Survey for Detection and Analysis of Android Malware(s) Through Artificial Intelligence Techniques

Sandeep Sharma, Kavita Khanna, and Prachi Ahlawat

Abstract Artificial intelligence techniques have been intensively used for android malware detection and analysis in the last past few years. The proposed methodologies do not suffice the requirement while characteristics of malwares are changing so rapidly and evolving new complex malwares. Therefore, it is a very complex task to classify and identify these malwares. This paper presents an organized and comprehensive survey for the detection techniques of android malware(s) in chronological order. These detection and analysis techniques are elaborated in two core categories: statics and dynamic analysis and hybrid analysis with machine learning or artificial intelligence. The core contributions of this paper are: (1) explaining a methodical, chronicle and organized summary of the existing techniques of android malware detection, (2) exploring the major elements and challenges in the detection methods and (3) explaining the importance of artificial intelligence for android malware detection. The detection approaches are explained in a manner that new approaches are compared with the old ones based on their features. The advantages and disadvantages of each approach are discussed. This study facilitates researchers and academics to have a wide-ranging conception of the field of android malware detection and provides a platform to enhance the fundamental knowledge to implement the new idea and subsequent improvement further in existing techniques.

Keywords Android · Malware detection · Static · Dynamic · Hybrid · Artificial intelligence

S. Sharma (✉) · K. Khanna · P. Ahlawat
The Northcap University, Gurugram, Haryana, India
e-mail: kavitakhanna@ncuindia.edu

P. Ahlawat
e-mail: prachi@ncuindia.edu

K. Khanna et al. (eds.), *Cyber Security and Digital Forensics*,
Lecture Notes on Data Engineering and Communications Technologies 73,
https://doi.org/10.1007/978-981-16-3961-6_28

1 Introduction

The smart mobile phones are attracting all due to their tremendous features and have provided online banking, online marketing, online study, etc., in addition to fundamental telephony services. These mobile terminals are outfitted with huge processing power and ample storage that holds a variety of sensitive and critical data like contacts, pictures, passwords, cookies, credit card details, location information, etc. As per statista's report, the global figure of subscribers of smart mobile phone has reached 4.01 billion in 2020 and is expected to reach 4.7 billion in 2022 [1]. However, this popularity of smartphones has attracted the attention of hackers who may steal the data stored in smartphones and further may deduce the code to unlock the mobile to compromise the device or manipulate popular services. The various Android Operating Systems (OS) is the most accepted operating systems for the existing smartphones (82.8%) [2]. On one hand, the Android OS is an open-source software system to provide a wide range of accessibility to users and on the other hand, Android OS is more susceptible to cyber-attacks. The malicious apps or malwares are developing so rapidly worldwide that was never observed in the past. Normally, android apps and games are downloaded and installed from the Play store verified by play protect developed by Google. However, android apps and games can also be installed from other non-authenticated and not verified stores where apps and games are integrated with malicious codes that contaminate the android phones and exfiltrate the critical data to hackers. The protected Play store is also not secured from the hackers even after their continuous effort to monitor and remove infected applications. Moreover, several variants of Android Operating System are available in multiple manufacturer's devices and all variants are needed patches and updates regularly from the respective developer immediately. There are some major aspects like diverse ARM processors, limited RAM, and power battery restriction, etc. that create the complexity in detection of malicious codes. TrendMicro declared in the first quarter of 2020 that, mobile cyber espionage operations have increased by 1400%, distributed among multiple firmware(s) and operating systems from 2015 to 2019 [3].

Several anti-malware techniques, frameworks and solutions have been developed to protect android phones. These analysis frameworks are divided into static and dynamic features analysis. The static features analysis is based on the reverse engineering of.apk file of apps. Applications are scanned for malicious code instead of executing them. This is helpful for detecting the malicious codes that only run in explicit circumstances like rebooting; however, the same techniques are not able to identify the encrypted and dynamically loaded malicious codes of android. In the dynamic feature analysis, runtime activities of apps are analyzed and encrypted and dynamically loaded malicious codes of android are identified. Since, all execution paths cannot be examined, which limits overall and complete analysis of the code/apk. The Hybrid techniques combine the advantage of static and dynamic features analysis; however, these techniques have several limitations that force the researchers to develop new hybrid analysis technique based on artificial or machine

learning to achieve high detection rates. All analysis techniques have some advantages and weaknesses for the detection of android malwares. It is also observed that there are several deficiencies in the surveyed presented in the research paper. A quantity of papers has not considered the latest articles in the comparison and analysis. On the other hand, some surveys have not classified the detection techniques systematically for their research work. To overcome these flaws, this paper presents an organized literature survey on the latest android malware detection and analysis frameworks. The main contributions of this paper and study are:

(a) resenting a systematic, chronically and categorized study of the current approaches to android malware detection.
(b) Exploring the architecture and elements of noteworthy android malware detection methods and challenges in these methods.
(c) Demonstrating the importance of deep learning and artificial intelligence for android malware detection and analysis techniques.

The paper is organized as follows. "Review of Android Malware Analysis Techniques" presents a systematic assessment of the existing techniques for detection and analysis of android malwares. In "Discussion," a brief tabulated comparison of malware detection techniques is presented for ready reference. A comparison chart for the efficiency of major android malware detection methods is presented which can work as a benchmark level for new research works. At last, "Conclusion" exhibits the crux of the same study.

2 Review of Android Malware Analysis Techniques

The existing android malware detection and analysis framework are elaborated based on their basic architecture, feature extraction module, test data set, efficiency and advantages and disadvantages. The technique is divided into two parts. The first basic techniques including static and dynamic analysis that are base for any artificial intelligence technique are discussed in chronological order. Then, the latest artificial intelligent techniques for the analysis of android malware are analyzed in detailed.

2.1 Integrated Static and Dynamic Malware Analysis Techniques

There are several android malware analysis techniques that implement the combination of static and dynamic analysis to improve the accuracy of detection of malware. The general architecture of an integrated static and dynamic analysis framework is shown in Fig. 1. Blaising [4] demonstrated an android application (AA) sandbox in 2010 that detects the malicious activities in android apps through

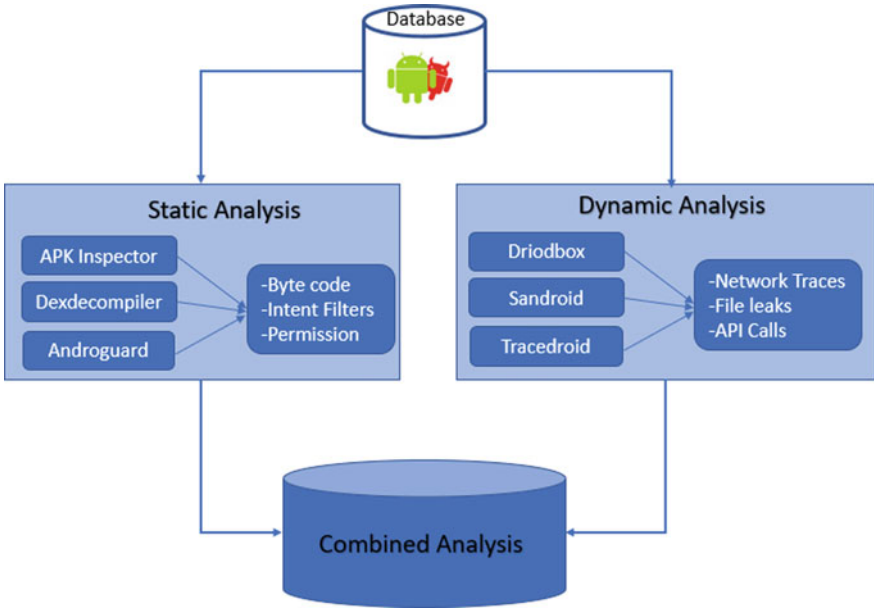

Fig. 1 Statics and dynamic combined malware analysis

analysis both statically and dynamically. In the section of static analysis, android api/code is decompiled by the Baksmali module. Then, the decompiled smali files are scanned and static features are extracted. In the section of dynamic analysis, the apk/code is run through the emulator devices. The Monkey module is also integrated to activate the apps with testing parameters like gestures and clicks. This framework is loaded in the kernel under a fully controlled environment for execution of the apk/code that generates the logs files by capturing the system calls. Vectors are defined based on the log of static and dynamic behavior analysis and these same vectors are inserted into a framework for monitoring of malicious activities of apps and further detection. The main constraint of AASandbox was that it could not be implemented in a large volume of android devices as the root privilege is required for capturing system calls. Authors did not calculate the accuracy of the proposed framework.

Zhou et al. [5] developed a new framework DroidRanger in 2012 for detection of unknown and known malwares. In first section, DroidRanger examines and detects the essential permissions which are used by the malwares to execute the malicious actions that segregate the malicious apps for further analysis in second stage. This data is then collaborated with the defined behavioral rules to distinguish, filter and analysis of identified malwares. However, detection of unknown malwares is achieved in two segments: In the first segment, the filter is defined based on heuristic algorithms. Researcher has implemented two heuristics algorithms in this technique. In the first heuristic technique, the code from the remote server is loaded

dynamically. In the other heuristic technique, the native code is loaded dynamically and the behavior of app is monitored. If app is not stored in default directory, then this behavior is noticeable. Heuristic technique facilitates the framework to determine the malwares which exploit the OS kernel and get root access and provides a capability to detect the DroidDreamLight [6]. Dynamic executions are performed to monitor the runtime behavior of applications. A data set of 204040 android applications was collected in which 75% and 25% were taken from the official and other alternative android markets, respectively. The limitation of this framework is that it developed for only official android markets (Google Playstore) and alternative android markets (eoe market, gfan, alcatelclubm mmoovv, etc.), not for android devices.

Talha et al. [7] developed a new framework for android malware analysis, i.e., APK Auditor in 2015, that uses permissions features for detection of malware behavior. This framework comprises of following segments: (1) database for signatures: store signatures of all the apps; (2) android client: offers the facility of malware analysis to clients; (3) server for processing, which offers a connection between android client and database and processing. The server executes the analysis without installing the apps on the device and optimizes the available assets. APK Auditor monitors all the permissions as called by the apps and determines the value of permission malware score (PMS). Thereafter, this tool classifies the apps as malware if the value of PMS apps scores higher than the threshold value. The outcome demonstrated that APK Auditor accomplished 92.5% accuracy but cannot identify the malicious payloads which installed dynamically.

Abraham et al. [8] developed a 2-Hybrid malware detection framework in 2016 which executes the detection and analysis of android malwares on the server installed at a distant location. The extractor module of framework extracts and accumulates the parameters of the apps and classifies the apps as malware or benign. This framework does not conclude the categorization of apps only based on host analysis but also sent to the server installed at a distant location for exhaustive analysis. If malicious activities are observed in app then server forwards the same data to local device for future detection. As a test set, 39 malicious samples were collected and 69 permissions were monitored from these samples. The results were satisfactory; however, authors did not compare the efficiency of this framework with other frameworks.

Sun et al. [9], developed a new framework MONET in 2017, that detects the malicious codes of an already acknowledged malware family. This framework extracts the static parameters, disassemble the codes, and monitors the dynamic activities of apps for uncovering of malwares. MONET uses client-end applications that run on the user device for analysis of application and design its specific signature. The sets of signatures are stored in the server and a signature matching algorithm is applied. A test set of 3723 and 500 malwares and legitimate apps, respectively, were used for experiments. MONET demonstrated 99% accuracy of detection of malware but shown lacks of resource efficient.

2.2 *Hybrid Malware Analysis with Artificial Intelligence*

The general architecture of the hybrid malware analysis technique with artificial intelligence is shown in Fig. 2. Wang et al. [10] developed a hybrid malicious code analysis framework in 2015 which performs both static and dynamic analysis and implement machine learning for training the framework. It implements signature-based detection to detect the malwares. In first stage, the manifest file is used for static features extraction and android packaging tools are used for disassembling the dex files [11]. In second stage, Cuckoo Droid [12] and Robotium [13] are used for dynamic features extraction. These static and dynamic features are converted into vectors and mapped into vector space. Different feature extraction and selection methods are implemented in this method to increase the accuracy and susceptibility for misuse and anomaly detection. In misuse detection, SelectKBest method is applied where k highest scoring features are selected and Chi2 is used as a scoring function. After feature selection, SVC [14] and SVM classifier [15] are applied for the classification of known and unknown malware, respectively. Based on the probability of signature matching, apps are categorized as malwares and training database is updated accordingly. In case, if there is any uncertainty, then only anomaly detection is applied and if any abnormal behavior is detected, then app is classified as unknown malware and training database is updated accordingly. A test set of 12000 and 5560 for benign and malware apps are collected,

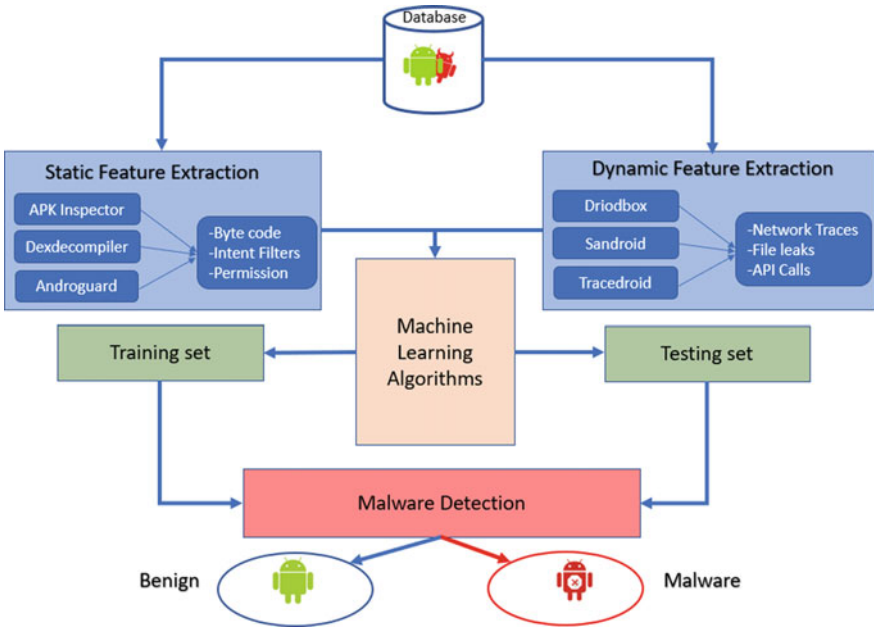

Fig. 2 Hybrid malware analysis using artificial intelligence

respectively, from various App stores. The Outcome demonstrated that the framework can detect the malwares with an accuracy of 98.79% from the used test set.

Sahijo et al. [16], developed an integrated framework for malware detection and analysis based on machine learning in 2015. This framework assimilates the static analysis and dynamic analysis for detection and analysis with training of modules of machine learning. Framework extracts printable strings information as static feature from app and then sorted as per the rate of incidence in each file. After extraction, feature selection is performed such that PSI features with the rate of incidence above benchmark are selected. Selected features are then enlisted into global list named feature list. This feature list is used for creating static feature vector. In next stage, Cuckoo platform is implemented and API call logs are extracted for dynamic feature extraction of apps. API call grams are generated for all codes and sorted as per the rate of incidents. The vectors for static and dynamic features are designed and inserted into two separate machine learning modules SVM [17, 18] and random forest [19] for classification. This framework demonstrated 98.7% accuracy on a test set of 997 malicious and 490 benign apps; however, huge storage and high processing power are required for the same framework.

Yuan et al. [20] developed an online framework DroidDetector in 2016 to detect malwares online. In this framework, both the static and dynamic features extraction tools are implemented in the servers hosted remotely and then machine learning modules are integrated for discriminating between malwares and benign apps. Major permissions and sensitive API are monitored to determine the static features. DroidBox is applied for a specific period of time to monitor the dynamic activities of apps. The static and dynamic features are mapped into vectors and these features are inserted into the machine learning module for learning of detection of malicious apps. A test set of 20,000 apps of a combination of both legitimate and malware apps is used to evaluate the framework of DroidDetector. Experimental results demonstrated the achievement of 96.7% accuracy for the detection of malwares. The key flaw of this framework is that the various malicious apps can be escaped from the detection of framework if malicious behavior of apps is not observed during the observation time-period.

Yu et al. [21] developed a hybrid automatic static-dynamic switch framework in 2016 to counter the transformation technique adopted by the malwares. This framework can distinguish the malicious codes of apps by either static or dynamic detection and analysis technique. In first section, apps are decompiled through Apktool [22] and accordingly manifest and Smali files are created and static malware analysis is applied. The extracted static features are mapped into the vectors and these vectors are inserted into machine learning modules for detection of the apps as malware. However, if apps are not properly decompiled due to transformation techniques, then automatic dynamic malware analysis is executed on the apps. In the same way, dynamic malware analysis is performed on apps and extracted dynamic features are mapped with vectors. These vectors are also inserted into machine learning modules like kNN and Naive Bayes, etc., for training of framework for detection. Framework demonstrated the 99% accuracy for static

detection and 90% accuracy for dynamic detection. However, the main flaw of this framework is that only static malware analysis process is executed if the app is decompiled properly. Framework does not have capability of detection of dynamically loaded codes in the app. Therefore, app that is not able to execute accurately cannot be analyzed properly and accordingly cannot be classified as malware or benign app.

Saracino et al. [23], developed a host-based novel framework MADAM in 2016 which suspicious the apps based on their misbehavior and malicious activities. The framework segregates the malware into a number of behavioral modules and each module execute specific misbehavior that is tagged with the same malware. Framework extracted and correlated features at four stages: package, application, user, and kernel. MADAM structural design consists of four main pillars that are App Risk Assessment, Global Monitor, App Monitor, and User Interface and Prevention. MADAM executes App Risk Assessment process and generates a list of suspicious apps. Framework applies the static and dynamic analysis to determine the risk of app. The Global monitor process monitors the system call, user activity, message and activity logger and generates feature vectors. These vectors are inserted in a machine learning module like kNN and are used to train the framework for legitimate and malicious behaviors. App monitor process decompiled the malicious codes by analyzing the behavior of malwares at API and kernel level. App Monitor continuously monitors the (a) Background apps with administrator privileges, (b) Automatic SMS send-receive apps, and (c) Foreground apps. Lastly, the user interface warns about the malicious apps and assistance for blocking and removing these apps. The proposed framework is tested on a test set collected from virus share and demonstrated 96.9% detection rate. However, MADAM framework has one flaw that a root privilege is required on the device to execute the detection and this cannot be executed in the mass market.

Hou et al. [24] developed a heterogeneous information network, i.e., Hindroid for android malware analysis in 2017. This framework created higher-level semantics in place of application programming interface (API) calls. HinDroid framework created a heterogeneous information network (HIN) of android applications and a meta-path to provide connection to these applications. All meta-path is required to calculate a similarity measure over android applications and automatically weighted by the learning algorithm to make predictions. The authors claimed that HinDroid shows better performance than other android malware detection systems; however, accuracy and efficiency were not provided.

Karbab et al. [25] developed an automatic framework MalDozer using deep learning in 2018 for android malware analysis. MalDozer automatically extracts the features and learns the new patterns from the actual samples to detect the malwares. MalDozer can be deployed not only on remote servers but also on android mobile agents. MalDozer offered many advance features like automatic features extraction in the training phase and minimal preprocessing power. A test sets of 33,000 malwares and 38,000 benign apps are used to evaluate the performance of framework. MalDozer demonstrated that malware can be correctly detected with an F1-Score of 96–99%.

Li et al. [26] introduced a new framework in 2018 for malware analysis system based on Significant Permission IDentification (SigPID). This framework is designed in three systematic stages of pruning approach with machine learning techniques to identify the most significant permissions so that framework can be utilized effectively. SigPID demonstrated that only 22 permissions out of 132 permissions are significant and improve the runtime performance by 85.6%. This framework also demonstrated that 90% of precision can be achieved by the support vector machine (SVM). A test set of 1,000 malwares was applied for testing and SigPID technique demonstrated 93.62% and 91.4% efficiency in detection of known malware and unknown/new malware samples, respectively.

Feng et al. [27] proposed an innovative ensemble learning-based EnDroid in 2019 for android malware detection. This framework integrated multiple types of dynamic features and analysis techniques and achieved very precise malware detection accuracy. These analysis techniques include the monitoring of malicious behavior of system and application level like stealing of critical data, uploading on a remote server, and update of firmware with malicious codes. Two datasets were taken for experiments and result demonstrated that stacking accomplished the best detection performance and exhibits 96.56% efficiency in android malware detection.

Zhou et al. [28] presented a new framework incorporating the control flow graph with machine learning algorithms in 2019 for android malware analysis. In this framework, the applications are decompiled and a control flow graph is constructed to obtain the API information. Three types of system API uses data sets, (1) API calls, (2) API frequency, and (3) API sequence based on control flow graphs are constructed to develop three detection models. The accuracy of all three models is compared using Precision, Recall and F-score metrics. The Framework demonstrated 98.98% detection precision on a test set of 10,683 malicious and 10,010 benign applications.

Mehtab et al. [29] proposed an innovative approach, AdDroid in 2019 to detect android malwares based on a variety of permutations of artifacts. The rules/artifacts designate the activities of codes of android device like establishing a connection to the ISP through the internet, secretly uploading personal data to the pre-defined server, updating malicious package/patches of firmware, etc. AdDroid uses ensemble-based machine learning algorithms, i.e. Adaboost to train the model for static analysis of android apps. AdDroid is able to extraction and selection of statics feature and then able to recognizing malicious applications based on the most unique rules. This machine learning framework is trained and developed by applying a test set comprising 1,420 apps including 910 malicious and 510 benign android apps. The framework demonstrated an accuracy of 99.11% on a similar test set. The Authors did not include dynamic features for machine learning.

Ma et al. [30] proposed the deep learning-based framework Droidetec in 2020, for android malware detection and exact localization of malicious codes in the apps. This framework implemented an innovative feature extraction method to monitor behavior sequences from malicious apps. Extracted behavior sequences are represented as a vector that automatically scrutinizes the semantics of sequence of

fragment and determines the malicious code. A test set of 9,616 malicious and 11,982 benign programs is used to evaluate the capability of Droidetec framework and the result demonstrated a precision of 97.22% for detection.

Mohammed et al. [31] proposed a deep learning framework, i.e., DL-Droid in 2020 to detect the malicious apps through dynamic features and generation of stateful input. DL-Droid framework is implemented with 30,000 benign and malware apps on real android terminals. Result demonstrates that the performance of detection of this framework with deep learning module is better than existing traditional methods. DL-Droid achieved accuracy of 97.8% for detection with dynamic features only and accuracy of 99.6% for detection with dynamic and static features. However, self-adaptation for intrusion detection systems is not available to improve the performance of model for malware detection.

Su et al. [32] proposed DroidPortrait in 2020, an approach of construction of multi-dimensional and vertical behavioral representation for detection of android malwares. In the analysis, the behavior of android malware is monitored and static and dynamic behavior as dataset are extracted. In the next phase of analysis, different kinds of behaviors are segregated based on android malware and a specific and unique behavioral tag is attached with its signature. Machine learning (ML) algorithms are implemented to correlate these behavioral tags with specific malwares automatically. A high performance machine learning algorithm, i.e., random forest algorithm is very suitable and easily integrates with the basic framework to detect the android malware. The result demonstrated that DroidPortrait framework can illustrate the behavior appearances of android malware with high accuracy. The efficiency level was 90% which is lower and can be increased further.

Mahindru et al. [33] proposed MLDroid in 2020, the web-based model to analyze the android apps as malware and benign. Authors implemented feature selection techniques and trained the framework by these techniques. These selected features developed an innovative model by implementing different machine learning algorithms. A test set of 500,000 plus android apps are used for the experiment and the four distinct machine learning algorithms (1) deep learning, (2) first and farthest clustering, (3) YMLP, and (4) nonlinear ensemble decision tree forest is applied in parallel. Experiment results exhibit that framework developed by considering all the algorithms can achieve an accuracy rate of upto 98.8% for detection of malware from android apps.

Zhang et al. [34] proposed an automatic framework TC-Droid in 2020, for android applications analysis based on text classification technique. In this framework, the text sequences of APPs are analyzed by AndroPyTool and generated analysis reports are feed in deep learning algorithms. A machine learning algorithm, i.e., convolutional neural network (CNN) is used to extract considerable information instead of manual feature engineering from the same analysis report. A variety of well-known samples were collected for evaluation, and it is demonstrated that the performance of TC-Droid is better than other classic algorithms, i.e., NB, LR, KNN, RF, etc. However, actual data for accuracy and efficiency were not provided for comparison.

Thongsuwan et al. [35] demonstrated a hierarchical approach in 2021 using extraction of authorization-sensitive feature and implementing deep learning algorithms to design an android malware detection framework. This framework extracts four sensitive features: basic blocks, permissions, api, and key functions used for authorization. An innovative machine learning model, i.e., convolution neural network and eXtreme Gradient Boosting (CNNXGB) are implemented for training, learning and detection of malware. This framework sequences the key functions as per the timing of API calls and collects a similar section that confines the global semantics of malware family. Permissions and API calls were extracted from 1,330 android test samples to drill the model by XGBoost. The result demonstrated that efficiency increased upto 98% by the CNNXGB model. However, as the input data for analysis is increased, the number of the convolution layers and complexity increases.

McDonald et al. [36] developed a framework based on manifest permission and machine learning for android malware detection in 2021. In this framework, four different machine learning algorithms (1) random forest, (2) support vector machine, (3) Gaussian Naïve Bayes and (4) K-Means are applied in conjunction with features selected from android manifest file permissions to distinguish the apps as malicious or benign. A test set of 5,243 samples are used to test the framework and it was demonstrated that random forest ML algorithm performed the best with 82.5% precision and 81.5% accuracy.

Above mentioned techniques are briefly explained and tabulated in Table 1 for ready reference for researchers.

Table 1 Overview of the existing framework and research gaps

S.no	Reference	Year	Methodology/ contribution	Research gap
Integrated static and dynamic malware analysis techniques				
	Android application (AA) sandbox [4]	2010	Uses static and dynamics analysis	Root privilege is needed to capture the system calls
	DroidRanger [5]	2012	Uses static and dynamic analysis	Developed for official android and alternative android markets only
	APK Auditor [7]	2015	Uses static analysis with permission-based malware detection	Framework is installed at central server, and therefore, internet connection is required for android terminal for the malware analysis at server

(continued)

Table 1 (continued)

S.no	Reference	Year	Methodology/ contribution	Research gap
	Novel hybrid android malware detection [8]	2016	Static and dynamic features of the applications, are extracted at the server configured at distant. If app is distinguished as risk, database is updated for this risk signature	Efficiency and accuracy of framework were not determined and compared with another existing framework
	MONET [9]	2017	The specific signatures for respective malware are generated and forwarded to server for matching and detection	This framework works only with android DVM; however, latest android supports only ART
Hybrid malware analysis with artificial intelligence				
	A novel anomaly and misuse hybrid mobile malware detection system [10]	2015	Static features from manifest file and dynamic features through CuckooDroid are used with SVC classifier in misuse detection	Comparison table for detection with other frameworks are not computed
	Detection and mitigation of android malware through deep learning [16]	2015	Static features from printable strings information (PSI) and dynamic features from Cuckoo are extracted. These features are inserted in machine learning modules for classification	Power and storage consumption are intense
	DroidDetector [20]	2016	The extracted static and dynamic features through different tools are applied in deep learning algorithm for classification as malware	Malwares are escaped from the detection system during the dynamic monitoring time-interval
	A hybrid automatic static-dynamic switch framework [21]	2016	Analysis is conditional-based. If properly decompiled the app, extracted static vectors are inserted into deep learning algorithms like SVM, kNN and Naïve Bayes. However, in case, app is not decompiled properly then only dynamic are inserted into machine learning for classification	Only performs one analysis, i.e., static or dynamic. Framework will not be able to detect the malicious codes if app is decompiled properly

(continued)

Table 1 (continued)

S.no	Reference	Year	Methodology/ contribution	Research gap
	MADAM [23]	2016	Features are extracted at four stages: (1) package, (2) application, (3) user and (4) kernel. These features are mapped with vectors and inserted as input in machine learning kNN classifier for training	It runs only on rooted device, therefore, not supported in the large users. Moreover, pre-loaded apps/games cannot be analyzed by this framework
	Hindroid [24]	2017	Created heterogeneous information network (HIN) of android applications is used with machine learning algorithms	Accuracy and efficiency were not provided
	MalDozer [25]	2018	Automatically extracts the features and learns the new patterns from the actual samples through machine learning algorithms	Less effective in malware family attribution
	Significant permission IDentification (SigPID) [26]	2018	Instead of all android permissions, only most significant permissions SigPID are extracted and utilized for machine learning models	The analysis can be performed only on the rooted devices
	EnDroid [27]	2019	Dynamic features with system-based behavior trace and common apps-based malicious behaviors are used for deep learning	Power consumption is very high
	A control flow Graph-based android malware detection including machine learning [28]	2019	An ensemble of three detection models is created based on control flow graph of three data sets for android malware detection and analysis	Capability to establish the malware family is not presented
	AdDroid [29]	2019	Only static features are extracted and inserted into the ensemble-based machine learning model that is trained by the Adaboost	Artifacts called rules were defined for only static analysis. Dynamic analysis is not integrated with the model

(continued)

Table 1 (continued)

S.no	Reference	Year	Methodology/ contribution	Research gap
	DroidDetec [30]	2020	An innovative behavior sequences feature is extracted for analysis of syntax of linked segments and finding of malicious code	Not accurate framework for further classification into various malware families
	DL-Droid [31]	2020	Dynamic feature extraction with stateless approach is used with deep learning algorithms for analysis of malicious android apps	Self-adaptation for Intrusion detection systems is not available to improve the performance of model
	DroidPortrait [32]	2020	Extracted static and dynamic behavior and create an informative behavior dataset that includes specific behavior tag for specific android malware	The efficiency level was 90% which can be increased further
	MLDroid [33]	2020	Static and dynamic features are extracted and inserted into 04 different machine learning algorithms in parallel	This framework has capability for only detection the app as malware or benign. Detection rate is lower
	TC-Droid [34]	2020	Convolution neural network (CNN) algorithm is used for extraction of significant tags instead of manual features	Data for accuracy and efficiency were not provided
	ConvXGB [35]	2021	Authorization-sensitive features are extracted for machine learning algorithm of convolution neural network and eXtreme gradient boosting (CNNXGB)	Number of the convolution layers is increased, depending on the input data for analysis
	Manifest permission (MP) and machine learning (ML)-based framework [36]	2021	Static features are selected and inserted into four different machine learning algorithms for grading the apps as malicious or benign	Other remaining static features can also be explored further to form a greater feature set

3 Discussion

Based on the review of all frameworks, an overall comparative analysis among all malware detection frameworks based on static and dynamic features and hybrid features using artificial intelligence are analyzed. We discover that the hybrid method using artificial intelligence has the best accuracy in comparison with statics and dynamic malware analysis detection approaches. The accuracy efficiency of each framework is analyzed and it is observed that all frameworks have an accuracy efficiency higher than 80%. It is also examined that the DL-Droid malware analysis framework has maximum accuracy efficiency that is 99.6% [31], and machine learning-based android malware detection framework using manifest permissions has minimum accuracy efficiency that is 81% [36].

4 Conclusion

A methodical and chronically literature investigation of the detection and analysis frameworks and techniques for android malware are explained. The work done by researches were reviewed and investigated and existing android malware analysis frameworks were categorized into two categories: (1) static and dynamic malware analysis and (2) hybrid malware analysis using artificial intelligence. These malware analysis frameworks were compared and analyzed according to their specific features and technique. The advantage and disadvantage of each analysis framework were deliberated. It is essential to develop the frameworks which automatically learn without the intervention of human and detect the zero-day malwares. Therefore, artificial intelligence techniques surfaced as a potential solution to handle the different types of malwares. This state-of-the-art research survey is a fundamental instrument for further any research which can make a tremendous change in the development of a new framework.

References

1. Gartner says worldwide sales of smartphones grew in the first quarter of 2020 [Online]. Available: http://www.gartner.com/newsroom/id/3609817 (2020)
2. Number of smartphone users in India in 2015 to 2020 with a forecast until 2025 [Online]. Available: https://www.statista.com/statistics/467163/forecast-of-smartphone-users-in-india/ (2020)
3. Trend micro threat report | review, refocus and recalibrate, the 2019 mobile threat landscape [Online]. Available: https://www.trendmicro.com/vinfo/hk-en/security/research-and-analysis/threat-reports/roundup/review-refocus-and-recalibrate-the-2019-mobile-threat-landscape (2019). Accessed 25 Mar 2020

4. Bläsing, T., Batyuk, L., Schmidt, A.D., Camtepe, S.A., Albayrak, S.: An android application sandbox system for suspicious software detection. In: 2010 5th International Conference on Malicious and Unwanted Software, October 2020, pp. 55–62. IEEE
5. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In: NDSS, February 2012, vol. 25, no. 4, pp. 50–52
6. Balanza, M., Abendan, O., Alintanahin, K., Dizon, J., Caraig, B.: Droiddreamlight lurks behind legitimate android apps. In: 2011 6th International Conference on Malicious and Unwanted Software, October 2011, pp. 73–78. IEEE (2011)
7. Talha, K.A., Alper, D.I., Aydin, C.: APK auditor: permission-based android malware detection system. Digit. Investig. **13**, 1–14 (2015)
8. Rodriguez-Mota, A., Escamilla-Ambrosio, P.J., Morales-Ortega, S., Salinas-Rosales, M., Aguirre-Anaya, E.: Towards a 2-hybrid Android malware detection test framework. In: 2016 International Conference on Electronics, Communications and Computers (CONIELECOMP), February 2016, pp. 54–61. IEEE (2016)
9. Sun, M., Li, X., Lui, J.C., Ma, R.T., Liang, Z.: Monet: a user-oriented behavior-based malware variants detection system for android. IEEE Trans. Inf. Forensics Secur. **12**(5), 1103–1112 (2016)
10. Wang, X., Yang, Y., Zeng, Y., Tang, C., Shi, J., Xu, K.: A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection. In: Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services, September 2015, pp. 15–22
11. Android aapt—eLinux.org [Online]. Available: http://elinux.org/Android_aapt (2016). Accessed 13 Aug 2016
12. CuckooDROiD: Dalvik monitoring framework for CuckooDroid (2020)
13. Ghahrai, A.: 10+ open source mobile test automation tools. Retrieved September, 16, 2016 (2014)
14. Gunn, S.R.: Support vector machines for classification and regression. ISIS Tech Rep **14**(1), 5–16 (1998)
15. Li, K.L., Huang, H.K., Tian, S.F., Xu, W.: Improving one-class SVM for anomaly detection. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693), November 2003, vol. 5, pp. 3077–3081. IEEE (2003)
16. Shijo, P.V., Salim, A.J.P.C.S.: Integrated static and dynamic analysis for malware detection. Proc. Comput. Sci. **46**, 804–811 (2015)
17. Andrew, A.M.: An introduction to support vector machines and other kernel-based learning methods. Nello Christianini and John Shawe-Taylor, Cambridge University Press, Cambridge, 2000, xiii+ 189 pp., ISBN 0–521–78019–5 (Hbk,£ 27.50). Robotica **18**(6), 687–689 (2000)
18. Andrew, A.M.: An introduction to support vector machines and other kernel-based learning methods. Kybernetes (2001)
19. Dittman, D., Khoshgoftaar, T.M., Wald, R., Napolitano, A.: Random forest: a reliable tool for patient response prediction. In: 2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), November 2011, pp. 289–296. IEEE (2011)
20. Yuan, Z., Lu, Y., Xue, Y.: Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci Technol **21**(1), 114–123 (2016)
21. Liu, Y., Zhang, Y., Li, H., Chen, X.: A hybrid malware detecting scheme for mobile android applications. In: 2016 IEEE International Conference on Consumer Electronics (ICCE), January 2016, pp. 155–156. IEEE (2016)
22. Winsniewski, R.: Apktool: a tool for reverse engineering android apk files. https://ibotpeaches.github.io/Apktool/ (2012). Visited on 27 July 2016
23. Saracino, A., Sgandurra, D., Dini, G., Martinelli, F.: Madam: effective and efficient behavior-based android malware detection and prevention. IEEE Trans. Dependable Secure Comput. **15**(1), 83–97 (2016)

24. Hou, S., Ye, Y., Song, Y., Abdulhayoglu, M.: Hindroid: an intelligent android malware detection system based on structured heterogeneous information network. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1507–1515. ACM (2017)
25. Billah, K.E.M., Mourad, D., Abdelouahid, D., Djedjiga, M.: MalDozer: automatic framework for android malware detection using deep learning. Digit Investig **24**, S48–S59 (2018)
26. Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., Ye, H.: Significant permission identification for machine-learning-based android malware detection. IEEE Trans. Industr. Inf. **14**(7), 3216–3225 (2018)
27. Feng, P., Ma, J., Sun, C., Xu, X., Ma, Y.: A novel dynamic android malware detection system with ensemble learning. IEEE Access **6**, 30996–31011 (2018)
28. Ma, Z., Ge, H., Liu, Y., Zhao, M., Ma, J.: A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE Access **7**, 21235–21245 (2019)
29. Mehtab, A., Shahid, W.B., Yaqoob, T., Amjad, M.F., Abbas, H., Afzal, H., Saqib, M.N.: AdDroid: rule-based machine learning framework for android malware analysis. Mobile Netw. Appl. **25**(1), 180–192 (2020)
30. Ma, Z., Ge, H., Wang, Z., Liu, Y., Liu, X.: Droidetec: Android malware detection and malicious code localization through deep learning. arXiv preprint arXiv:2002.03594 (2020)
31. Alzaylaee, M.K., Yerima, S.Y., Sezer, S.: DL-Droid: deep learning based android malware detection using real devices. Comput. Secur. **89**, 101663 (2020)
32. Su, X., Xiao, L., Li, W., Liu, X., Li, K.C., Liang, W.: DroidPortrait: android malware portrait construction based on multidimensional behavior analysis. Appl. Sci. **10**(11), 3978 (2020)
33. Mahindru, A., Sangal, A.L.: MLDroid—framework for android malware detection using machine learning techniques. Neural Comput. Appl. 1–58 (2020)
34. Zhang, N., Tan, Y.A., Yang, C., Li, Y.: Deep learning feature exploration for android malware detection. Appl. Soft Comput. **102**, 107069 (2021)
35. Thongsuwan, S., Jaiyen, S., Padcharoen, A., Agarwal, P.: ConvXGB: a new deep learning model for classification problems based on CNN and XGBoost. Nucl. Eng. Technol. **53**(2), 522–531 (2021)
36. Mcdonald, J., Herron, N., Glisson, W., Benton, R.: Machine learning-based android malware detection using manifest permissions. In: Proceedings of the 54th Hawaii International Conference on System Sciences, January 2021, p. 6976