobile botnet detection employing hybrid analysis and machine learning classifiers. To the best of our knowledge, static and dynamic approaches are used in existing methodologies, depending on the properties of API Calls and Permissions. For that reason, direct comparison is not appropriate. However, we can compare the results in terms of accuracy, techniques, and features.

Table 6. Comparison with other existing studies

Reference	Technique	Features	Accuracy
[25]	Static	Permissions & API_Calls	94.83%
[26]	Static	Permissions	92.10%
[27]	Static	Permissions	89.30%
[28]	Static	Permissions & API_Calls	96%
Purposed framework	Hybrid Analysis	Permissions & API_Calls	99%

As shown in **Table 6**, the purposed framework is a more effective and efficient technique rather than Rashidi & Fung (2016), who used the permissions feature to obtain 89.30%. Another research study, by Sanz et al. (2013), used the permissions and API Calls features to achieve 94.83 % accuracy. Yerima et al., (2014) attained an accuracy of 92.10 percent using features extraction technology. The proposed framework is more accurate than Rashidi & Fung, (2016), Perivian & Zhu (2013), Sanz et al., (2013), and Yerima et al., (2013). The better outcomes of the suggested framework are due to the utilization of a large number of benign, botnet, and malware applications, as well as the features selection approach.

Fig. 6 demonstrated the accuracy displayed by a bar graph. It revealed that the proposed framework has high accuracy when compared to other studies.

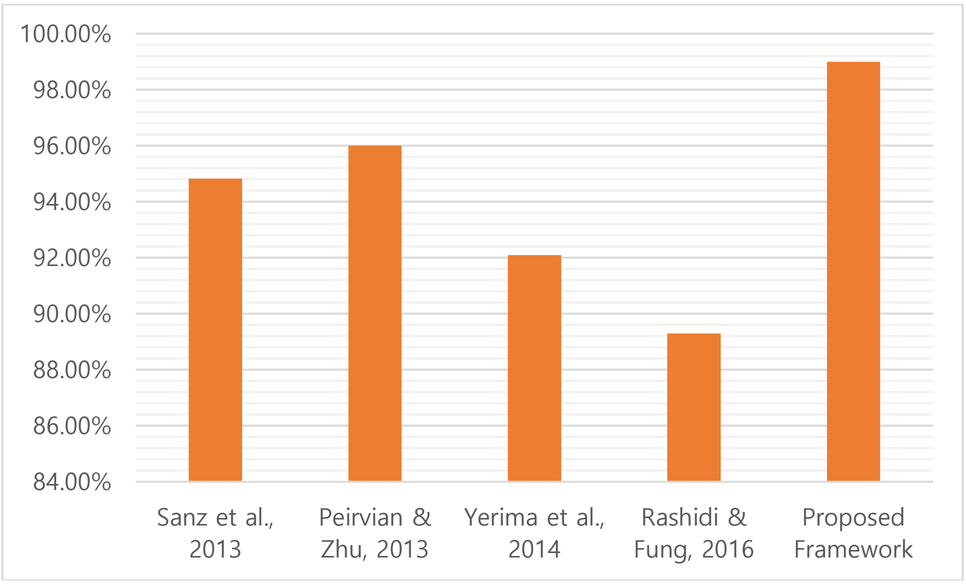

Fig. 6. Comparative Analysis

5. Conclusion

Botnets have become a big threat to smartphones due to the rapid evolution of mobile phone capabilities. Mobile phone is frequently connected to the Internet at all times to Android applications and online services such as entertainment, social media sites, web applications, and financial activities. Because of internet accessibility, botnet applications are growing more popular among cybercriminals. For this reason, we purpose a hybrid analysis framework to analyze and examine the Android botnet applications. The framework is divided into three parts. Dataset collection dataset preparation, and classifier evaluation.

In the first part, we prepare a dataset of benign, malware and botnet applications with a total of 100 instances in our work. For dataset preparation, we use reverse engineering techniques to extract features for the detection of Android botnet applications. Additionally, we designed a hybrid technique that extracts various features using static and dynamic applications to detect an Android botnet. During the feature extraction process, we have also discovered any permissions and API calls that potentially lead to botnet activity. In addition, applications are required to run in a secure Droid box, and the results are collected for further classification.
Finally, In the last phase, we compared the results by applying different machine learning classifiers on Android Botnet applications with other existing studies. The results show that the decision tree has the highest accuracy of 99% as compared to the random forest and Naivebayesian with 96% and 94% accuracy, respectively.

Conclusively, this paper can achieve the goal of providing the best mobile botnet detection and accuracy by merging a feature set of permissions and API Calls. Other key techniques to improve accuracy will be examined in the future, including intent, string, and system component selections.

References

[1] Poonguzhali, P., et al., "Secure storage of data on android based devices," *International Journal of Engineering and Technology*, 8(3), pp. 177-182, 2016. Article (CrossRef Link)

[2] Arif, M.N., A.A. Bakar, and M.M. Saudi, "A new mobile malware classification for audio exploitation," *International Journal of Engineering and Technology (UAE)*, 7(4.15), pp. 59–62, 2018. Article (CrossRef Link)

[3] Lindorfer, M., et al., "AndRadar: fast discovery of android applications in alternative markets," in *Proc. of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 51-71, 2014. Article (CrossRef Link)

[4] Erturk, E, "A case study in open source software security and privacy: Android adware," in *Proc. of World Congress on Internet Security (WorldCIS-2012)*, 2012. Article (CrossRef Link)

[5] Oberheide, J. and C. Miller, "Dissecting the android bouncer," *SummerCon2012*, New York, 95, pp. 110, 2012. Article (CrossRef Link)

[6] Percoco, N.J. and S. Schulte, "Adventures in bouncerland," *Black Hat USA*, 95, pp. 110, 2012.

[7] Pieterse, H. and M. Olivier, "Design of a hybrid command and control mobile botnet," *Journal of Information Warfare*, 12(1), pp. 70-82, 2013. Article (CrossRef Link)

[8] Geng, G., et al., "The Design of SMS Based Heterogeneous Mobile Botnet," *J. Comput.*, 7(1), pp. 235-243, 2012. Article (CrossRef Link)

[9] Schmidt, A.-D., et al, "Static analysis of executables for collaborative malware detection on android," in *Proc. of* 2009 *IEEE International Conference on Communications*, 2009.

[10] Karim, A., R. Salleh, and M.K. Khan, "SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications," *PloS One*, 11(3), pp. e0150077, 2016. Article (CrossRef Link)

[11] Aswini, A. and PP. Vinod, “Droid permission miner: Mining prominent permissions for Android malware analysis,” in *Proc. of The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 2014. Article (CrossRef Link)

[12] Choi, B., S.-K. Choi, and K. Cho, “Detection of mobile botnet using VPN,” in *Proc. of 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013. Article (CrossRef Link)

[13] Karim, A., R. Salleh, and S.A.A. Shah, “DeDroid: a mobile botnet detection approach based on static analysis,” in *Proc. of 2015 IEEE 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)*, 2015. Article (CrossRef Link)

[14] Liang, S. and X. Du, “Permission-combination-based scheme for android mobile malware detection” in *Proc. of 2014 IEEE international conference on communications (ICC)*, 2014. Article (CrossRef Link)

[15] Pravin, M.N.P., “Vetdroid: Analysis using permission for vetting undesirable behaviours in android applications,” *Int. J. Innov. Emerg. Res. Eng*, 2, pp. 131-136, 2015.

[16] Desnos, A., G. Gueguen, and S. Bachmann, “Androguard: Reverse engineering, malware and goodware analysis of android applications, and more (ninja!),". [Online]. Available: https://www.kitploit.com/2016/07/androguard-reverse-engineering-malware.html?m=0

[17] Su, M.-Y. and K.-T. Fung, “Detection of android malware by static analysis on permissions and sensitive functions” in *Proc. of 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016. Article (CrossRef Link)

[18] da Costa, V.G., et al, “Detecting mobile botnets through machine learning and system calls analysis,” in *Proc. of 2017 IEEE International Conference on Communications (ICC)*, 2017. Article (CrossRef Link)

[19] Yang, M., et al., “Detection of malicious behavior in android apps through API calls and permission uses analysis,” *Concurrency and Computation: Practice and Experience*, 29(19), pp. e4172, 2017. Article (CrossRef Link)

[20] Karim, A., Chang, V., & Firdaus, A., "Android botnets: a proof-of-concept using hybrid analysis approach," *Journal of Organizational and End User Computin*, vol. 32, no. 3, pp. 50-67, 2020. Article(CrossRef Link)

[21] Karim, A., “Detection of Mobile Botnet Applications Using Structural and Behavioral Patterns,” *Fakulti Sains Komputer dan Teknologi Maklumat, Universiti Malaya*, 2016.

[22] Desnos, A. and PP. Lantz, “Droidbox: An Android application sandbox for dynamic analysis,” Lund Univ., Lund, Sweden, Tech. Rep, 2011.

[23] Arp, D., et al, “Drebin: Effective and explainable detection of android malware in your pocket,” *Ndss*, 2014. Article (CrossRef Link)

[24] Karim, A., et al., “On the analysis and detection of mobile botnet applications,” *J. Univers. Comput. Sci.*, 22(4), pp. 567-588, 2016.

[25] Sanz, B., et al., “MAMA: manifest analysis for malware detection in android," *Cybernetics and Systems*, 44(6-7), pp. 469-488, 2013. Article (CrossRef Link)

[26] Yerima, S.Y., S. Sezer, and I. Muttik, "High accuracy android malware detection using ensemble learning," *IET Information Security*, 9(6), pp. 313-320, 2015. Article (CrossRef Link)

[27] Rashidi, B. and C. Fung, "BotTracer: Bot user detection using clustering method in RecDroid," in *Proc. of NOMS* 2016-2016 *IEEE/IFIP Network Operations and Management Symposium*, 2016. Article (CrossRef Link)

[28] Peiravian, N. and X. Zhu, “Machine learning for Android malware detection using permission and API calls,” in *Proc. of 2013 IEEE 25th international conference on Tools with artificial intelligence*, 2013. Article (CrossRef Link)

Mamoona Arshad has received his MS Degree (Information Technology) from Bahauddin Zakariya University Multan, Pakistan. Her area of research includes Botnet Detection, Machine learning, Deep Learning and Computer Vision. She is currently Visiting Lecturer in the Bahauddin Zakariya University Multan, Pakistan.

Ahmad Karim has received his Phd Degree (Computer Science) from University of Malaya, Malaysia. His area of research includes Botnet Detection, Mobile Cloud Computing, Computer Networks and Mobile Computing. He is currently Senior lecturer in the Bahauddin Zakariya University Multan, Pakistan. He has also achieved Cisco International Certifications (CCNA, CCNP, CCAI).