IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 April 2022, accepted 19 June 2022, date of publication 29 June 2022, date of current version 12 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3187094

Applications of Artificial Intelligence to Detect Android Botnets: A Survey

ABDULLAH M. ALMUHAIDEB[1] **AND DALAL Y. ALYNANBAAWI**[2]

[1]Saudi Aramco Cybersecurity Chair, Department of Networks and Communications, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

[2]Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

Corresponding author: Abdullah M. Almuhaideb (amAlmuhaideb@iau.edu.sa)

This work was supported by the Saudi Aramco Cybersecurity Chair, Imam Abdulrahman Bin Faisal University, Saudi Arabia.

ABSTRACT From the growing popularity of Android smart devices, and especially with the recent advances brought on by the COVID-19 pandemic on digital adoption and transformation, the importance of protecting these devices has grown, as they carry very sensitive data. Malicious attacks are targeting Android since it is open source and has the highest adoption rate among mobile platforms. Botnet attacks are one of the most often forgotten types of attacks. In addition, there is a lack of review papers that can clarify the state of knowledge and indicate research gaps in detecting android botnets. Therefore, in this paper, we conduct a literature review to highlight the contributions of several studies in the domain of Android Botnet detection. This study attempts to provide a comprehensive overview of the deployed AI apps for future academics interested in performing Android Botnet Detection studies. We focused on the applications of artificial intelligence and its two prominent subdomains, machine learning (ML) and deep learning (DL) techniques. The study presents available Android Botnet datasets suitable for detection using ML and DL algorithms. Moreover, this study provides an overview of the methodologies and tools utilized in APK analysis. The paper also serves as a comprehensive taxonomy of Android Botnet detection methods and highlights a number of challenges encountered while analyzing Android Botnet detection techniques. The research gaps indicated an absence of hybrid analysis research in the area, as well as a lack of an up-to-date dataset and a time-series dataset. The findings of this paper show valuable prospective directions for future research and development opportunities.

INDEX TERMS Android security, android attacks, android botnets, android botnet detection, artificial intelligence (AI), machine learning (ML), deep learning (DL).

I. INTRODUCTION

The industry of smartphones has changed the lives of people dramatically. Smartphones are no longer just for communication; they have evolved into one of life's necessities. Furthermore, as a consequence of the considerable changes in the nature of our daily vocations, schools, and routines induced by COVID19-related social distance laws and obligations, smart devices have become increasingly engaged in people's regular duties. The Android system is the largest mobile operating system market share worldwide, with a reported 72.21% of usage by July 2021 [1]. Because of this popularity, its open-source nature [2], and the ease with which new applications may be added to Google's Play Store [3] thanks to the well-documented guidance for building new apps offered by the Android official website [4], Android has more attack attempts than any other system.

Several reviews in the literature have been conducted on Android security challenges, attack types, and detection mechanisms. On the other hand, few countable reviews were performed for Android botnet detection. For example, [5] highlighted some of the most typical features of Android botnets. Another review of Android botnet attacks may be found at [6]. This paper analyzes Android botnet families in terms

The associate editor coordinating the review of this manuscript and approving it for publication was Sunil Karamchandani.

TABLE 1. Summary of surveys in Android malware detection methods.

Paper	Year	Key Contribution
[5]	2012	Identified detailed characteristics of Android Botnets
[7]	2015	Presented a taxonomy of mobile botnet attacks and architecture, and discussed detection approaches
[6]	2015	Provided a chronology of Android botnets' families with their characteristics, and discussed their attack strategies
[8]	2018	Categorized Android datasets and described: AndroZoo, Gnome Project, and AndroVault datasets
[9]	2018	Discussed the origin of common Android datasets with their drawbacks, and introduced their dataset CICAndMal2017

of their chronology, features, and attack strategies. Another Android botnet investigation is conducted in [7]. This work provides a taxonomy of botnet architecture, platform, target audience, vulnerabilities, and detection approaches. Despite being out of date (released between 2012 and 2015), these evaluations on Android botnets lack in-depth information and classification of related attacks, existing detection techniques, and available datasets.

In this review, we will give a broad overview of the literature on Android Botnet detection, including datasets, analysis methodologies, and detection techniques. We will also go through the Machine Learning (ML) and Deep Learning (DL) approaches used in Android Botnet detection, as well as the best classifiers that have been evaluated. Table 1 summarizes some of the surveys conducted on Android Botnet detection.

The essential goals and contributions of this work are to study the current state of Android Botnet detection methods and give a comprehensive view of ML and DL techniques in this area, which has been identified as a gap that needs to be addressed.

In summary, the following are the key contributions of this paper:

1. Display the available Android Botnet datasets appropriate for detection using ML and DL techniques.
2. Present overview of the methodologies and tools utilized in APK analysis.
3. Provide a comprehensive taxonomy of Android Botnet detection approaches.

The rest of this paper is organized as follows: Section II provides an overview of Android Botnet characteristics. Section III depicts Android Botnet analytical methods. Section IV reviews the detection techniques for Android Botnets in the literature. Section V analyzes the findings and provides an outline of the challenges that have been identified. Lastly, the paper is concluded in Section VI.

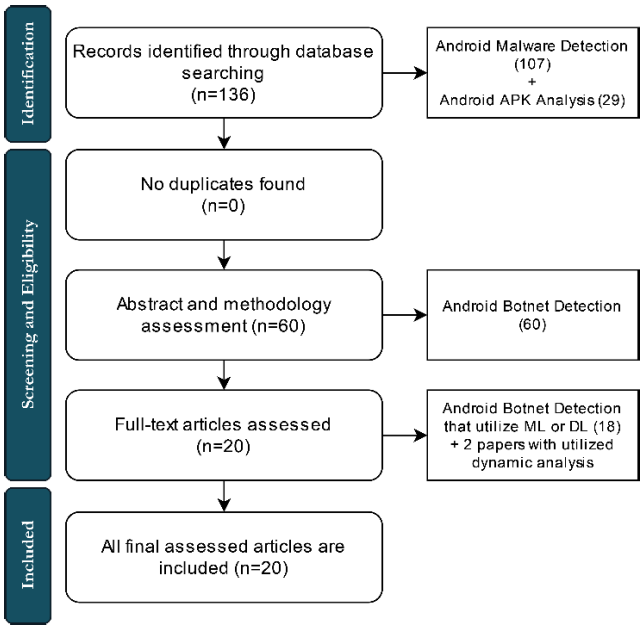

FIGURE 1. Methodology of the proposed review.

II. METHODOLOGY

This paper seeks to describe the various methodologies for detecting botnet attacks on Android smart devices that utilize ML and DL techniques. Hence, PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) technique was used to search for and choose papers that were relevant to the study's scope. Figure 1 illustrates the process followed in this paper.

The databases considered in the identification phase were chosen from the best-known publications and conferences, such as IEEE, ACM, and Elsevier. Only papers published (2015-2021) were selected. To find relevant articles, different keywords were utilized, including Android intrusion detection, Android malware detection, Android security, Android Botnets, and Deep Learning in Android Botnet Detection. Initially, 107 papers covering the topic of Android malware detection were discovered, while 29 articles covering Android APK analysis were found. There were no duplicate papers.

In terms of screening and eligibility, the abstract and methodology sections of these publications were assessed to choose studies focusing on Android botnet attacks. Following that, an evaluation of the full-text articles was performed to identify studies that used ML and DL approaches in the Android botnet detection process. In our study, two additional publications were included to address the dynamic analysis method in the analysis section. We addressed the current state of the art research on Android APK analysis methods and Android botnet detection strategies with a total of 20 publications, as result of our study.

III. ANDROID BOTNET

Botnets are networks of "malware-compromised machines" [10] which are host computers (or smart devices, our focus

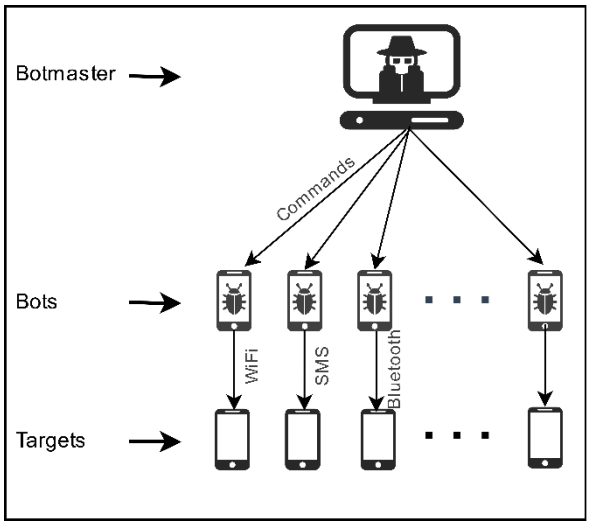

FIGURE 2. Overview of a botnet structure.

in this study) with unlimited geographical places, called bots, enslaved and controlled by one or several attackers, called botmasters, for future malicious actions. The infected devices might not act maliciously, but at the right time could be remotely activated by the botmaster to perform the desired goals, such as stealing information, financial charges, spreading viruses or warms, identity-related frauds and thefts, or performing other types of attacks such as Distributed Denial of Service (DDoS), Smurf attack, or spreading other malware [11]. Anserverbot, Beanbot, and Geinimi are three well-known Android botnet families.

Historically, the traditional channel for botnets was Internet Relay Chat (IRC), a text-based communication protocol that was used before in group discussion forums and one-to-one private communication and file sharing [10]. Nowadays, other types of protocols are used for bot spreading and communication, such as Peer-to-Peer (P2P) or HyperText Transfer Protocol(HTTP) protocols. Botnet attacks start with implanting malicious segments of code into the device by several means, which could be by Bluetooth communication, malicious email attachments, or an apparently good applications in the store to be downloaded. Figure 2 illustrates an overview of the Botnet structure.

IV. ANDROID BOTNET ANALYSIS

Several studies have been conducted to address the mobile botnet detection problem, proposing several techniques and mechanisms. In this section, we explore the journey of detecting Android Botnets from choosing a suitable dataset to validating whether or not an application is a botnet.

A. AVAILABLE DATASETS

Various studies employ ready-to-use datasets in their experiments. These datasets, such as the Drebin dataset [25], are primarily made up of extracted features in the form of zero-and-one vectors or numerical data. Other datasets are collections of applications in the form of APK files that the researchers use to conduct their analysis.

TABLE 2. Android Botnet datasets and studies used them.

Dataset	By	Number of Samples	Dataset Format	Botnet Studies
28-SABD 2019	[12]	112,485 malwares and 122,919 benign from ISCX	Vectors of 0s and 1s	[13] [14]
ISCX 2015	[15]	1929 malwares	APK files	[16] [17] [18] [19] [20] [21] [22] [23] [24]
Drebin 2014	[25]	5560 malwares and 123453 benign	Vectors of 0s and 1s	[26] [23]
Gnome/ MalGnome 2012	[27]	1260 Malwares 2010 - 2011	Vectors of 0s and 1s	[23] [28]
Others	-	-	-	[29] [30] [31] [32]

There are a variety of ready-to-use datasets to pick from, depending on the type of dataset required. Each of these datasets is described below, in the order of newest to oldest. The number of records in each dataset, as well as the research that used them, are displayed in Table 2.

1) ISCX

In the Android Botnet detection field, only one source dataset is made specifically for this type of malware, which is the ISCX dataset [15] (also known as the University of New Brunswick (UNB) dataset in certain research). This dataset contains APK files for studies that want to extract the features themselves.

2) 28-SABD

This is a new dataset specific to the Android botnet created by [12]. This dataset presents a new dataset based on the ISCX dataset that includes features derived via dynamic analysis in the form of vectors of zeros and ones.

3) DREBIN

This is one of the most used datasets in malware detection studies generated by [25]. Its features were extracted using static analysis on a total of 131,611 including malicious and benign applications collected from August 2010 to October 2012 [25].

4) GNOME / MALGNOME

This dataset is produced by [27], which contains 1260 malware samples categorized by malware family and gathered from August 2010 to October 2011.

5) OTHERS

All research that employed various datasets from different resources, gathered by the same study, falls under this

category in the following table. As a result, datasets tailored to Android Botnets are required. Furthermore, it has been noted that a time-series dataset is required for researchers who want to run experiments using DL classifiers that rely on time-series data.

For studies that rely on extracting features from the original APK files as part of their study, they must analyze the APK files themselves. Based on the methodology used and the tools used to extract the essential characteristics, these studies may be categorized into three parts: static, dynamic, and hybrid analysis. Due to the lack of Android Botnet detection based on the hybrid method, only the first two methods are discussed here.

B. STATIC ANALYSIS

Static analysis examines applications without running them, mainly by looking for harmful segments in the source code based on known malicious characteristics, which is similar to a signature-based method. In static analysis, we analyze the application and extract specific features that would help in building a detection model. Where in signature-based methods, segments of code are extracted from the examined application to be compared with the signature codes of known malware.

Several tools are essential to do static analysis on an Android application. ApkTool, which is designed to reverse engineer APK files, is the most prominent. APK files, which are analogous to EXE files on Windows computers, are Android Package files for executable apps. AAPT (Android Asset Packaging Tool) is a similar tool that can decode and extract data from AndroidManifest.xml without requiring the whole APK file to be decompiled [33]. ApkParser is a tool that converts the Manifest file to a text-readable format [34]. Baksmali and AXMLPrinter2 are both decompilers that extract readable bytcode from APK files [35]. AndroGuard, WALA, Soot, IACDroid, and Amandroid are among the frameworks for static analysis of Android apps available in the literature.

Several studies performed on Android Botnet detection followed the static analysis method. Both studies [23] and [26] employed ready-to-use datasets: Drebin and Gnome, respectively. These datasets are created by statically analyzing APK files. The work [23] looked at the link between permissions and used features and concluded that a certain combination of these two factors may be used to identify a botnet application. Then, as a feature selection approach, it used the Information Gain (IG) algorithm to ensure that the most impactful features were chosen.

For their own static analysis, all of the research [16], [17], [19]– [22], employed the ISCX dataset. Permissions and API Calls are the most commonly used elements in botnet detection techniques by the majority of them. Hijawi *et al.* [16] have used permissions and their corresponding protection levels as features for botnet detection using different ML classifiers. Each permission in Android systems has a protection level that ranges from normal to dangerous, as follows: if the degree of protection is normal, there is no need to ask the user to use the permission. This is something that can be used right away. Dangerous permissions, on the other hand, must be explicitly requested by the user.

Both [21] and [23] utilized the Information Gain (IG) feature selection algorithm, whereas [40] used WEKA's SubSetEval to pick the most correlated features since the features were represented using the word of bag approach.

The following is a summary of the most commonly used features in the static analysis method.

1) PERMISSIONS

Android is a permission-based access control system to control the actions that a process can do. This implies that every app that wants to execute a certain operation must first request permission from Android [36]. Permissions are extracted from the *AndroidManifest.xml* file.

2) API CALLS

The functions called by an application in its source code are referred to as API calls. They are evoked during runtime by an application, to request specific information or to perform a specific task.

3) BACKGROUND SERVICES

Android Background Services is a component of Android that operates in the background. It is often launched in an Android activity and runs in the same thread as the activity.

4) RECEIVERS

A broadcast receiver is an Android component that allows apps to send and receive events from the Android system or other apps. Its used to communicate between processes in an asynchronous manner.

5) INTENTS

Intents are used to notify the Android system of a certain event. Intents are used to communicate between different components of different applications. This feature is also used in dynamic analysis approaches.

Although static analysis saves time and effort, and it appears to be the most common analytical approach for Android Botnet detection studies, it is ineffective in identifying the characteristics of unknown or obfuscated malware.

C. DYNAMIC ANALYSIS

Dynamic analysis, on the other hand, examines an application's behavior throughout the execution time by analyzing the application's runtime, system calls, and performance. Dynamic analysis, unlike static analysis, is successful in identifying obfuscation and delivers relevant information in a short amount of time. However, dynamic analysis necessitates a large amount of computer power and may not be effective in detecting malicious activities in real-time.

Numerous tools are used to do dynamic analysis on an Android application. The adb tool (Android Debug Bridge)

TABLE 3. Summary of some of the existing Android analytic tools.

Static Analysis	Dynamic Analysis	Helping tools
ApkTool	adb tool	Genymotion Android Emulator
AAPT	Strace	BlueStack emulator
ApkParser	APIMonitor	Monkey tool
Baksmali	tcpdump/Wireshark	Appium tool
AXMLPrinter2	DroidBox	Kobiton
IACDroid	DynaLog	DriodBot
WALA	Robotium	Santoku Linux Distribution
Soot	CuckooDriod framework	
Amandroid		
AndroGuard		

is the most often used tool for running commands directly on Android-connected devices or emulators. For this reason, it is one of the most critical tools supplied with the Android SDK platform. Several tools, like Strace for collecting system calls [37], APIMonitor for monitoring an app's API calls [38], and tcpdump/Wireshark, a well-known packet analyzer [30], are used in conjunction with the adb tool to capture and record some of the app's features.

The malicious applications are executed on an Android device, either a real physical device or an emulator, to do the dynamic analysis. As a result, an emulator is a commonly used auxiliary tool in dynamic analysis. The native emulator of the Android SDK is the most often used. However, several other emulators, such as Genymotion [39] and BlueStack [40], are also utilized. The Monkey tool is an extremely valuable tool for dynamic analysis [41]. The Monkey tool is a generator that uses the stateless input generation approach to generate a pseudorandom sequence of events on an emulator or the real device to observe how the application behaves throughout various phases of regular smart device operation.

Several open-source frameworks/sandboxes are also used for dynamic analysis, including DroidBox, DynaLog, which is based on DroidBox and can automatically analyze hundreds of programs [42], Robotium, Café, which is based on Robotium and includes extra features, and CuckooDroid, a sandbox analyzer based on the Cuckoo platform. Unfortunately, the majority of these frameworks are no longer supported and hence cannot be utilized correctly.

The Santoku Linux environment, which is designed for providing tools and utilities for Android security analysis, is an underutilized capable framework for security analysis of Android platforms. Santoku contains the tools that DroidBox needs to work effectively, such as adb, logcat, Android emulator, images, etc. [42]. Table 3 summarizes some of the Android analytic tools that are currently available.

Android Botnet detection was carried out in a number of research based on dynamic analysis. For example in [28], the authors have employed the tPacketCapture pro tool to collect network visitors of software from the Gnome dataset. Another study by [12] analyzed the network traffic from botnet apps derived from the ISCX dataset using four actual machines and a BlueStack emulator. Network traffic has been shown to be the most trustworthy characteristic according to several studies such as [18] and [29].

1) NETWORK TRAFFIC

In network traffic, the analyzer would keep track of various aspects of the data transmission and reception process. These aspects might include the amount of data sent, received, destination IP address, or any other network-related information.

In general, dynamics-based analysis requires high computational power and may not be efficient in detecting malicious behavior in real-time.

D. DISCUSSION

The thirst method in the malware detection process, in general, is a hybrid analysis. A hybrid method is used to evaluate system behavior by employing both static and dynamic analysis in parallel. It is used to benefit from both static and dynamic analysis advantages while overcoming each one's drawbacks.

To the best of our knowledge, there has never been any research done on botnet detection using a hybrid analysis method. As a result, this has been identified as a knowledge gap in the literature.

Figure 3 shows a taxonomy of the discussed analytical methods. Presenting this taxonomy would aid in categorizing studies based on the type of analysis method adopted, directing researchers to papers that cover certain analysis methods or utilize specific datasets, and identifying the need for greater scientific research on overlooked methodologies and features.

V. ANDROID BOTNET DETECTION

Although various attempts have offered an overview of using ML techniques in botnet detection in general [43], few research leveraging AI applications on botnet detection in Android systems have been undertaken. Most of which were good attempts to generate labelled datasets as botnet samples. Although [12] has used ML-based classifiers to label the data for generating a ready-to-use Botnet dataset, the study [20] has applied ML using the WEKA platform to generate its own dataset to perform the detection. WEKA has been utilized in other several existing works, such as [19], [29], and [44], which is a basic tool to use in terms of implementing ML techniques in detecting malware, while cloud-based botnet detection techniques were adopted by other studies such as [30] and [44].

A. ML-BASED DETECTION

Most Android botnet detection studies utilized the static analysis method for feature extracting. For example, [17] utilized

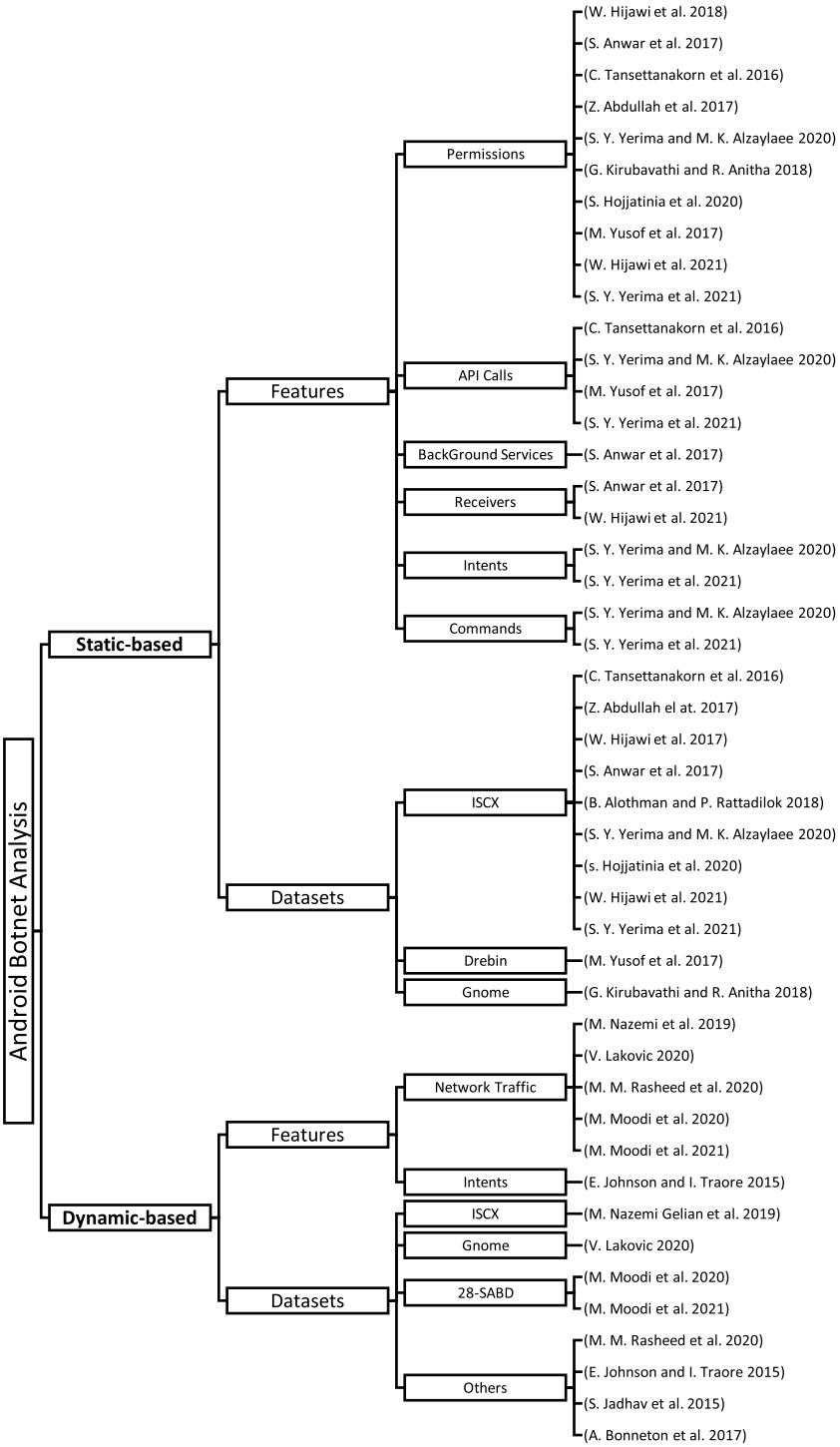

FIGURE 3. Taxonomy of features and datasets used for android botnet analysis in the literature.

four filters to classify the collected features statically: MD5, which is a known code of popular malicious applications, basically is used as a message integrity technique [45], Permissions, Broadcast Receiver, and Background Service. The study used SVM, KNN, J48, Bagging, NB, and RF classifiers with different features after the application was filtered out of these four layers. On Permission, NB produced the best results among the other classifiers. Both [21] and [23] employed the Information Gain (IG) feature selection algorithm on statically extracted features, whereas [13] used and emphasized the merits of the Fuzzy SAPSO selection algorithm on dynamically generated features.

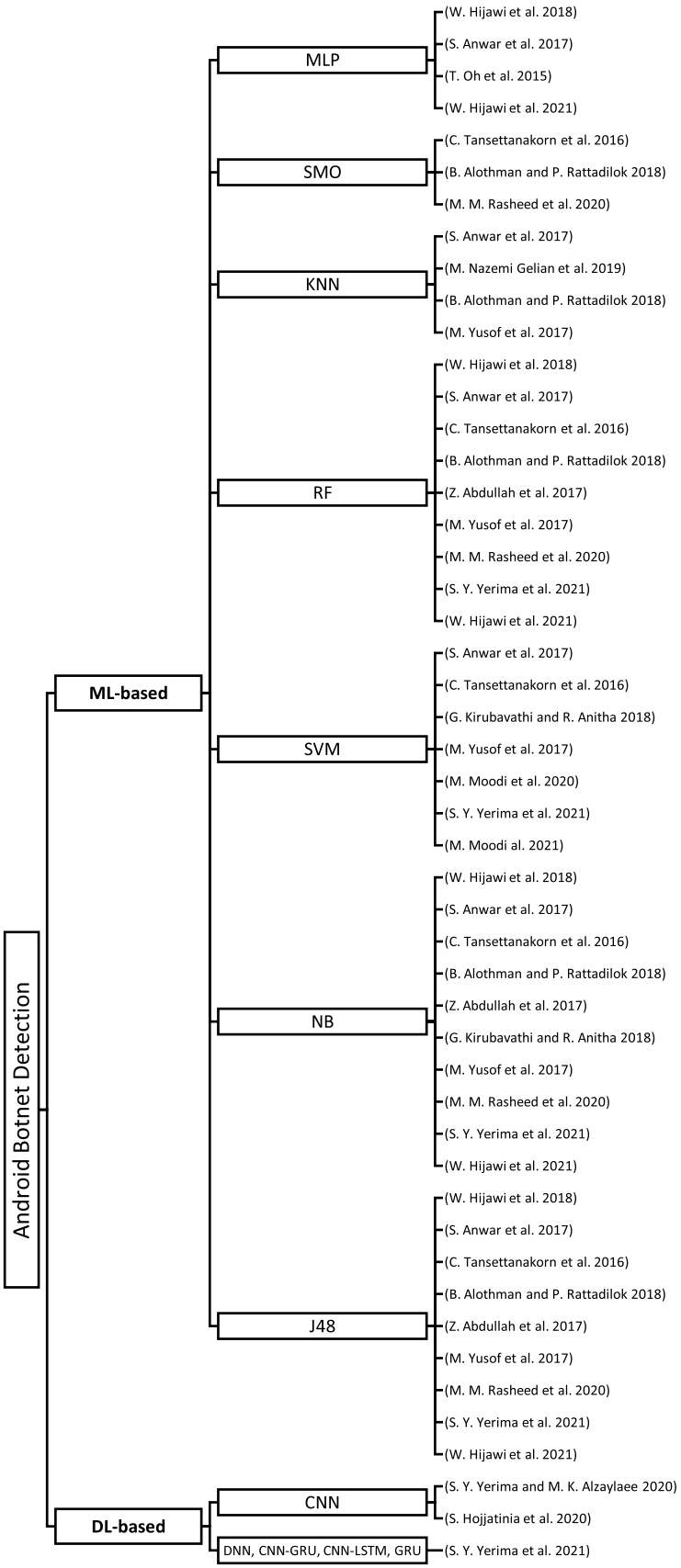

FIGURE 4. Android botnet detection techniques and classifiers applied in the literature.

Another work that employed a feature selection technique is [20]. The authors utilized text mining on the Java source code of the apps in this study, and in the first approach, they used Natural Language Processing (NLP) technique with varied quantities of words as the adjusted parameter. In the second approach, they utilized the CodeAnalyzer tool to extract certain quantitative features. As a consequence, three datasets have been generated. Finally, they used WEKA's SubSetEval feature selection method on these datasets to reduce the number of features and pick the best features for classification to prevent overfitted models. On these diverse datasets, they tested Naïve Bayes (NB), Decision Tree (DT, also known by J48), K-Nearest Neighbor (KNN), Random Forest (RF), and Sequential Minimal Optimization (SMO), and found that the last three performed better than the others.

The features selected by [26], on the other hand, were chosen after reviewing other research and resulted in 16 permissions and 31 API Calls being the most significant characteristics of Android botnet applications. This work also implemented NB, KNN, J48, RF, and SVM classifiers in their detection experiments, with RF achieving the highest accuracy rate of 99.4%.

The primary detecting feature of [18], [28], [29], [44], and [13] was Network Traffic. The researchers in [18] selected the top four features based on the [46] study and then added seven additional features based on the accuracy of their own assessment. They began the detection with a small set of labeled data and then used the incoming Network Traffic to train the KNN classifier incrementally. By using self-learning classification, they were able to obtain a detection rate of 90.3%.

B. DL-BASED DETECTION

Despite the fact that DL is a subset of classical ML, DL classifiers have recently caught the interest of academics. Because it teaches itself to process and learn from the data by optimizing the parameters, DL beats traditional ML.

Only [22] and [24] adopted DL classifiers in the identification of Android botnets using features retrieved using a static-based analysis method. Both used the CNN classification technique, and both achieved high detection accuracy, with 98.9 and 97.2 percent, respectively.

Figure 4 shows a taxonomy of the discussed detection techniques with corresponding classifiers applied in the literature. According to this taxonomy, only three papers have investigated the potential of DL techniques, indicating that DL approaches are currently fairly unexplored.

VI. DISCUSSION AND OPEN CHALLENGES

In this section, we will go through the findings of the presented survey on Android botnet detection utilizing AI applications, which may be summarized as follows:

- Methods of analysis used in prior research on Android APK files.
- Tools and features associated with the analysis method used.

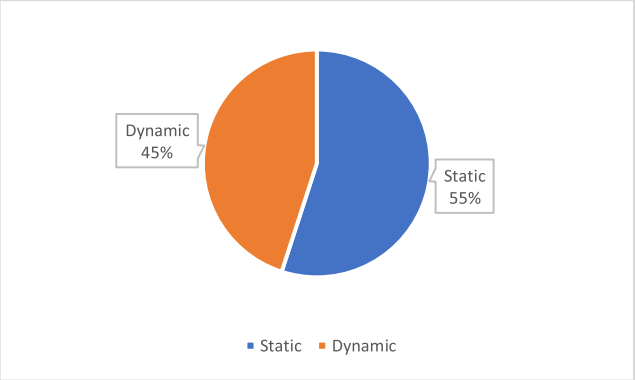

FIGURE 5. Analysis methods used in android botnet detection.

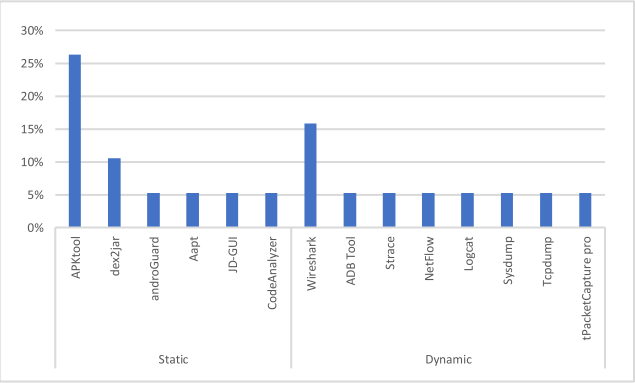

FIGURE 6. Tools used in botnet detection with each analysis method.

- Datasets for Android botnet applications that are currently available.
- Current AI applications, including its primary subfields ML and DL, have been covered in the literature to detect Android botnets.

By doing this evaluation, we discover that the majority of studies have utilized the static analysis method in Android botnet detection, as seen in Figure 5. We can also observe that the Android researchers have not yet investigated how hybrid analysis could be used to detect botnets. Hybrid analysis, as previously indicated, may be more successful in extracting more relevant characteristics of a certain type of malware.

Each analysis method has its own set of tools and processes, some of which are selected by researchers because they are straightforward to use, while others are not employed because they lack adequate support. Figure 6 provides the analysis tools that were utilized in the literature for each of the analytical methods. APKtool, which is the most popular tool for reverse engineering APK files, is one of the most powerful and commonly chosen in static-based and signature-based analysis, as well as being known for its ease of use.

In the case of dynamic and hybrid analysis, which need the observation of tested applications during the execution of the app, it is observed that there is a shortage of acceptable tools, whether due to a lack of adequate support, a lack of

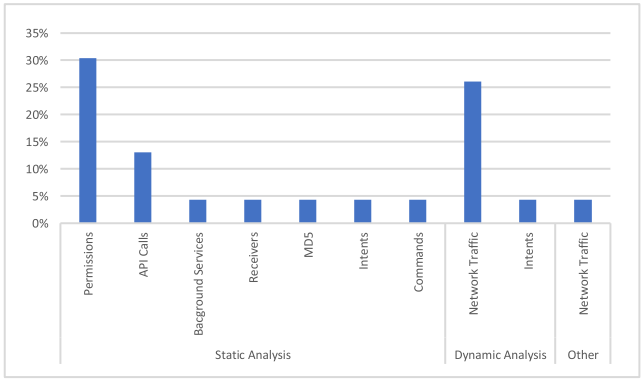

FIGURE 7. Features used in botnet detection with each analysis method.

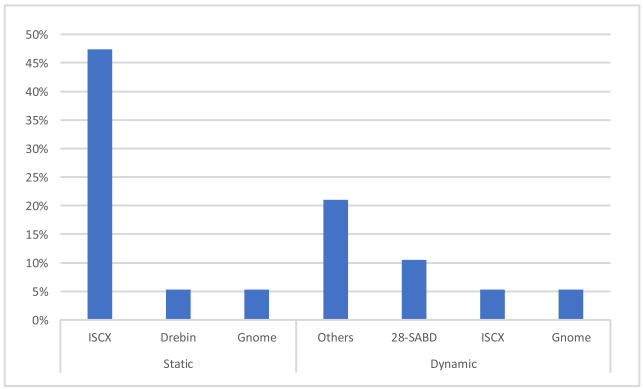

FIGURE 8. Datasets used in botnet detection in each analytical method.

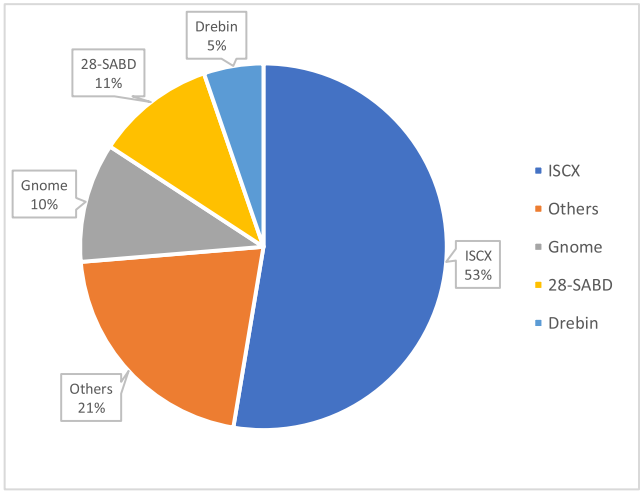

FIGURE 9. Overall usage of android botnet datasets.

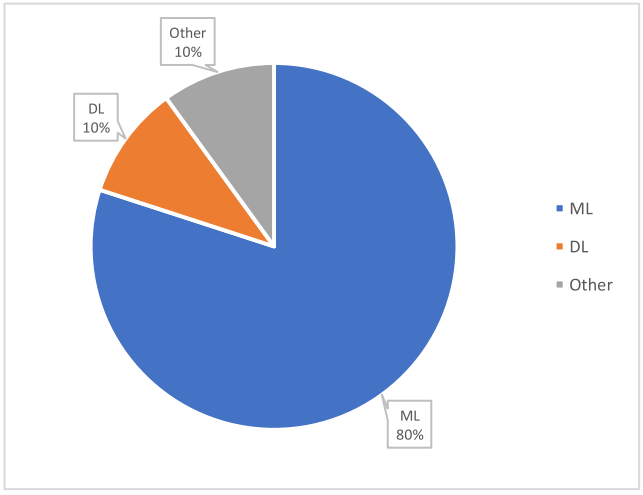

FIGURE 10. Android botnet detection techniques in the literature.

appropriate frameworks to use, or difficulties and challenges in using them as stated previously in Table 3.

Besides that, the features that are employed are influenced by the sort of analysis that is conducted. Figure 7 illustrates how these features are considered according to the analytical method employed. Permissions are the most commonly used feature in static-based analysis, whereas Network Traffic is the most prominent feature in dynamics-based analysis, as shown in the diagram. Intents are employed in both types of analyses, and their impact on detection accuracy requires additional research. Nevertheless, as previously mentioned, only a few researchers have used feature selection algorithms in their investigations.

In terms of datasets, as previously stated, just one dataset, the ISCX dataset, is created exclusively for Android Botnet apps. The 28-SABD dataset is the second accessible dataset, which is a vector of zeros and ones produced from the aforementioned ISCX dataset. Figure 8 demonstrates the various usage of these datasets in each analysis method, whereas Figure 9 depicts the ISCX dataset's dominance in the majority of Android Botnet studies.

Furthermore, as demonstrated in Figure 10, ML approaches are increasingly used in botnet detection. It is noted that the researchers used a variety of classifiers, with some classifiers, such as NB, J48, RF, and SVM, being preferred by the researchers (demonstrated in Figure 11). The most examined classifier in Android Botnet Detection is NB as shown in Figure 12. RF, on the other hand, has the best performance among the other classifiers, with an accuracy of 99.4 percent (see Tables 4 and 5). Only one research looked at the potential of using a time-series-based detection method. This indicates that we should pay more attention to experiments and the need for time-series datasets, which are not currently available in the literature. In terms of accuracy, Figure 13 presents a comparison of the strongest classifiers employed in the reviewed studies.

Its worth noting that DL classifiers have only been explored in three recent studies. However, it should be emphasized that these employed deep learning classifiers, CNN (Convolutional Neural Networks) and DNN (Deep Neural Networks), have achieved extremely high accuracy rates, ranging from 97.2 to 99.1 percent respectively (see Table 3 and Table 4). Other DL classifiers are proving their

TABLE 4. Comparison of android botnet detection studies using static analysis method.

Paper	Year	Dataset	Features	Feature Selection	Detection Technique	Used Classifiers	Best Classifier	Specificity	Recall	F-score	Precision	Acc
[19]	2016	ISCX	Permissions 63 and API calls 1414	-	ML	SVM, SMO, Bagging, NB, J48, MLP, RF	RF	-	96.9	-	97.2	-
[21]	2017		Requested Permissions 138	IG		NB, RF, J48		-	94.6	-	93.1	-
[16]	2017		Requested Permissions 145	-		RF, MLP NN, J48, NB		-	-	-	-	96.8
[17]	2017		Permissions, Receivers, BG services	-		SVM, KNN, J48, NB, RF	NB	-	93.2	-	97	95.1
[20]	2018		Most informative features using text mining WEKA	WEKA's SubSetEval		NB, KNN, J48, RF, SMO	KNN, RF, SMO	-	-	-	-	95.24
[26]	2017	Drebin	Permissions 16 API calls 31	-		NB, KNN, J48, RF, SVM	RF	-	99.4	-	93.2	99.4
[23]	2018	Gnome, Drebin, DroidAnalytics, and ISCX	Requested Permissions, Used Features 25	IG		SVM, NB, REPTree	NB	96.42	-	96.9	96.42	96.53
[47]	2021	ISCX	Permissions, Protection Level, SW/HW Access, SW/HD Access Category, URL, Receivers	CARET Package		J48, RF, NB, MLP	RF	-	98	98	99	97.7
[48]	2021		Permissions 130, API calls 135, Commands 19, Extra Files 5, and Intents 53	-		SVM, RF, SL, j48, NB, BN	SVM		97.3	97.6	98	98.7
				-	DL	DNN, CNN-GRU, CNN-LSTM, GRU	DNN	-	97.9	98.4	99	99.1
[22]	2020			-		CNN	CNN	-	97.8	98.1	98.3	98.9
[24]	2020		Permissions	-				-	96	95.7	95.5	97.2

TABLE 5. Comparison of android botnet detection studies using dynamic analysis method.

Paper	Year	Dataset	Features	Feature Selection	Detection Technique	Used Classifiers	Best Classifier	Specificity	Recall	F-score	Precision	Acc
[18]	2019	ISCX	Network Traffic 10 after feature selection	-	ML	KNN	KNN	-	90.3	-	-	-
[28]	2020	Gnome	Network Traffic	-		adaptive neuro-fuzzy inference system (ANFIS)		-	-	-	-	-
[13]	2020	28-SABD	Network Traffic 75	fuzzy SAPSO		SVM technique using SAPSO	SVM	95.32	96.49	-	97.81	98.33
[14]	2021			SSLPSO		SVM technique using SSLPSO		95.53	96.76	-	97.74	98.28
[29]	2020	Beanbot, Biige, Fakeinst, FakeMart, FakeNotify, Jifake, Mazarbot, Nandrobox, Plankton, and SMSsniffer	Network Traffic 75	-		NB, J48, SMO, RT, LMT	J48	95.35	-	-	-	-
[44]	2015	-	Network Traffic and System Calls	-		MLP	-	-	-	-	-	-
[32]	2017	Traffic captured from DNS server of Large ISP	DNS Temporal Pattern	-		time-series (A CART) DT		-	85.82	85.5	-	92.2
[31]	2015	collected data from 7 mobile phones for 3 weeks monitoring	Intent logging (Time Difference between sent and received SMS intent)	-	Others	An algorithm computes the delay time between sent and received SMS		-	95.59	-	-	-
[30]	2015	-	Network Traffic	-		A two-layer algorithm, using a plotted graph		-	-	-	-	-

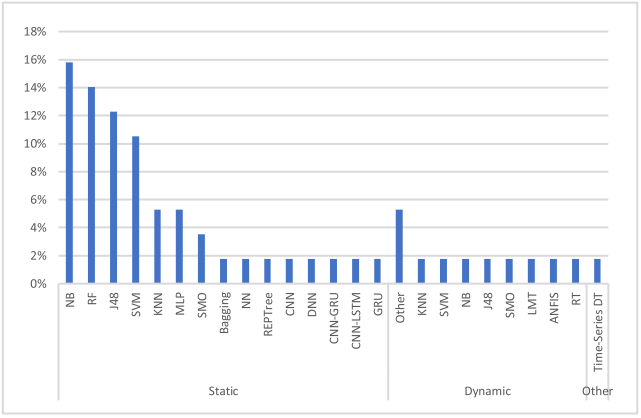

FIGURE 11. Classifiers for android botnet detection in each analytical method.

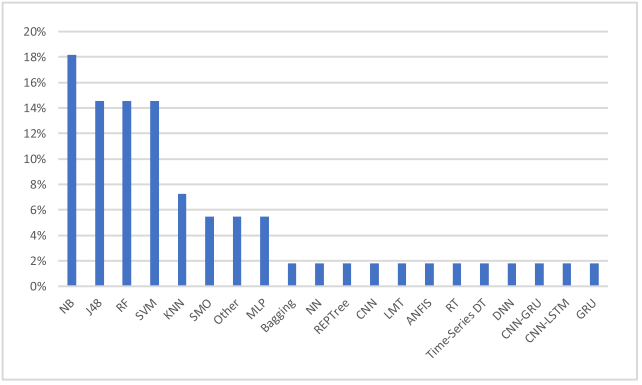

FIGURE 12. Top used classifiers in android botnet detection.

power to identify malware, and their promise in this field simply has to be explored.

Table 4 shows a comparison of research on Android Botnet Detection that used a static analytical method, whereas Table 5 shows a comparison of studies on Android Botnet Detection that used a dynamic analytical method.

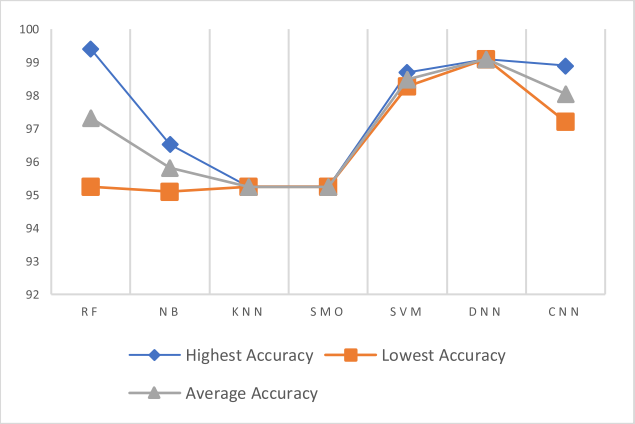

FIGURE 13. Comparison of the best classifiers discussed with their highest, lowest, and average accuracy.

VII. CONCLUSION / AND FUTURE WORK

Security flaws may affect any smartphone, although attackers are particularly interested in Android devices. This is related to various factors that have already been mentioned in the paper. This study aimed to offer a thorough overview of the implemented AI applications for future academics interested in conducting experiments on Android Botnet Detection. AI applications, including its subdomains ML and DL, have demonstrated their capacity to identify many types of malware. Botnets on Android are a relatively old subject with few investigations. In this assessment, 20 papers in this field from 2015 to the present were gathered and examined. The main analytical approaches for extracting features were addressed, including static, dynamic, and hybrid analyses. It also mentioned the lack of hybrid analysis research in the area, as well as the lack of an up-to-date dataset and a time-series dataset. To serve as a reference for future studies, a taxonomy of features and datasets utilized in the literature was provided. In static-based analysis, permissions was the most commonly used feature, whereas, in dynamics-based analysis, Network Traffic was the most commonly used feature. Another taxonomy was presented to distinguish between studies that utilized NB, J48, RF, SVM, KNN, SMO, Bagging, NN, and DT classifiers, as well as DL classifiers. Detailed tables were produced to provide a picture of the current work. The best classifier among all ML classifiers is RF, while DNN is the best one among DL classifiers in this subject, according to these tables.

For future work, we intend to perform a hybrid analysis on Android APK files, extract a time-series dataset, and then use DL-based classification to detect Android botnets, which has been recognized as a research need. Furthermore, an in-depth examination of various Android vulnerabilities and attacks such as SMS, Email, Spying, Application Sandboxing, and Rooting attacks will be carried out.

REFERENCES

[1] StatCounter. (2020). *Mobile Operating System Market Share Worldwide | StatCounter Global Stats*. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide

[2] A. Ayyasamy, "Survey on Android application advancement and security," in *Proc. 7th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2015, pp. 1–4.

[3] A. S. Yuksel, A. H. Zaim, and M. A. Aydin, "A comprehensive analysis of Android security and proposed solutions," *Int. J. Comput. Netw. Inf. Secur.*, vol. 6, no. 12, pp. 9–20, Nov. 2014.

[4] L. Nishani, "Review on security threats for mobile devices and significant countermeasures on securing Android mobiles," in *Automated Enterprise Systems for Maximizing Business Performance*. Hershey, PA, USA: IGI Global, 2015, ch. 1, pp. 1–18. [Online]. Available: https://www.igi-global.com/book/automated-enterprise-systems-maximizing-business/129616

[5] H. Pieterse and M. S. Olivier, "Android botnets on the rise: Trends and characteristics," in *Proc. Inf. Secur. South Afr.*, Aug. 2012, pp. 1–5.

[6] H. Pieterse and I. Burke, "Evolution study of Android botnets," in *Proc. 10th Int. Conf. Cyber Warfare Secur. (ICCWS)*, 2015, pp. 232–240.

[7] A. Karim, S. A. A. Shah, R. B. Salleh, M. Arif, R. M. Noor, and S. Shamshirband, "Mobile botnet attacks—An emerging threat: Classification, review and open issues," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 4, pp. 1471–1492, 2015.

[8] F.-X. Geiger and I. Malavolta, "Datasets of Android applications: A literature review," 2018, *arXiv:1809.10069*.

[9] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark Android malware datasets and classification," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2018, pp. 1–7.

[10] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Comput. Netw.*, vol. 57, no. 2, pp. 378–403, 2013.

[11] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, "Botnet: Classification, attacks, detection, tracing, and preventive measures," *EURASIP J. Wireless Commun. Netw.*, vol. 2009, no. 1, pp. 1–11, Dec. 2009.

[12] M. Moodi and M. Ghazvini, "A new method for assigning appropriate labels to create a 28 standard Android botnet dataset (28-SABD)," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 11, pp. 4579–4593, Nov. 2019.

[13] M. Moodi, M. Ghazvini, H. Moodi, and B. Ghavami, "A smart adaptive particle swarm optimization–support vector machine: Android botnet detection application," *J. Supercomput.*, vol. 76, no. 12, pp. 9854–9881, Dec. 2020.

[14] M. Moodi, M. Ghazvini, and H. Moodi, "A hybrid intelligent approach to detect Android botnet using smart self-adaptive learning-based PSO-SVM," *Knowl.-Based Syst.*, vol. 222, Jun. 2021, Art. no. 106988.

[15] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "Android botnets: What URLs are telling us," in *Proc. Int. Conf. Netw. Syst. Secur.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 9408, 2015, pp. 78–91.

[16] W. Hijawi, J. Alqatawna, and H. Faris, "Toward a detection framework for Android botnet," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 197–202.

[17] S. Anwar, J. M. Zain, Z. Inayat, R. U. Haq, A. Karim, and A. N. Jabir, "A static approach towards mobile botnet detection," in *Proc. 3rd Int. Conf. Electron. Design (ICED)*, Aug. 2016, pp. 563–567.

[18] M. N. Gelian, H. Mashayekhi, and Y. Mashayekhi, "A self-learning stream classifier for flow-based botnet detection," *Int. J. Commun. Syst.*, vol. 32, no. 16, pp. 1–15, 2019.

[19] C. Tansettanakorn, S. Thongprasit, S. Thamkongka, and V. Visoottiviseth, "ABIS: A prototype of Android botnet identification system," in *Proc. 5th ICT Int. Student Project Conf. (ICT-ISPC)*, May 2016, pp. 1–5.

[20] B. Alothman and P. Rattadilok, "Android botnet detection: An integrated source code mining approach," in *Proc. 12th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 111–115.

[21] Z. Abdullah, M. M. Saudi, and N. B. Anuar, "ABC: Android botnet classification using feature selection and classification algorithms," *Adv. Sci. Lett.*, vol. 23, no. 5, pp. 4717–4720, May 2017.

[22] S. Y. Yerima and M. K. Alzaylaee, "Mobile botnet detection: A deep learning approach using convolutional neural networks," in *Proc. Int. Conf. Cyber Situational Awareness, Data Anal. Assessment (CyberSA)*, Jun. 2020, pp. 15–19.

[23] G. Kirubavathi and R. Anitha, "Structural analysis and detection of Android botnets using machine learning techniques," *Int. J. Inf. Secur.*, vol. 17, no. 2, pp. 153–167, Apr. 2018.

[24] S. Hojjatinia, S. Hamzenejadi, and H. Mohseni, "Android botnet detection using convolutional neural networks," in *Proc. 28th Iranian Conf. Electr. Eng. (ICEE)*, Aug. 2020, pp. 1–6.

[25] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "DREBIN: Effective and explainable detection of Android malware in your pocket," in *Proc. NDSS*, 2014, vol. 14, no. 1, pp. 23–26.
[26] M. Yusof, M. M. Saudi, and F. Ridzuan, "A new mobile botnet classification based on permission and API calls," in *Proc. 7th Int. Conf. Emerg. Secur. Technol. (EST)*, Sep. 2017, pp. 122–127.
[27] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 95–109.
[28] V. Lakovic, "Crisis management of Android botnet detection using adaptive neuro-fuzzy inference system," *Ann. Data Sci.*, vol. 7, no. 2, pp. 347–355, Jun. 2020.
[29] M. M. Rasheed, A. K. Faieq, and A. A. Hashim, "Android botnet detection using machine learning," *Ingénierie des systèmes d Inf.*, vol. 25, no. 1, pp. 127–130, Feb. 2020.
[30] S. Jadhav, S. Dutia, K. Calangutkar, T. Oh, Y. H. Kim, and J. N. Kim, "Cloud-based Android botnet malware detection system," in *Proc. 17th Int. Conf. Adv. Commun. Technol. (ICACT)*, Jul. 2015, pp. 347–352.
[31] E. Johnson and I. Traore, "SMS botnet detection for Android devices through intent capture and modeling," in *Proc. IEEE 34th Symp. Reliable Distrib. Syst. Workshop (SRDSW)*, Sep. 2015, pp. 36–41.
[32] A. Bonneton, D. Migault, S. Senecal, and N. Kheir, "DGA bot detection with time series decision trees," in *Proc. 4th Int. Workshop Building Anal. Datasets Gathering Exper. Returns Secur. (BADGERS)*, Jan. 2016, pp. 42–53.
[33] R. Schmicker, F. Breitinger, and I. Baggili, "AndroParse—An Android feature extraction framework and dataset," in *Proc. Int. Conf. Digit. Forensics Cyber Crime*, 2018, pp. 66–88.
[34] V. P. Dharmalingam and V. Palanisamy, "A novel permission ranking system for Android malware detection—The permission grader," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 5071–5081, May 2022.
[35] J. Tang, R. Li, K. Wang, X. Gu, and Z. Xu, "A novel hybrid method to analyze security vulnerabilities in Android applications," *Tsinghua Sci. Technol.*, vol. 25, no. 5, pp. 589–603, Oct. 2020.
[36] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif, "A multimodal malware detection technique for Android IoT devices using various features," *IEEE Access*, vol. 7, pp. 64411–64430, 2019.
[37] P. K. Das, A. Joshi, and T. Finin, "App behavioral analysis using system calls," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 487–492.
[38] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Improving dynamic analysis of Android apps using hybrid test input generation," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services (Cyber Secur.)*, Jun. 2017, pp. 1–8.
[39] D. Chaulagain, P. Poudel, P. Pathak, S. Roy, D. Caragea, G. Liu, and X. Ou, "Hybrid analysis of Android apps for security vetting using deep learning," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Jun. 2020, pp. 1–9.
[40] M. Borek, G. Creech, and U. Canberra, "Intrusion detection system for Android: Linux kernel system calls analysis," School Sci. Secur. Mobile Comput., Aalto Univ., Espoo, Finland, 2017.
[41] J. Milosevic, M. Malek, and A. Ferrante, "Time, accuracy and power consumption tradeoff in mobile malware detection systems," *Comput. Secur.*, vol. 82, pp. 314–328, May 2019.
[42] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DynaLog: An automated dynamic analysis framework for characterizing Android applications," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services (Cyber Secur.)*, Jun. 2016, pp. 1–8.
[43] S. Miller and C. Busby-Earle, "The role of machine learning in botnet detection," in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 359–364.
[44] T. Oh, S. Jadhav, and Y. H. Kim, "Android botnet categorization and family detection based on behavioural and signature data," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2015, pp. 647–652.
[45] 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. 1–36, 2016.
[46] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.
[47] W. Hijawi, J. Alqatawna, A. M. Al-Zoubi, M. A. Hassonah, and H. Faris, "Android botnet detection using machine learning models based on a comprehensive static analysis approach," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102735.
[48] S. Y. Yerima, M. K. Alzaylaee, A. Shajan, and P. Vinod, "Deep learning techniques for Android botnet detection," *Electronics*, vol. 10, no. 4, pp. 1–17, 2021.

ABDULLAH M. ALMUHAIDEB received the B.S. degree (Hons.) in computer information system from King Faisal University, Saudi Arabia, in 2003, and the M.S. (Hons.) and Ph.D. degrees in network security from Monash University, Melbourne, Australia, in 2007 and 2013, respectively. He is currently an Associate Professor in information security, a Supervisor with the Saudi Aramco Cybersecurity Chair, and the Dean of the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Saudi Arabia. He has published two patents and more than 45 scientific articles in journals and premier ACM/IEEE/Springer conferences. His research interests include mobile security, authentication and identification, and ubiquitous wireless access.

DALAL Y. ALYNANBAAWI received the B.S. degree (Hons.) in computer science from King Abdulaziz University, Saudi Arabia, in 2004. She is currently pursuing the M.S. degree with Imam Abdulrahman Bin Faisal University, Saudi Arabia. Her research interests include information security, machine learning, and deep learning techniques.