IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 25, Issue 6, Ser. 2 (Nov. – Dec. 2023), PP 72-78
www.iosrjournals.org

A review of Static Analysis of Android Malware

Ladwinder Kaur,*Charanjiv Singh Saroa,*Jaswinder Singh
Department of Computer Science and Engineering Punjabi University Patiala, 147002, Punjab, India
*Email: ladwinderladdi1206@gmail.com,*charanjiv@pbi.ac.in,*jaswindersinghmtech@gmail.com*

Abstract:
Android, being an open-source operating system, offers a comprehensive understanding of its architecture. To build mobile devices (smartphones, smartwatches, and smart glasses) under various brand names, including as Google Pixel, Motorola, Samsung, and Sony, numerous manufacturers are using this OS. This widespread adoption has significantly increased the Android user base. However, some unscrupulous individuals develop malware for personal gain, recognition, or other nefarious purposes. To combat Android malware, researchers have proposed different techniques. Among these techniques, ML-based methods that utilize static features of apps as input vectors have demonstrated clear advantages in terms of code coverage, operational efficiency, and the ability to detect a large number of samples. In this study, we examined the structure of Android applications, analyzed various sources of static features, reviewed ML methods used for detecting Android malware, assessed the strengths and limitations of these methods, and explored future directions in this field. Overall, this research acts as a valuable resource for both beginner security professionals and experienced researchers, providing direction for the creation of innovative studies aimed at identifying malware using static analysis techniques.

I. Introduction

Android:

In the last ten years, cellphones have become incredibly popular, attracting billions of users worldwide. The convenience and versatility of smartphones have contributed to their widespread adoption. These devices offer a multitude of functionalities, including sending emails, playing games, capturing photos and videos, browsing the internet, utilizing GPS, and much more. The Android operating system, initially developed as a modified Linux kernel for touchscreen devices, has played a significant role in enabling these capabilities. The Android ecosystem has witnessed an exponential growth of applications, surpassing 3.5 million apps last year. These apps serve various purposes such as banking, social media, healthcare, education, and entertainment. The total number of mobile network subscriptions for smartphones worldwide was almost 6.4 billion in 2022 and is expected to surpass 7.7 billion by 2028. Leading countries in terms of smartphone subscriptions include China, India, and the United States [1].According to a report by the International Data Corporation (IDC) in the second quarter of 2016, Google's Android OS dominated the smartphone market with a substantial market share of 87.6%. On the other hand, Apple's iOS secured the second position with a share of 11.7%. However, iOS has experienced stagnant growth and a gradual decline in market share. The appealing aspect of these mobile platforms is the availability of feature-rich applications.

Android Malware

The majority of the above applications are utilized for the benefit of their users. However, some are employed maliciously by hackers and exploiters. These malicious programmes, sometimes referred to as malware, are intrusive programmes intended to steal data or destroy a user's machine. Cybercriminals create malware in a variety of formats, such as Trojan viruses, worms, ransomware, and adware.[2]. Keeping up with security breaches has become increasingly challenging due to the constant evolution of malicious software. For example, in 2021, Check Point, a cybersecurity firm, warned Android users about the vulnerability of millions of smartphones to the Agent Smith malware [3]. This spyware also utilized WhatsApp as a disguise to target Android systems. In the same year, coccus reported that over a billion Android smartphones were at risk of hacking due to the absence of up-to-date security updates [3]. Recently, there have been cases of Facebook accounts being hacked using the android malware app called "FlyTrap". These malicious apps steal personal information, initiate unauthorized calls and text messages to initiate deductions, and exploit root privileges to take access over the machine, causing serious harm to users. To combat these security threats, various Android Malware detection techniques have been developed. One effective approach is the utilization of machine learning-based detection methods.

Malware Detection Techniques:

Anomaly Detection: Anomaly detection in cybersecurity utilizes artificial intelligence to establish a standard operating pattern and detects variances from this pattern. While it can uncover new threats, it frequently produces a considerable number of false positive results.

Behavioral Detection: Unusual activities, like opening and encrypting numerous files, are frequently exhibited by malware. Behavioral detection is employed to identify the existence of malware on a system by identifying these atypical behaviors

Static Analysis: Static analysis refers to the examination of a suspicious or malicious executable without executing it. This method ensures safety while examining malware and offers valuable insights into the functionality of the malware, as well as indicators of compromise (IoCs) that can aid in signature detection.

Honeypots: Honeypots are purposely created systems that mimic attractive targets for attackers or malware. If these systems become infected, security experts can analyze the malware and develop protective measures for their actual systems.

Allowlisting: An allowlist outlines the approved items on a system, while blocking anything not listed. In the context of malware detection, an allowlist can be used to specify authorized files on a system, with the assumption that all other programs are potentially malicious.

Classification Techniques of Android Malware

This section provides an overview of the process of classifying Android malware based on extracted features from selected subsets of valid features. There are three main categories of Android malware detection methods: signature-based, heuristic-based, and machine learning-based.

Signature-based detection is the most mature method and relies on pattern matching using a library of malware signatures. Each known Android malware has a unique signature stored in the library, and the detection process involves checking if a sample matches any of the signatures. This method is widely used due to its speed and accuracy, but it requires manual or automatic maintenance of the signature library and may not be effective against new malware.

Heuristic-based detection, also known as anomaly-based detection, focuses on identifying unknown malicious software. In order to detect whether a sample belongs to a certain malware family, this method compares the traits of unknown samples with those of known malware families and applies sets of rules. It can discover unknown malicious software and employs multiple methods to distinguish between benign software and malicious. However, it may have a higher error rate for zero-day malware.

ML-based detection involves training a learner by adjusting parameters to make accurate predictions. It has proven to be an effective method for detecting Android malware, and numerous studies have explored its potential in detecting unknown Android malware. In the following subsections, we will delve into the details of machine learning-based detection technologies.

Feature used for static analysis of android malware detection

- Permissions: Android malware often requests excessive or unusual permissions. For example, a flashlight app may not need permission to access the user's contacts or location.
- API calls: Static analysis can be used to identify suspicious API calls, such as calls to methods that send data to remote servers or that modify the Android system.
- Code structure: Static analysis can be used to identify unusual or suspicious code patterns, such as the use of reflection or the loading of code from unknown sources.
- String resources: Static analysis can be used to identify malicious strings in the app's resources, such as strings that contain phishing URLs or that are known to be associated with malware.
- Manifest file: The manifest file contains information about the app, such as its permissions, components, and activities. Static analysis can be used to identify suspicious entries in the manifest file, such as apps that declare activities that are not actually implemented.

Methodology: In Figure 1 the architecture of static analysis of android malware Detection has been shown. Firstly, we gather several applications from app stores as well as unofficial malapps. Second step would be to extract static features from from each apk. After that we use different Classification Algorithms for detecting malicious or benign application.

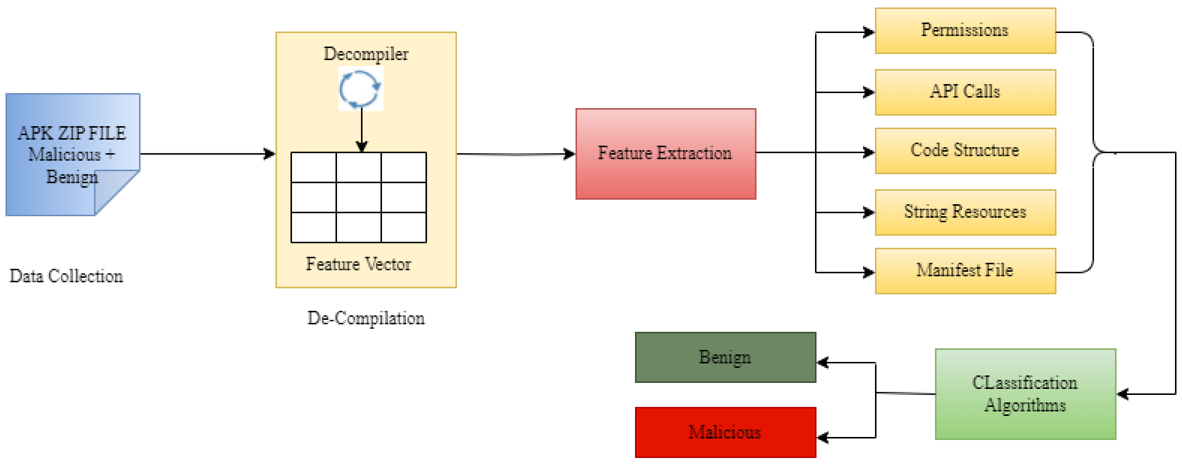

Figure 1 Architecture of Static Analysis of Android Malware Detection

Related works

Amin et al. [4] proposes an innovative and definitive method for detecting malware on Android devices. Researcher have introduced two distinct detection methods: one based on network analysis, which involves capturing network packets, tracing their URLs, and cross-referencing them with a database of malicious servers, and another based on system call analysis, which entails measuring the frequency of system calls made by an app and comparing it to those of known malicious apps. The results indicate that our system call-based approach achieves an accuracy of 87% in detecting malware, which is quite significant within the broader context of malware detection.

Shabtai et al. [5] conducted a study on the utilization of static analysis techniques to examine Android source code. Additionally, they employed ML methods to categorize games and tools based on static features extracted from Android apps. We conducted the evaluation using a dataset comprising 2,850 games and tools. For this purpose, we extracted features from Android application (.apk) files and employed them as input during the classification process. As there were no malicious applications available for Android at the time, our evaluation focused solely on distinguishing between tools and games. We conducted the evaluation using a dataset comprising 2,850 games and tools. The results indicate that by merging Boosted Bayesian Networks with the top 800 characteristicschosen using Information Gain, we achieved an accuracy level of 0.918, with a false positive rate (FPR) of 0.172.

The author in [6] introduce MARVIN, a system that integrates static and dynamic analysis and utilizes ML methods to evaluate the potential risk posed by unfamiliar Android apps, quantified as a malice score. MARVIN conducts off-device static and dynamic analysis to capture various properties and behavioral characteristics of an app, employing a comprehensive set of featuresIn our analysis, utilizing the largest dataset of Android malware classifications to date, which includes over 135,000 Android applications and 15,000 instances of malware, MARVIN achieves an accuracy of 98.24% in identifying malicious apps, with an extremely low false positive rate of under 0.04%.

This article[7] presents ThinAV, an Android anti-malware system that utilizes existing web-based file scanning services for detecting malware. The objective of developing ThinAV was to investigate the feasibility of offering real-time anti-malware scanning over a resource-constrained wide area network. Consequently, our research offers a contrasting perspective to the resource-intensive, high-budget solutions often proposed in the field of cloud-based security. The evaluation of ThinAV demonstrates its effective performance over a wide area network, establishing it as a highly practical system for delivering anti-malware security on smartphones.

In this paper,[8] we have presented a solid approach that utilizes static analysis of Android APK files for extracting features such as API Calls, Permissions, Command signature and Intents. To train our machine learning classifier, we have utilized the Malgenome and Drebin datasets.After performing feature extraction from the Android APK files, we conducted several experiments, considering both the combined extracted features and the pairwise combinations of permissions.We then trainedSVM, Random Forest , and SVM with Principal Component Analysis classifiers to differentiate between benign and malicious applications. In terms of accuracy, the RF classifier outperformed the others, achieving a 96.27% accuracy rate when using the Drebin permission feature combination. On the other hand, the SVM with PCA classifier performed the best when using the Malgenome dataset, achieving a maximum accuracy of 97.63% with the Malgenome permission combination.

In this research paper[9], In our academic paper [9], we present a method for identifying Android applications using a static approach. This method involves extracting permissions and API calls as features. We utilize various ML methods in our approach. The experimental results show that our technique achieves an accuracy rate of over 90%.. Additionally, we conclude that the feature set combining "Permissions + API" generally performs better for classification compared to a feature set consisting of only "Permissions".

In this research paper[10], our proposal involves combining permissions and API calls in order to detect malicious Android applications. Our approach entails extracting permissions from each application's profile information and extracting APIs from the packed application file using packages and classes to represent API calls. By utilizing these permissions and API calls as features, we are able to train a classifier to determine whether an application is potentially malicious. An inherent advantage of our method is that it relies solely on simple static analysis and does not require dynamic tracing of system calls.

In this research paper[11], we utilize an existing attack framework to classify various attack scenarios against machine learning-based malware detection tools. We consider attackers with different levels of expertise and capabilities, and based on this, we develop evasion attacks to evaluate the effectiveness of Drebin, an Android malware detector. The key contribution of our study is the introduction of a secure-learning paradigm that effectively addresses the impact of evasion attacks, while only marginally decreasing the detection rate when there are no attacks. We also suggest that our secure-learning approach can be easily applied to other malware detection tasks.

In this research[12], we propose a novel approach called DroidEnsemble for detecting Android malicious applications (malapps) by utilizing both string features and structural features. Our goal is to build a more accurate detection model by comprehensively characterizing the static behaviors of Android apps. We extract various string features such as hardware features, restricted API calls, permissions, filter intents, used permissions, and code patterns, along with structural features like function call graphs.To evaluate the performance of these feature types and their ensemble, we employ three ML algorithms: SVM, KNN, and RF. Our evaluation involves a dataset consisting of 1386 benign apps and 1296 malapps. The extensive experimental results demonstrate the effectiveness of DroidEnsemble. When using only string features, DroidEnsemble achieves a detection accuracy of 95.8%.

In this research paper[13], we presented a novel technique for detecting Android malware by utilizing multiple features and the TF-IDF algorithm. Our approach incorporates static features of the Android application package. To process the permission feature, we employed the bag of words (BOW) technique, while standardization was used for processing the package size feature. The TF-IDF algorithm was utilized to handle the API calls features. For classification, we employed machine learning algorithms including AdaBoost, LinearSVC, and GaussianNB. Through experiments conducted on public datasets, we demonstrated that our system is capable of efficiently detecting Android malware.

The author in [14] propose novel machine learning techniques for detecting Android malware by utilizing a feature set consisting of API calls and permissions. These features are commonly used to identify malicious applications. Initially, we examine the viability of a "yearly dataset-based trained classifier" (YDataC), which involves training the classifier on 80% of the dataset from a specific year (2014-2021) and testing it on the remaining 20% from each year. Our experiments reveal a significant decline in classification accuracy since 2019, indicating a major shift. Consequently, the YDataC approach is deemed unsustainable. To address this, we introduce and evaluate two incremental learning methods for gradual training: an incrementally trained Random Forest (RF) and an incrementally trained Neural Network (NN). Evaluation results demonstrate that both incremental learning classifiers exhibit superior sustainability compared to the YDataC approach. Specifically, the incrementally trained RF demonstrates better sustainability than the incrementally trained NN based on metrics such as f1 score and AUT (Area under Time).

In this research paper[15], a ML-based intelligent model is introduced to detect malware applications on smartphones using static malware analysis techniques. The model analyzes various features of Android applications, including permissions, intents, system commands, and API calls. These features are extracted from the application manifest file and source code. The proposed model is evaluated on two distinct datasets, namely DREBIN and MALGENOME. The experimental findings reveal that the RF classifier outperforms other algorithms in accurately identifying Android malware applications.

In this study[16], we propose a mechanism based on static features to detect Android malware using a static analyst approach. The mechanism analyzes static information such as permissions, component deployment, Intent message passing, and API calls to characterize the behavior of Android applications. To improve the capability of modeling malware with different intentions, various clustering algorithms can be applied. We have developed a system called Droid Mat that utilizes this mechanism. Droid Mat extracts information from the manifest file of each application, including requested permissions and Intent message passing. It considers components (Activity, Service, Receiver) as entry points and traces API calls related to permissions. The system then applies the K-means algorithm to enhance malware modeling, determining the number of clusters using the Singular Value Decomposition (SVD) method. Finally, it uses the kNN algorithm to classify applications as benign or malicious. Experimental results demonstrate that our approach has a higher recall rate compared to the well-known tool Androguard. Additionally, Droid Mat is efficient, taking only half the time of Androguard to predict the status of 1738 apps as either benign or Android malware. It achieves an accuracy of up to 97.87 percentage points.

In this research paper[17], we have introduced a malware detection system for the Android platform that relies on machine learning techniques. Our system utilizes a dataset containing both benign and malicious apps to train the classifiers. Unlike previous approaches, our system considers additional features such as requested permissions, vulnerable API calls, and the presence of dynamic code, reflection code, native code, cryptographic code, and database in the applications. These features are used in various machine learning classifiers to develop a classification model. To evaluate the effectiveness of our system, we conducted several experiments using a tool called "RanDroid". The results demonstrated that RanDroid achieved a high classification accuracy of 97.7 percent.

No.	Dataset	Samples	Feature Source	Algorithm	Result	Ref
1.	PRAGaurd Google Play	669 malware 652 benign	Permissions API calls	Logistic Regression	Accuracy = 0.9725	[18]
2.	MalGenome Official and third markets	1000 malware 1000 benign	API calls Resources Assets Libraries Permissions	Naïve Bayes	Accuracy = 0.921 TNR=0.937 FPR = 0.063 TPR = 0.906 FNR = 0.094 AUC = 0.9722 Precision =0.935	[19]
3.	Drebin	5559 malware 123,452 benign	Permissions Components API calls Network addresses	KNN	Accuracy = 0.9948	[20]
4.	Drebin Several public malware libraries Google Play	3986 malware 3986 benign	Permissions API calls	SVM KNN	TPR= 0.975 FPR= 0.032 Accuracy = 0.975 Recall=0.975	[21]
5.	Collected by the researchers	No malware 820 benign	Frequency of strings Permissions from app Permissions from markets	RF KNN SVM	AUC = 0.93	[22]
6.	DREBIN and MALGENOME.		permissions, intents, system commands, and API calls.	RF	---	[15]

II. Result:

In the survey conducted in Figure 2 it was concluded that there are numerous datasets used by researchers however most of the researchers were more inclined to use Derbin dataset and the second most popular dataset was Mal Genome other than that there were many datasets used from many other sources below is a figure showcasing the results of the survey:

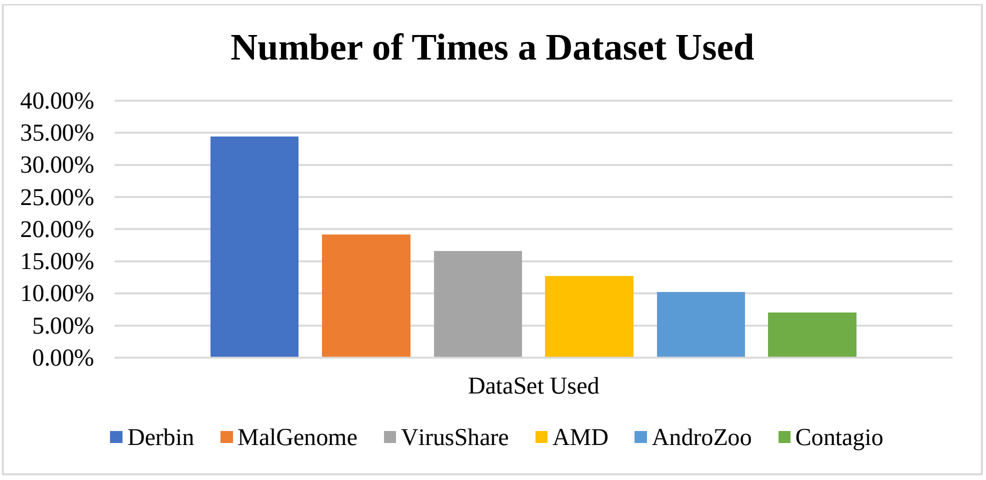

Figure 2 Dataset used for static Analysis

As shown in Figure 3, SVM were the most commonly employed models for detecting the Android Malware using ML. However, some researchers have used the other models like Random Forest, KNN, Decision Tree etc.

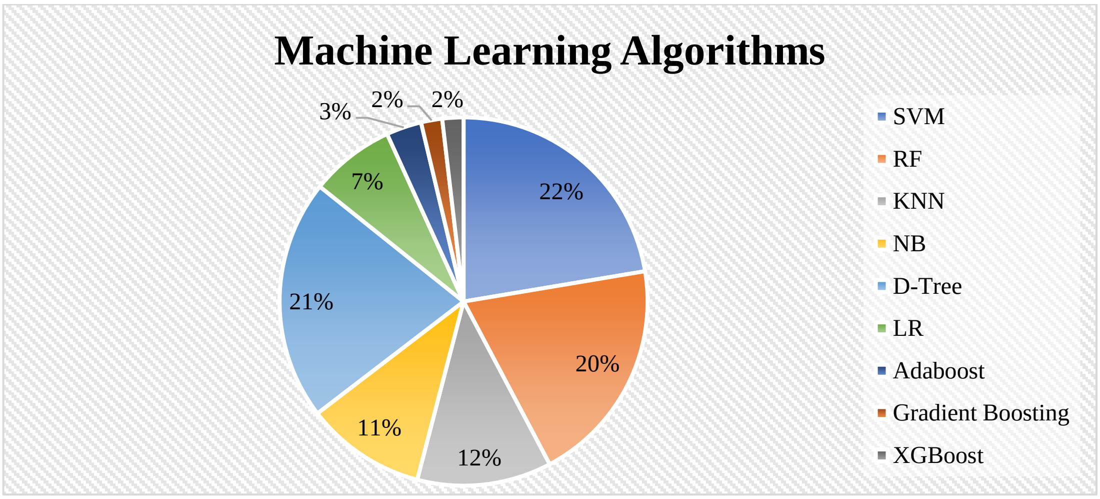

Figure 3 ML algorithms used for static analysis

III. Conclusion

The issue of Android malware has been brought into focus due to the rapid growth in the use of Android devices. This has resulted to the development of machine learning (ML)-based techniques for malware detection. These ML-based approaches have the potential to identify previously unseen malware, thus offering the ability to prevent zero day attacks. In this survey, the author thoroughly examined the use of static malware detection in Android malware apps. They compared the current techniques based on certain criteria. The study delved into Android app composition, analyzed the source of static features, reviewed machine learning-based static detection technology for Android malware, and discussed future development directions. The datasets and ML methods that were employed in the Android malware detection static analysis were also examined. Compared to other types of Android malicious application detection technology, the static detection method based on ML demonstrates advantages in terms of comprehensiveness, accuracy, and reduced dependence on experts, although it does have some weaknesses.

References:

[1]. Statista, "Statistic_Id330695_Smartphone-Mobile-Network-Subscriptions-Worldwide-2016-2028.Pdf." p. 1, 2021.
[2]. F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, "DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection," Appl. Sci., vol. 13, no. 13, 2023, doi: 10.3390/app13137720.
[3]. A. Rajagopal, "Incident Of The Week: Malware Infects 25M Android Phones ," Cyber Security Hub. 2019. [Online]. Available: https://www.cshub.com/malware/articles/incident-of-the-week-malware-infects-25m-android-phones
[4]. M. R. Amin, M. Zaman, M. S. Hossain, and M. Atiquzzaman, "Behavioral malware detection approaches for Android," 2016 IEEE Int. Conf. Commun. ICC 2016, pp. 1–6, 2016, doi: 10.1109/ICC.2016.7511573.
[5]. A. Shabtai, Y. Fledel, and Y. Elovici, "Automated static code analysis for classifying android applications using machine learning," Proc. - 2010 Int. Conf. Comput. Intell. Secur. CIS 2010, pp. 329–333, 2010, doi: 10.1109/CIS.2010.77.
[6]. M. Lindorfer, M. Neugschwandtner, and C. Platzer, "MARVIN: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis," Proc. - Int. Comput. Softw. Appl. Conf., vol. 2, pp. 422–433, 2015, doi: 10.1109/COMPSAC.2015.103.
[7]. C. Jarabek, D. Barrera, and J. Aycock, "ThinAV: truly lightweight mobile cloud-based anti-malware," ACSAC '12 Proc. 28th Annu. Comput. Secur. Appl. Conf., pp. 209–218, 2012, doi: 10.1145/2420950.2420983.
[8]. P. Raghuvanshi and J. P. Singh, "Android Malware Detection Using Machine Learning Techniques," 2022 Int. Conf. Comput. Sci. Comput. Intell., pp. 1117–1121, 2023, doi: 10.1109/csci58124.2022.00200.
[9]. P. P. K. Chan and W. K. Song, "Static detection of Android malware by using permissions and API calls," Proc. - Int. Conf. Mach. Learn. Cybern., vol. 1, pp. 82–87, 2014, doi: 10.1109/ICMLC.2014.7009096.
[10]. N. Peiravian and X. Zhu, "Machine learning for Android malware detection using permission and API calls," Proc. - Int. Conf. Tools with Artif. Intell. ICTAI, pp. 300–305, 2013, doi: 10.1109/ICTAI.2013.53.
[11]. A. Demontis et al., "Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection," IEEE Trans. Dependable Secur. Comput., vol. 16, no. 4, pp. 711–724, 2019, doi: 10.1109/TDSC.2017.2700270.
[12]. W. Wang, Z. Gao, M. Zhao, Y. Li, J. Liu, and X. Zhang, "DroidEnsemble: Detecting Android Malicious Applications with Ensemble of String and Structural Static Features," IEEE Access, vol. 6, no. c, pp. 31798–31807, 2018, doi: 10.1109/ACCESS.2018.2835654.
[13]. H. Zhang, S. Li, X. Wu, W. Han, and L. Wang, "An android malware detection approach using multi-feature fusion and TF-IDF algorithm," Proc. - 2021 IEEE 6th Int. Conf. Data Sci. Cyberspace, DSC 2021, pp. 629–634, 2021, doi: 10.1109/DSC53577.2021.00100.
[14]. W. Cho, H. Lee, S. Han, Y. Hwang, and S. J. Cho, "Sustainability of Machine Learning-based Android Malware Detection Using API calls and Permissions," Proc. - 2022 IEEE 5th Int. Conf. Artif. Intell. Knowl. Eng. AIKE 2022, pp. 18–25, 2022, doi: 10.1109/AIKE55402.2022.00009.
[15]. A. H. El Fiky, A. Elshenawy, and M. A. Madkour, "Detection of Android Malware using Machine Learning," 2021 Int. Mobile, Intelligent, Ubiquitous Comput. Conf. MIUCC 2021, pp. 9–16, 2021, doi: 10.1109/MIUCC52538.2021.9447661.
[16]. D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing," Proc. 2012 7th Asia Jt. Conf. Inf. Secur. AsiaJCIS 2012, pp. 62–69, 2012, doi: 10.1109/AsiaJCIS.2012.18.

[17]. J. D. Koli, “RanDroid: Android malware detection using random machine learning classifiers,” Int. Conf. Technol. Smart City Energy Secur. Power Smart Solut. Smart Cities, ICSESP 2018 - Proc., vol. 2018-January, pp. 1–6, 2018, doi: 10.1109/ICSESP.2018.8376705.

[18]. S. R. Tiwari and R. U. Shukla, “An Android Malware Detection Technique Using Optimized Permission and API with PCA,” Proc. 2nd Int. Conf. Intell. Comput. Control Syst. ICICCS 2018, no. Icirca, pp. 134–139, 2019, doi: 10.1109/ICCONS.2018.8662939.

[19]. Y. Suleiman, S. Sezer, G. McWilliams, and I. Muttik, “New Android malware detection approach using Bayesian classification,” Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, pp. 121–128, 2013, doi: 10.1109/AINA.2013.88.

[20]. G. Baldini and D. Geneiatakis, “A performance evaluation on distance measures in KNN for mobile malware detection,” 2019 6th Int. Conf. Control. Decis. Inf. Technol. CoDIT 2019, pp. 193–198, 2019, doi: 10.1109/CoDIT.2019.8820510.

[21]. K. Zhao, D. Zhang, X. Su, and W. Li, “Fest: A feature extraction and selection tool for Android malware detection,” Proc. - IEEE Symp. Comput. Commun., vol. 2016-Febru, pp. 714–720, 2016, doi: 10.1109/ISCC.2015.7405598.

[22]. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, “On the automatic categorisation of android applications,” 2012 IEEE Consum. Commun. Netw. Conf. CCNC’2012, pp. 149–153, 2012, doi: 10.1109/CCNC.2012.6181075.