A Comprehensive Review on Permissions-Based Android Malware Detection

Yash Sharma
ysharma2098@gmail.com

Delhi Technological University

Anshul Arora
Delhi Technological University

Research Article

Keywords: Android Security, Android Malware, Permissions Based Detection, Static Detection, Mobile Security, Literature Review.

Posted Date: November 29th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3657506/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at International Journal of Information Security on March 4th, 2024. See the published version at https://doi.org/10.1007/s10207-024-00822-2.

A Comprehensive Review on Permissions-Based Android Malware Detection

Yash Sharma and Anshul Arora†

[1]Department of Applied Mathematics, Delhi Technological University, Delhi, -110042, India.

*Corresponding author(s). E-mail(s): ysharma2098@gmail.com;
Contributing authors: anshul15arora@gmail.com;
†These authors contributed equally to this work.

Abstract

The first Android-ready "G1" phone debuted in late October 2008. Since then, the growth of Android malware has been explosive analogous to the rise in the popularity of Android. The major positive aspect of Android has been its open-source nature, which empowers app developers to expand their work. But at the same time, authors with malicious intentions pose grave threats to users. In the presence of such threats, Android malware detection is the need of an hour. Consequently, researchers have proposed various techniques involving static, dynamic, and hybrid analysis to address such threats using numerous features in the last decade. But the feature that most researchers have extensively used to perform malware analysis and detection in Android security is Android permission. Hence, to provide a clarified overview of the latest and past work done in Android malware analysis and detection, we perform a comprehensive literature review using permissions as a central feature or in combination with other components by collecting and analyzing 200 studies from January 2009 to February 2023. We extracted information such as the choice opted by researchers between analysis or detection, techniques used to select or rank the permissions feature set, features used along with permissions, detection models employed, the malware datasets used by researchers, and lastly, the limitations and challenges in the field of Android malware detection to propose some future research directions. Additionally, based on the information extracted, we answer the six research questions designed considering the above factors.

Keywords: Android Security, Android Malware, Permissions Based Detection, Static Detection, Mobile Security, Literature Review.

1 Introduction

In the last decade, we have witnessed the exponential growth of the Android operating system in the mobile market. According to a recent report, the Android system constitutes more than 80% of the whole market of smart phones[1]. The main reason behind Android's success is its free, open-source code, which empowers smartphone manufacturers to transform their devices with pre-installed applications and customized user interfaces for a beautiful customer experience. But Android's

[1] www.tenda.com.cn

open-source nature is a boon and a bane. On the one hand, where it brings the benefits of technological broadband and update, it also allows criminals to use it for ill practices. These days mobile phones are not only used for communication purposes, but gradually they have become a crucial part of our lives containing the smallest to the most critical and private user data. In such a situation, robust and effective Android detection mechanisms are the need of the hour.

Android OS was released in 2008, and just then, the first Android malware was spotted in 2010, which targeted users by subscribing to premium SMS services. Since then, malware attacks have been on the rise, and at the same time, security attempts have been fighting to keep up with the ever-increasing and constantly changing malicious attacks. The total number of Android malware worldwide have already increased from 22,088 in 2012 to 33,237,653 in Jan 2023[2]. Looking closer at the real-time threat analysis and statistics of Android malware worldwide, we will understand how desperately the Android Market needs Android security and malware detection systems.

Researchers and practitioners use various analysis and detection measures to address the above-mentioned concerning numbers. Based on the standard research techniques in the literature, Android malware detection is usually carried out using three analysis techniques: static, dynamic, and hybrid. The static analysis aims to investigate malware without executing the actual code but by collecting basic information about the app's functionality. In contrast, dynamic analysis performs and monitors an application to track its nature to find traces of malicious behavior. The working of hybrid analysis works in a way such that it combines the advantages of both static and dynamic analysis. Although static analysis poses some limitations while dealing with advanced malicious deformation techniques such as dynamic code loading, static analysis still proves to be quite efficient, usable, and scalable in prohibiting malware before execution. Amongst all the static features, the most popular and commonly used static feature is permissions. Google introduced a permission system for the Android OS, making it mandatory for all developers to define the necessary permissions required for the functionality of their product. It is up to the user to grant or deny access to the requested permissions during the installation. Hence, monitoring the usage of permissions before installing any application can prevent the spread of malware. It may seem challenging to a standard user, but the researchers have used the meaning, frequency, and combination of permissions requested over the past 13 years to build robust Android malware detection models.

Table I shows that this review offers much more than most previous works in this area. This review is not based on a general topic like [5], but instead focuses systematically on applying for permissions as a feature in Android Malware analysis/detection. Some surveys and reviews exist in the literature, such as [13] and [12], to analyze the work done in Android malware detection using analysis techniques, features, and machine learning models. A systematic literature review focusing on the trend of static analysis and the usage of machine learning models, respectively, was covered in the two surveys mentioned above. However, there has been a gap in the Android malware detection investigation in recent years. More specifically, a survey highlighting the popularity of Android permissions as a feature in Android malware analysis/detection is missing. Therefore, it is indispensable to summarize the related work in the field of Android malware detection using permissions from the advent of the first proposed Android malware-related model.

To have a clear view of the usage and popularity of permissions as a feature in Android malware analysis and detection over almost 14 years (2009-2023), we conducted this Comprehensive Literature Review (CLR) after analyzing the related work thoroughly. The main contributions of this work are highlighted below :

- We perform this CLR using a vast dataset of 200 research papers that aim to use permissions for Android malware analysis/detection, almost covering the advent of Android OS [14] and the first malware in 2009 to the current research scenario in the second month of 2023 [15].
- Apart from obtaining information about the usage and permission-based techniques, we

[2] https://portal.av-atlas.org/malware

Table 1: Comparison of recent reviews having similar coverage with this article.
(✓= Having content, x = having no or little content)

Reference	Feature Analysis	Technique used	Model used (ML/DL)	Analysis/ Detection	Methodology Comparison	Dataset used	Discussion on Limitations and Future Directions
[1]	✓	x	✓	x	x	x	✓
[2]	✓	x	✓	x	x	x	x
[3]	✓	x	✓	x	x	✓	x
[4]	✓	x	✓	x	x	✓	x
[5]	x	x	x	x	x	x	x
[6]	✓	x	x	x	x	x	x
[7]	x	x	✓	x	x	x	x
[8]	✓	x	✓	x	x	x	✓
[9]	x	x	✓	x	x	✓	✓
[10]	x	x	x	x	x	x	x
[11]	✓	x	x	x	✓	x	✓
[12]	✓	✓	✓	x	x	✓	✓
This article	✓	✓	✓	✓	✓	✓	✓

comprehensively analyzed other features combined with permissions, detection models, and datasets used.

- According to the results of empirical evidence, permissions prove to be quite informative and efficient as a means of detecting malware in Android smartphones.
- Finally, we make the discussions about the results, limitations and possible future work directions using permissions as a feature.

1.1 Review Protocol

In this section, we discuss in detail the main steps involved in conducting this CLR :

1. Research questions: After reviewing the related work and locating the different queries that need to be analyzed in this review, we put forward six research questions and their answers that further perform the basis of the following sections. Table 2 presents six research questions about the usage of permissions in Android malware detection.
2. Search Strategy: Our first step while building a comprehensive literature survey is collecting related work by various authors in Android security, mainly based on permissions. Our work revolves around the usage and impact of permissions in the area of Android malware detection; hence we chose the related work that has used permissions alone or combined with other static/dynamic features to analyze or detect malware in Android smartphones. We identified several search sources and search items to cover the related research between the period of 2009 to 2023 and, in the end, selected seven electronic databases, which include main journals and conferences, namely -
    - IEEE Xplore Digital Library
    - ScienceDirect
    - ACM Digital Library
    - Wiley Online Library
    - Google Scholar
    - SpringerLink
    - Web of Science
3. Data extraction and synthesis: The last step of our survey, which is the data extraction and synthesis, is directly related to the first step, i.e., the research questions. According to the information extracted from various research papers, we can find the answers to the research

Table 2: Research Questions

ID	Research Questions	Motivation
RQ1	What was the primary underlying purpose of using permissions defined by the researchers- behavioral analysis or malware detection?	Assess the goals of researchers and practitioners.
RQ2	What feature ranking, selection, or other techniques are used to build an Android malware analysis/detection system considering permissions?	Identify the commonly used method for selecting and extracting relevant features.
RQ3	Which features are primarily used in combination with permissions for Android malware analysis/detection?	Identify the commonly used features along with permissions.
RQ4	Which ML/DL/other models are used for Android malware analysis/detection?	Compare the popularity of ML and DL models.
RQ5	Which datasets are used for malware analysis/detection?	Identify the most famous experimental datasets.
RQ6	What are the limitations, challenges, and future directions for permissions-based Android malware analysis or detection?	Assess the limitations or challenges and consequently, propose future research directions.

questions and queries. We extracted the following information from all the research papers used for this review -

- Purpose of research - Choice between creating an Android malware detection model or analyzing benign and malware apps.
- Technique used - As described above, our focus relies upon permissions; hence we obtained information on how authors utilized permissions to build their malware analysis or detection model.
- Features used - Most common features chosen and utilized by authors while carrying out their research work in combination with permissions.
- Type of model used - Choice of model and classifiers preferred between Machine learning (ML) and Deep learning (DL) or any other kind if used for analysis or detection.
- Malware dataset used - What malware datasets are used by the authors in their corresponding works?
- Limitations - Limitations of detection/analysis models mainly associated with permissions as a feature in the field of Android security?
- Future Directions - Assessing the limitations and challenges faced by the permissions-based Android malware analysis/detection models, we propose some future research directions.

2 Purpose of Research

In response to RQ1, we present the data summarizing the choice preferred by authors in Table 3, between only analyzing the permissions in malware/benign apps or detecting malicious behavior in applications by proposing full-fledged malware detection using a dataset comprising of both benign and malware applications. The analysis process closely observes the behavioral pattern of features requested by applications. It could be performed using both applications classes [16] or even individual types too [17]. On the other hand, a research paper falls under the detection category if the corresponding authors have performed detection using a mixed or unlabelled dataset after completing the analysis or training phase.

Table 3 indicates that over the span of around 14 years, almost all the researchers and practitioners aimed to detect malware in Android smartphones using the datasets available in the market comprising both normal and malware applications. But some authors chose to carry out only the analysis of feature behavior; for instance, the authors in [18] analyzed descriptions of apps downloaded from the Google Play Store to predict the requested permissions but didn't use any malware applications. If we try to come to a conclusion using the data highlighted by Table 3 to understand the situation better, we can observe that a little over 93% of the work mentioned in the table revolved around malware detection using permissions and not only analyzing apps' permissions. **Hence, based on the results presented above, we answer the first Research Question that the majority of researchers have chosen the path of building permissions-based Android malware detection models instead of merely analyzing the permissions of malware.**

Table 3: Purpose of Research (Analysis or Detection)

Related works	Analysis		Malware Detection
	Normal	Malware	
Zhang et al. [19]	✓	✓	✓
Li et al. [20]	✓	✓	✓
Sahin et al. [21]	✓	✓	✓
Talha et al. [22]	✓	✓	✓
Varma et al. [23]	✓	✓	✓
Mahindru et al. [24]	✓	✓	✓
Dogru et al. [25]	✓	✓	✓
Rathore et al. [26]	✓	✓	✓
Shang et al. [27]	✓	✓	✓
Tchakounte et al. [28]	✓	✓	✓
Ju et al. [16]	✓	✓	
Ilham et al. [29]	✓	✓	✓
Sahin et al. [30]	✓	✓	✓
Angelo et al. [31]	✓	✓	✓
Xiong et al. [32]	✓	✓	✓
Lu et al. [33]	✓	✓	✓
Kavitha et al. [34]	-	-	-
E. Amer [35]	✓	✓	✓
Chakravarty et al. [36]	✓	✓	✓
Pondugula et al. [37]	✓	✓	✓
Sahal et al. [38]	✓	✓	✓
Tuan Mat et al. [39]	✓	✓	✓
Wang et al. [40]	✓	✓	✓
Park et al. [41]	✓	✓	✓
Liang et al. [42]	✓	✓	✓
Enck et al. [14]	✓	-	-
Enck et al. [17]	✓	-	-
Wang et al. [43]	✓	✓	-
Peng et al. [44]	✓	✓	-
Pandita et al. [45]	✓	-	-
Samra et al. [46]	✓	-	-
Yerima et al. [47]	✓	✓	✓
Aung et al. [48]	✓	✓	✓
Yerima et al. [49]	✓	✓	✓
Sanz et al. [50]	✓	✓	✓
Moonsamy et al. [51]	✓	✓	✓
Backes et al. [52]	✓	-	-
Wu et al. [53]	✓	✓	✓
Kato et al. [54]	✓	✓	✓
Arora et al. [55]	✓	✓	✓
Alsoghyer et al. [56]	✓	✓	✓
Saleem et al. [57]	✓	✓	-
Ghasempour et al. [58]	✓	✓	✓
Shrivastava et al. [59]	✓	✓	✓
Upadhayay et al. [60]	✓	✓	✓
Lee et al. [61]			
Surendran et al. [62]	✓	✓	✓
A. T. Kabakus [63]	✓	✓	✓
Wang et al. [64]	✓	✓	✓
Akbar et al. [65]	✓	✓	✓
Zhu et al. [66]	✓	✓	✓
Wang et al. [67]	✓	✓	✓
N. McLauglin [68]	✓	✓	✓
Wang et al. [69]	✓	✓	✓
Grace et al. [70]	✓	✓	✓
Liu et al. [71]	✓	✓	✓
Bayazit et al. [72]	✓	✓	✓
Lee et al. [73]	✓	✓	✓
Zhu et al. [74]	✓	✓	✓

3 Techniques used

In response to RQ2, this section aims to present the various feature selection or ranking or other similar techniques used by researchers to exploit permissions while building an analysis/detection model. Be it feature selection or feature ranking, researchers aim to select features highly dependent on the response. Various feature reduction techniques have been used in primary studies to determine and choose significant features in Android malware detection. But some authors have approached the detection process in different ways too. Hence, we divided all the techniques into five categories: feature ranking, frequent patterns-based, graph-based approach, feature selection, and others. Some of the most commonly used methods are discussed in brief below.

(a) Frequency-based techniques - Frequency-based techniques generally fall under the

Related works	Analysis		Malware Detection
	Normal	Malware	
Almahmoud et al. [75]	✓	✓	✓
Feng et al. [76]	✓	✓	✓
Kandukuru et al. [77]	✓	✓	✓
Arora et al. [78]	✓	✓	✓
Ding et al. [79]	✓	✓	✓
Sahin et al. [80]	✓	✓	✓
Idrees et al. [81]	✓	✓	✓
Khariwal et al. [82]	✓	✓	✓
Idrees et al. [83]	✓	✓	✓
Zhu et al. [15]	✓	✓	✓
Bai et al. [84]	✓	✓	✓
Taheri et al. [85]	✓	✓	✓
Alazab et al. [86]	✓	✓	✓
Mathur et al. [87]	✓	✓	✓
Imtiaz et al. [88]	✓	✓	✓
Liu et al. [89]	✓	✓	✓
Chen et al. [90]	✓	✓	✓
Guan et al. [91]	✓	✓	✓
Mohamed et al. [92]	✓	✓	✓
Varma et al. [93]	✓	✓	✓
Gyunka et al.[94]	✓	✓	✓
Taha et al. [95]	✓	✓	✓
Peng et al. [96]	✓	✓	✓
Ashwini et al. [97]	✓	✓	✓
Jiang et al. [98]	✓	✓	✓
Wang et al. [99]	✓	✓	✓
Rana et al. [100]	✓	✓	✓
Lu et al. [101]	✓	✓	✓
Millar et al. [102]	✓	✓	✓
Barrera et al. [103]	✓	-	-
Shabtai et al. [104]	✓	-	-
Felt et al. [105]	✓	-	-
Erickson et al. [106]	✓	-	-
Sarma et al. [107]	✓	✓	✓
Frank et al. [108]	✓	-	-
Jhu et al. [109]	✓	✓	✓
Peiravian et al. [110]	✓	✓	✓
Sanz et al. [111]	✓	✓	✓
Feldman et al. [112]	✓	✓	✓
Pehlivan et al. [113]	✓	✓	✓
Rahman et al. [114]	✓	✓	✓
Rovelli et al. [115]	✓	✓	✓
Arp et al. [116]	✓	✓	✓
Yerima et al. [117]	✓	✓	✓
Kang et al. [118]	✓	✓	✓
Zhao et al. [119]	✓	✓	✓
Qiao et al. [120]	✓	✓	✓
Chen et al. [121]	✓	✓	✓
Demertzis et al. [122]	✓	✓	✓
Verma et al. [123]	✓	✓	✓
Wang et al. [124]	✓	✓	✓
Tangil et al. [125]	✓	✓	✓
Wang et al. [126]	✓	✓	✓
Li et al. [127]	✓	✓	✓
Bhattacharya et al. [128]	✓	✓	✓
Xie et al. [129]	✓	✓	✓
Xie et al. [130]	✓	✓	✓
Ren et al. [131]	✓	✓	✓
Tao et al. [132]	✓	✓	✓
Namrud et al. [133]	✓	✓	✓
Alswaina et al. [134]	✓	✓	✓
Qiu et al. [135]	✓	✓	✓
Zhu et al. [136]	✓	✓	✓
Feng et al. [137]	✓	-	-

Related works	Analysis		Malware Detection
	Normal	Malware	
Aonzo et al. [138]	✓	✓	✓
Urooj et al. [139]	✓	✓	✓
Wang et al. [140]	✓	-	-
Wang et al. [141]	✓	✓	✓
Zhang et al. [142]	✓	-	-
Kesswani et al. [143]	✓	-	-
Ibrahim et al. [144]	✓	✓	✓
Arshad et al. [145]	✓	✓	✓
Yuan et al. [146]	✓	✓	✓
Zhou et al. [147]	✓	✓	✓
Cilleruelo et al. [148]	✓	✓	✓
Firdaus et al. [149]	✓	✓	✓
Wang et al. [150]	✓	✓	✓
Singh et al. [151]	✓	✓	✓
Rafiq et al. [152]	✓	✓	✓
Mahdavifar et al. [153]	✓	✓	✓
Seraj et al. [154]	✓	✓	✓
Mahindru et al. [155]	✓	✓	✓
Sahin et al. [21]	✓	✓	✓
Anupama et al. [156]	✓	✓	✓
Chen et al. [157]	✓	✓	✓
Mahindru et al. [158]	✓	✓	✓
Tchakounté et al. [159]	✓	✓	✓
Nissim et al. [160]	✓	✓	✓
Peynirci et al. [161]	✓	✓	✓
Nauman et al. [162]	✓	✓	✓
Bhattacharya et al. [163]	✓	✓	✓
Bao et al.[164]	✓	-	-
Medrano et al. [165]	✓	-	-
Mat et al. [165]	✓	✓	✓
Shatnawi et al. [166]	✓	✓	✓
Smmarwar et al. [167]	✓	✓	✓
Arif et al. [168]	✓	✓	✓
Manzanares et al. [169]	✓	✓	-
Bhat et al. [170]	✓	✓	✓
Elayan et al.[171]	✓	✓	✓
Syrris et al. [172]	✓	✓	✓
Idrees et al. [173]	✓	✓	✓
Rehman et al. [174]	✓	✓	✓
Martin et al. [175]	✓	✓	✓
Navarro et al. [176]	✓	✓	✓
Milosevic et al. [177]	✓	✓	✓
Alzaylaee et al. [178]	✓	✓	✓
Cai et al. [179]	✓	✓	✓
Badhani et al. [180]	✓	✓	✓
Hijawi et al. [181]	✓	✓	✓
Sheen et al. [182]	✓	✓	✓
Nisha et al. [183]	✓	✓	✓
Song et al. [184]	✓	✓	✓
Zhang et al. [185]	✓	✓	✓
Yang et al. [186]	✓		-
Thiyagarajan et al. [187]	✓	✓	✓
Qaisar et al. [188]	✓	✓	✓
Appice et al. [189]	✓	✓	✓
Zhu et al. [190]	✓	✓	✓
A. Altaher [191]	✓	✓	✓
Su et al. [192]	✓	✓	✓
Mahindru et al. [193]	✓	✓	✓
Dehkordy et al. [194]	✓	✓	✓
Nguyen et al. [195]	✓	✓	✓
Taheri et al. [196]	✓	✓	✓

Related works	**Analysis**		**Malware Detection**
	Normal	**Malware**	
Mahesh et al. [197]	✓	✓	✓
Firdaus et al. [198]	✓	✓	✓
Shrivastava et al. [199]	✓	✓	✓
Varsha et al. [200]	✓	✓	✓
M. Deypir [201]	✓	✓	✓
Mahindru et al. [202]	✓	✓	✓
Keyvanpour et al. [203]	✓	✓	✓
Razak et al. [204]	✓	✓	✓
Xie et al. [205]	✓	✓	
Mahindru et al. [206]	✓	✓	✓
Alecakir et al. [18]	✓	-	-
Ali et al. [207]	✓	-	-
Sun et al. [208]	✓	✓	✓
AlJarrah et al. [209]	✓	✓	✓
Gharib et al. [210]	✓	✓	✓
Sun et al. [211]	✓	✓	✓

category of feature ranking techniques; however, they can sometimes be used as a feature selection technique too. The main underlying concepts that these techniques tend to exploit are -

- Some features are frequently requested by only one class of dataset, either benign or malware, and as the goal is generally to differentiate between normal and malicious applications, the features that are frequently requested only by the malware applications are considered dangerous, and the ones having a high frequency in the case of normal applications are considered to have low-risk factor [58].
- Some features are commonly used by both the classes, benign and malware, and hence they can be excluded to choose only the more informative features. For instance, permissions such as "INTERNET" are frequently requested by both malware and benign apps; hence the authors in [20] chose to eliminate such permissions as they might introduce ambiguity in the malware detection process. Moreover, they ranked the features based on their frequency of usage in malicious and benign apps.
- Apart from the two approaches mentioned above, the authors also focus on how often one permission or feature is repeated in the whole dataset or has a low support value. In such cases also, the frequency is used as a parameter to rank and further select only the relevant features.

(b) Information Gain (Information Gain) - Information Gain can be defined as a measure of reduction in entropy. One can understand it as a measure of reduction in the amount of information upon splitting a dataset according to a certain value of a random variable. Information Gain is inversely proportional to entropy, i.e., the higher the information gain, the lower the entropy of that particular group, as the element of surprise would be less. Mathematically, Information Gain is calculated by comparing the entropy of the dataset before and after a transformation. Common usage of Information Gain includes forming decision trees from a training dataset. Information Gain is calculated for each variable in the first step, followed by selecting the variable with the maximum Information Gain value and thus minimizing the entropy and obtaining the best splits of the dataset for efficient classification.

As feature ranking is a subset of feature selection, Information Gain can be used as both techniques, i.e., to reduce the dataset and simultaneously choose only the most informative features. This is done by calculating the gain of each variable in the context of the target variable. Information Gain value for each independent attribute is calculated and further ranked from top to bottom, the top being the most relevant with the highest Information Gain score. After this, a threshold could be decided to filter out only the features with Information Gain values above the threshold, which can be further included in the machine or deep learning classifiers.

(c) Principal Component Analysis (PCA) - Another commonly used feature selection technique in Android malware detection is Principal Component Analysis, which works on reducing the number of variables while preserving as much information as possible. PCA identifies the correlations to calculate the eigenvectors and eigenvalues, which identifies the principal components in return. After that, one can choose which principal components to keep and which ones to discard by creating a feature vector.

(d) Chi-square - Chi-square is generally used in statistics to test the independence of two events, but quite often, it has also been used as a feature selection technique in Android malware detection to reduce the size of the feature set. Mathematically, Chi-square measures how expected count E and observed count O deviate from each other. The Chi-square formula [212] is defined in the below equation.

$$\chi_c{}^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:
c=Degrees of freedom,
O=Observed value(s), and
E=Expected value(s)

While aiming for feature selection, the features having higher chi-square values are selected for model building as a higher chi-square value depicts higher dependence over the target variable or the response.

(e) Clustering - Clustering is another technique that has been extensively used in Android malware detection first to understand a dataset and further club the similar and less informative features. In terms of machine learning, unlabelled grouping examples are called clustering.

In response to RQ2, we conclude by the information depicted in Table 4 that the most commonly used techniques involving the use of permissions are Information Gain and frequency-based, e.g., [58] and [20]. Apart from these, most researchers based their approach around utilizing either ML or DL classifiers by tuning the hyperparameters, e.g., [196] and [171]. Figure 1 gives a better and closer understanding of Table 4. As we can see, close to 48% of the total research works used in this review based their approach on either ML or DL classifiers/techniques [110]-[115]. Either they fed the extracted features directly to the machine learning classifiers to be dealt with or used techniques like gain ratio, correlation coefficient [29], mutual information, relief [36]

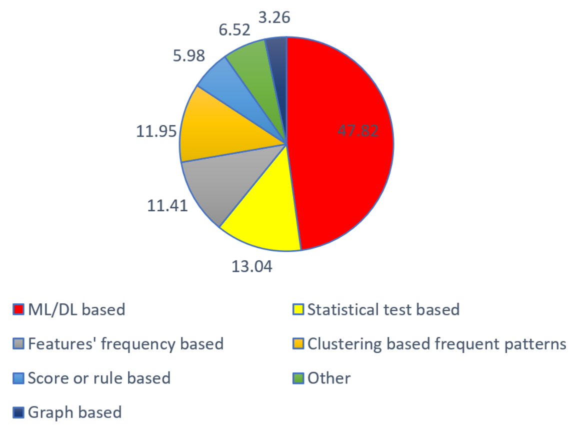

Fig. 1: Statistics depicting the most commonly used techniques to build an Android malware analysis/detection system considering permissions

etc., to compute the feature score. The second most common approach used by researchers and practitioners to reduce or choose the best set of features is utilizing the statistical tests (13.04%) such as Chi-square [119], PCA [34], Mann–Whitney test [132], variance threshold [26], ANOVA [120] and many more. Around 11% of the total work account for approaches based upon features' frequency [19]-[20] and clustering-based frequent patterns [46] each. 5.98% papers utilised some sort of similarity score [22] or rule based techniques [17] to analyse or detect Android malware. Lastly, 6.52% and 3.26% of the comprehensive studies used graph-based [185], [186], and some other techniques, respectively. For instance, in [31], the authors mapped the permissions on the x-y plane using their corresponding protection level, whereas in [45], they utilized app descriptions to deduce the permissions required by an application. **Hence, based on the results presented above, we answer the second Research Question that popular techniques to handle the extracted permissions and other features are the ML and DL classifiers as their hyper parameters can be easily tuned to reduce, rank, or select the unlabelled or unbalanced feature set.**

Table 4: Technique used

Related works	**Feature Ranking**	**Frequent Patterns**	**Feature Selection**	**Graph based**	**Others**
Zhang et al. [19]		Monitored the permission usage within the application and with the system			
Li et al. [20]	Ranking based on the frequency of permissions being requested.				
Sahin et al. [21]			Linear Regression		
Talha et al. [22]					Combined Risk score calculated for each app
Varma et al. [23]					Used permissions as features to study the performance of ML algorithms
Mahindru et al. [24]			Discarded the ones not installed or starting at launching stage		
Dogru et al. [25]					Permission groups score calculated, to sum up an app's Risk Score
Rathore et al. [26]			Variance threshold, autoencoders, and PCA		
Shang et al. [27]			Reduced the permission set with Pearson's Correlation Coefficient		
Tchakounte et al. [28]		Similarity score based on sequence alignment			
Ju et al. [16]		Manual pattern recognition to existing malware permissions patterns			
Ilham et al. [29]			Gain Ratio, Information Gain, Correlation Coefficient, CFS subset Evaluator		
Sahin et al. [30]					Relevance Frequency
Angelo et al. [31]					Mapped the permissions on the x-y plane using their corresponding protection level
Xiong et al. [32]		Used unique and common permissions patterns from both datasets as weak Classifier			
Lu et al. [33]			Improved RF algorithms along with introducing fuzzy sets of samples		
Kavitha et al. [34]			PCA and Sequential Forward selection, Limiting the permissions by accepting or denying each permission separately according to Dangerous level		
E. Amer [35]					Developed an ensemble comprising multiple classifiers
Chakravarty et al. [36]			Information Gain, Relief, Gain Ratio		
Pondugula et al. [37]					Deep Neural Network model

Sahal et al. [38]	Ranking based on permissions class frequency				
Tuan Mat et al. [39]			Used Bayes classifier after optimising features using Chi-Square Test		
Wang et al. [40]			Association rule Mining, PCA , Deep Cross Network		
Park et al. [41]			Reduced features by removing built-in, custom, dangerous, and permissions that are used at least once		
Liang et al. [42]		Generated k maps for permission combinations based on their usage			
Enck et al. [14]					Presented analysis of the newly launched Android OS in 2009
Enck et al. [17]		Defined rules based upon dangerous level and possible negative impact			
Wang et al. [43]		Calculated the risk score of each app using Baye's rule based upon frequency of permissions and risk levels			
Peng et al. [44]					Calculated the risk scores of applications using probabilistic methods like Naïve Bayes and its modifications
Pandita et al. [45]					Introduced a framework based upon natural language processing called "WHYPER" which used application descriptions to describe which permissions are needed and why
Samra et al. [46]		Permissions are used as features to make clusters using K-means cluster algorithm			
Yerima et al. [47]	Permissions are used as features along with others to be fed into the Bayesian Classifier				
Aung et al. [48]			K best features selected using Information Gain		
Yerima et al. [49]	Permissions are used as features along with others to be fed into the Bayesian Classifier				
Sanz et al. [50]					Work based upon permission frequency
Moonsamy et al. [51]			Biclustering method to visualize for rare, unique as well as frequent patterns, followed by reducing permissions based on their support difference between normal and malicious datasets		

Backes et al. [52]					Appguard, a powerful system capable of modifying user-defined security policies on untrusted applications and converting them into new trusted ones
Wu et al. [53]		Used permissions, and other information to form a feature set, further applied K-means and EM algorithm to form clusters for malware detection			
Kato et al. [54]					Calculated similarity score between malware and normal permission pairs after dividing them into different categories
Arora et al. [55]				Used permission pairs to construct graphs for normal and malware apps	
Alsoghyer et al. [56]					Malware detection model based on the frequency of permissions occurrences followed by using machine learning algorithm to assess the model
Saleem et al. [57]					Calculated the first four moments for permission's binary data after using Kernel Density Estimation and used the varying values as a means to distinguish between applications
Ghasempour et al. [58]	Permissions ranked by using frequency-based weighting method		PCA followed by using a statistical method based upon eigenvalues and eigenvectors		
Shrivastava et al. [59]					Risk score was calculated based on permissions frequency for each permission which in turn was used to classify applications as high risk, medium risk, and low risk
Upadhayay et al. [60]	Ranked the permissions based on the frequency of their occurrence in the datasets and combined them with the best network traffic features				
Lee et al. [61]					Classified normal and malicious apps through ML-based detection techniques based on the frequency of permission of Android apps
Surendran et al. [62]					Fed into the Logistic Regression (LR) classifier first followed by applying the naïve bayes classifier to find interdependency between features
A. T. Kabakus [63]					Fed into the Convolutional Neural network (CNN) model as one-dimensional input to form a training model

Wang et al. [64]			Decision Tree, Extra trees, Chi-square test, Genetic algorithm, SVM based on recursive feature elimination, MI		
Akbar et al. [65]			Employed Random Forest to generate the feature importance and combined it with Google's dangerous permissions and the features used in [66] to select the best set of permissions		
Zhu et al. [66]			Used TF-IDF and Cosine similarity to choose the best features		
Wang et al. [67]			Fed the permissions data combined with the API sequence data into the LR model		
N. McLauglin [68]					Used permissions as features in PerceiverIO blocks to be combined with the opcode PerceiverIO blocks
Wang et al. [69]	Ranked using Mutual Information (MI), correlation coefficient and T-test				
Grace et al. [70]		Risk level on an unknown application is decided by checking for dangerous permission requests			
Liu et al. [71]		Sensitive permission patterns are extracted	Filtered out using improved FP growth, removing clusters having same support and low JARO distance followed by performing hierarchical clustering		
Bayazit et al. [72]					Fed as input into the Recurrent neural Networks (RNN)-based classifiers
Lee et al. [73]			Information Gain and genetic algorithm		
Zhu et al. [74]					Fed into deep learning hybrid methods such as unsupervised method Merged Sparse Auto-Encoder (MSAE) and supervised method Stacked Denoising Auto-encoders (SDAE) and fed the extracted results to SVM and KNN
Almahmoud et al. [75]	Ranked on the basis of cosine similarity				
Feng et al. [76]			Chi-square test and extremely randomized tree method		
Kandukuru et al. [77]					Computed the permission's score using Jaccard-bitwise similarity technique to compare an app's risk score with the threshold

Arora et al. [78]		Combined the permissions and network traffic feature values and generated the frequent patterns using FP- Growth algorithm			
Ding et al. [79]			Chi-square test, analysis of variance (ANOVA) F-value, and MI		
Sahin et al. [80]					Fed into linear regression based classifiers
Idrees et al. [81]		Distinguishing frequency pattern ranges			
Khariwal et al. [82]	Information Gain				
Idrees et al. [83]		Constructed a detection matrix			
Zhu et al. [15]					Fed into Multi-Head Squeeze-and-Excitation Residual block (MSer), and stacked it to construct a deep network MSerNet
Bai et al. [84]			Fast Correlation-Based Filter by [213]		
Taheri et al. [85]			Random Forest Algorithm		
Alazab et al. [86]			Information Gain		
Mathur et al. [87]			Frequency counting, backward elimination and collinearity check		
Imtiaz et al. [88]					Fed into a deep learning artificial neural network classifier
Liu et al. [89]			Fed into RBM combined with subspace methods to reduce the feature dimensionality after choosing the best subspaces by clustering techniques		
Chen et al. [90]			PCA		
Guan et al. [91]			k-means clustering algorithm		Fed the results to Synthetic Minority Over-Sampling Technique (SMOTE)
Mohamed et al. [92]			Chose the most common features		
Varma et al. [93]			Bat Optimization, Cuckoo Search, and Grey Wolf Optimization wrapper feature selection techniques		
Gyunka et al.[94]			PCA		
Taha et al. [95]		Clustering similar Permissions			
Peng et al. [96]			Adaptive shrinkage CNN		
Ashwini et al. [97]					Fed the features to machine learning classifiers
Jiang et al. [98]	The dangerous features backtracked from sensitive API calls are ranked in order of Information Gain score				

Wang et al. [99]					Fed to various classifiers to create an ensemble classifier based upon selective Ensemble method and genetic algorithm
Rana et al. [100]			Selected only the most useful feature word by creating a dictionary		
Lu et al. [101]					Fed the permissions to DBN classifier
Millar et al. [102]					Fed the permissions to a CNN classifier
Barrera et al. [103]		Found permissions usage pattern			Fed to the SOM to analyze permissions usage pattern
Shabtai et al. [104]	Information Gain, Fisher Score and Chi-Square				
Felt et al. [105]					Compared permissions required to invoke API methods and actually requested permissions to check for over privilege issues
Erickson et al. [106]					Mapped permissions required to invoke API methods to check for privacy leaks
Sarma et al. [107]		Considered the permission patterns from their malware dataset and some critical most requested permissions to generate risk signals for the users			
Frank et al. [108]		To extract statistically significant permission request patterns			
Jhu et al. [109]					To compare the permissions required by an application by analyzing the description given by the developer with its actual requested permissions
Peiravian et al. [110]					Fed to various machine learning classifiers
Sanz et al. [111]					Fed to various machine learning classifiers
Feldman et al. [112]					Fed to various machine learning classifiers
Pehlivan et al. [113]			Gain Ratio Attribute Evaluator, Relief Attribute Evaluator, Cfs Subset Evaluator, Consistency Subset Evaluator		
Rahman et al. [114]					Fed to various machine learning classifiers
Rovelli et al. [115]					Fed to various machine learning classifiers on the server side component
Arp et al. [116]		Mapped to a joint vector space, where patterns were analyzed geometrically			
Yerima et al. [117]					Fed to various machine learning classifiers

Kang et al. [118]		Calculated the likelihood ratio under the given distribution of permissions and further used the Needleman-Wunsch algorithm to calculate similarity score			
Zhao et al. [119]			Chi-square and Information Gain		
Qiao et al. [120]			ANOVA and SVM—Recursive Feature Elimination		
Chen et al. [121]					Fed to various machine learning classifiers
Demertzis et al. [122]					Fed to ELM classifier and eSNN along with various hardware components
Verma et al. [123]					Extracted the functional call graph of the applications, used a procedure inspired by the neighborhood hash graph kernel (NHGK) and create a graph classification problem
Wang et al. [124]	Absolute permission rate difference				Feature vector was created composed of permissions, receiver actions, and hardware components
Tangil et al. [125]	Extra Tree algorithm and rank them by mean decrease impurity				
Wang et al. [126]					*FrequenSel* [119] and Information Gain
Li et al. [127]			PCA Method based on singular value decomposition (SVD)		
Bhattacharya et al. [128]	Information Gain				
Xie et al. [129]					Fed the results of syntax and semantic features to the machine learning classifiers
Xie et al. [130]					Feature vectors were reshaped to matrices by embedding to feed to CNN classifier
Ren et al. [131]					Fed the features to various machine learning and ensemble classifiers
Tao et al. [132]			Mann–Whitney test to analyze statistical significance of permission usage in both the datasets		
Namrud et al. [133]		Analysed the permissions usage pattern w.r.t different categories by using a combination of SOM and K-means			
Alswaina et al. [134]			Extremely Randomized Trees		

Qiu et al. [135]					Created a Feature vector table with the help of TF-IDF technique to form binary type vectors with security/privacy-related capabilities as annotations
Zhu et al. [136]					Fed the extracted features to an ensemble classifier of MLP and SVM as fusion classifier
Feng et al. [137]					App descriptions preprocessed by NLP methods are fed into the Neural network model to produce binary probability distribution for each permission
Aonzo et al. [138]					Fed the features to their own proposed linear and nonlinear classifiers
Urooj et al. [139]					Fed the features to various machine learning and ensemble-based classifiers
Wang et al. [140]					Processed the user reviews by using permissions docs, API docs, and app descriptions to infer the permissions required by an application
Wang et al. [141]					Fed the extracted features to various machine learning classifiers
Zhang et al. [142]					Utilised Whole call graphs (WCG) and parsed the mapping list to locate sensitive operations, form User-aware Call Graph, and perform static analysis
Kesswani et al. [143]		Divided the permissions under generic and privacy-invasive categories to further calculate the percentage of generic and privacy-invasive permissions of an unknown application			
Ibrahim et al. [144]					Fed to the deep learning classifier, in particular to embedding layers, followed by clustering and flattening layers to make their shape appropriate
Arshad et al. [145]					Fed the static and dynamic features to a machine learning classifier
Yuan et al. [146]					Fed the static and dynamic features to a deep learning classifier
Zhou et al. [147]					Fed the extracted features to deep learning based classifier
Cilleruelo et al. [148]			Reduce the multiple permissions that are the same but present differences based on the application package		

Firdaus et al. [149]			Generic Search		
Wang et al. [150]		Support-based permission candidate method to mine unique required or used permission patterns			
Singh et al. [151]			BI-Normal Separation (BNS), MI, Relevancy Score (RS), and the Kullback-Leibler (KL)		
Rafiq et al. [152]					Fed the extracted features to machine learning classifiers tuned by using NIAs
Mahdavifar et al. [153]					Fed the extracted features to a deep learning classifier
Seraj et al. [154]					Fed the extracted permissions to a MLP neural network classifier
Mahindru et al. [155]			Chi-Square, Gain Ratio, Filtered Subset selection, Information Gain feature, LR analysis, PCA		
Sahin et al. [21]			Relevance frequency feature selection (RFFS), Document frequency thresholding (DF), Information Gain, Chi-square, Odds ratio (OR), IDF and other filter-based methods		
Anupama et al. [156]	Fischer Score				
Chen et al. [157]					Fed the extracted permissions after removing the useless ones to RF machine learning classifier
Mahindru et al. [158]	Chi-squared test, Information Gain feature evaluation, LR analysis, Information Gain, oneR feature evaluation, PCA		T-test, Pearson's correlation Coefficient, Rough set analysis (RSA), Consistency subset evaluation approach, Filtered subset evaluation		
Tchakounté et al. [159]					Permissions and other features are utilized to form fuzzy-hashed signatures of known malware to find similarity score between them and unknown applications
Nissim et al. [160]					Fed the extracted features to SVM and a couple of proposed active learning methods for detection after processing
Peynirci et al. [161]			*Delta IDF* based upon differential inverse document frequency (IDF) values		
Nauman et al. [162]					Fed the extracted features to various Deep learning and machine learning classifiers
Bhattacharya et al. [163]			Improvised Particle Swarm optimization (PSO) algorithm for a rough set with a new random key encoding method		

Bao et al.[164]		Traced the API - permission patterns to predict the permissions for an app by using API			Fed into the naive Bayes multinomial classification model for text classification and predicting permissions for an application
Medrano et al. [165]		Taint tracking to analyze and find mappings between Android Class Function (ACFs) and the Permissions			
Mat et al. [165]			Information Gain and chi-square		
Shatnawi et al. [166]	LR model		Recursive Feature Elimination (RFE)		
Smmarwar et al. [167]			Binary Grey Wolf Optimization (BGWO)		
Arif et al. [168]			Information Gain		
Manzanares et al. [169]					Extracted permissions and other static and dynamic features to prepare a comprehensive dataset
Bhat et al. [170]	Information Gain		Deleted the features that are too infrequent and the ones that are present in almost the same number in both the datasets		
Elayan et al.[171]					Fed the extracted results to various machine classifiers
Syrris et al. [172]			Removed the features having low variance		
Idrees et al. [173]	Information Gain				Found the correlation between permissions and intents using Pearson correlation coefficient
Rehman et al. [174]					Fed the results to various machine learning classifiers to find the cosine similarity between features
Martin et al. [175]					Extracted to create a comprehensive and complete dataset, then fed the results to various machine learning classifiers for detection
Navarro et al. [176]	RF feature importance		Eliminated the linearly dependent vectors while performing the Bag of Graphs (BoG) technique	Formed Ontology-based graphs to find the relationship between permissions defined, used, interfaces protected by permissions and the resources access through them	

Milosevic et al. [177]					Fed into various machine learning and ensemble classifiers for classification and clustering
Alzaylaee et al. [178]	Information Gain				
Cai et al. [179]	Information Gain				
Badhani et al. [180]			Removed features having constant value or zero variance		Fed the features to various machine learning and ensemble classifiers. Further applied k-mode algorithm for clustering of features
Hijawi et al. [181]	Fed the features to various machine learning classifiers to rank them on the basis of their importance				
Sheen et al. [182]	Chi-Square, Relief, Information Gain				
Nisha et al. [183]			PSO, Social Spider Algorithm (SSA), and Gravitational Search Algorithm (GSA)	Represented permissions in the form of ontology graphs to understand the relationship between permissions, packages and interface classes	
Song et al. [184]		Dangerous permissions are matched with the permissions requested by an unknown application to generate a detection report and submit it to users			
Zhang et al. [185]				Identified explicit and implicit permission use points to further create permission use graphs to analyze the permissions behavioral pattern	
Yang et al. [186]				Constructed State Transition Graphs (STG) from permissions to implement breadth-first search (BFS) in the dynamic exploration phase to analyze the permission behavior	

Thiyagarajan et al. [187]	Pruning on the basis of frequency		Chi-square, Support based, association-based, PCA		
Qaisar et al. [188]			PCA, Co-relation attribute evaluation (CAE)		Constructed clusters using k-means algorithm to match up with the newly added features in the case base
Appice et al. [189]					Clustering based k- means ++ algorithm to form separate clusters for each view which were combined later using stacking-based fusion method to learn the consensus malware detection pattern
Zhu et al. [190]					Fed the results to RF-based machine learning classifier
A. Altaher [191]	Information Gain ratio				Fed the extracted permissions to an evolving fuzzy neuro inference classifier involving clustering methods
Su et al. [192]		Fed the results to a DBN learning model to obtain unique behavioral characteristics			
Mahindru et al. [193]	Chi-squared test, Information Gain feature evaluation, logistic regression analysis, Information Gain, oneR feature evaluation, PCA		T-test, Pearson's correlation Coefficient, RSA, Consistency subset evaluation approach, Filtered subset evaluation		
Dehkordy et al. [194]	Ranked features on the basis of frequency in both the datasets and removed the common or rarely used or irrelevant features				Balanced the dataset using SMOTE, Random undersampling, and Hybrid
Nguyen et al. [195]	ANOVA		DNN, SVM		
Taheri et al. [196]	Random Forest Regressor algorithm				Fed the results to a C4N and Robust-NN classifier
Mahesh et al. [197]			Minmax technique		Fed the preprocessed results to CNN-ARFO proposed classifier
Firdaus et al. [198]	Information Gain, frequency-based top range selected				Fed the results bio-inspired versions of ANN
Shrivastava et al. [199]			Defined frequency based rules defined over permissions and intents to predict malicious risk level of an application		
Varsha et al. [200]	NB, weight calculation		Entropy based Category Coverage Difference (ECCD) and Weighted Mutual Information (WI)		
M. Deypir [201]	Ranked the permission based on the entropy and Information Gain score calculated by the proposed method				Entropy-based method to calculate information gain score for permission and correspondingly risk score for an application

Mahindru et al. [202]	Chi-squared test, Information Gain feature evaluation, logistic regression analysis, Information Gain, oneR feature evaluation, PCA		T-test, Pearson's correlation Coefficient, RSA, Consistency subset evaluation approach, Filtered subset evaluation		
Keyvanpour et al. [203]			Eliminated features having frequency count and RF weigh below the standard deviation, frequency counts over groups of features		
Razak et al. [204]			Particle Swarm optimization (PSO), evolutionary computation, Information Gain		
Xie et al. [205]	Frequency based		Fisher score		Fingerprinted Android malware families based on top permission behavioral patterns
Mahindru et al. [206]			T-test, multivariate linear regression stepwise forward selection and cross-correlation		Fed the extracted features to various machine learning and ensemble classifiers
Alecakir et al. [18]					App descriptions are fed into neural network classifiers to learn and analyze the sentences and predict the permissions requested
Ali et al. [207]					Fed the permissions and other information to the SVM classifier to learn and predict preferred application permissions to its user
Sun et al. [208]					Took keywords from permissions, API calls, intents, etc as parameters and found the Keywords Correlation Distance (KCD) between them to choose the most relevant features and fed the results to SVM classifier
AlJarrah et al. [209]	Information Gain				
Gharib et al. [210]					Fed the features to Deep Auto Encoder (DAE) neural network classifier
Sun et al. [211]			Criteria based on the ratio between the number of apps in both datasets, PCA		Fed the extracted features to a Positive and Unlabelled (PU) learning classifier

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_NUMBERS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
```

Fig. 2: Snapshot of permissions requested by WhatsApp Messenger app

4 Features used

In response to RQ3, this section aims to present the results obtained after analyzing the types and number of features used for Android malware detection in combination with permissions. We divide the features used into two categories: Static and Dynamic.

4.1 Static features

The static analysis investigates malware without the real code or instructions being executed. It provides basic information about app functionality and collect technical indicators from the AndroidManifest.xml and other resource files. In other words, it can be defined as a source code review of an Android application file. Several reverse engineering tools like Apktool[3] or AAPT2 [4] can be used to decompile an apk and extract the features required. The features that can perform static analysis of applications are called static features. Some of the commonly used examples of static features are explained in brief below with Android permission being the most popular one -

(a) Permissions - The presence of App permissions helps the user in two ways mainly-
  - By protecting access to the **Restricted data**, such as user's personal information.
  - By protecting access to the **Restricted actions**, actions such as recording audio.

Because of the above two reasons, it has been made mandatory by the Android OS and the Android permission check system for all application developers to declare the list of permissions their application needs for its functionality or invoke the Android API successfully. Hence, the manifest file contains the list of all Android permissions required to run the application efficiently. Permission is declared using the <uses-permission>tag within the manifest file. For example, as shown in Figure 2, which is the snapshot of the AndroidManifest.xml file of "The WhatsApp Messenger" app, requires permissions such as "READ_PHONE_STATE", "READ_PHONE_NUMBERS", "RECEIVE_SMS", "VIBRATE" and "AUTHENTICATE_ACCOUNTS" to execute on Android smartphones. Some permissions fall under the category of install-time permissions, i.e., they are automatically granted upon the installation of the app, whereas some permissions are known as runtime permissions, which are further requested at runtime. Install-time permissions permit the app limited access to restricted data or actions that can affect the user to a minimal amount. Install-time permissions can be further divided into the following types -

- Normal permissions - These permissions present minimal risk to the user's privacy and the functionality of other apps.
- Signature permissions - These permissions are granted by the permission check system only when the requesting app is signed by the same certificate as the one that declared the permission.

Runtime permissions, often addressed as dangerous permissions, are requested at runtime by the application to ask for access to view restricted data or perform any prohibited action by presenting a runtime permission request prompt.

4. API calls- Application Programming Interfaces (APIs) act as the medium for one program to interact with another, and an API call or request can be defined as a message sent to a server asking an API to provide a service or information. After traveling from a client to an API endpoint and being received by the server, it is processed, and the request is executed in return for a response.
5. Intent: An Intent is a messaging object a developer can use to request an action from another app component. Three main fundamental use cases of intents are starting an activity, starting a service, and delivering a broadcast.
6. Opcode sequence: Opcodes can reflect the behavior pattern of an application to a certain extent using the underlying machine code, so they are often used as static features. They are extracted by decompiling the APK file and are generally obtained to facilitate the input detection model.

4.2 Dynamic features

Dynamic analysis opts for a different approach than static analysis. Instead of examining the code, it relies upon monitoring an application's behavior while it is running over any virtual or real CPU. As the name suggests, dynamic analysis is performed by analyzing the runtime behavior of applications, and the features analyzed during this process are called dynamic features, such as -

[3] https://apktool.en.lo4d.com/windows

[4] https://developer.Android.com/studio/command-line/aapt2

1. Network traffic - In simple words, network traffic is the amount of data that moves across a network during any time interval.
2. System calls - A system call provides an interface between a process and the OS. They enable the operating system's services via API for the user programs to use. Some important system calls commonly used in the OS are wait(), fork(), exec(), kill():, exit(): etc.

   Other commonly used features include hardware components, description of applications, network addresses, code-related patterns, etc.

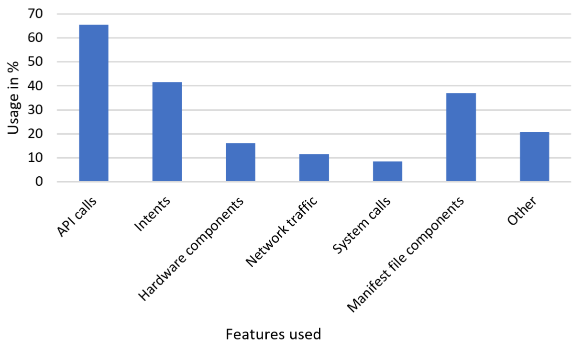

Fig. 3: Statistics depicting the usage of features in combination with permissions

Table 5 summarizes the information regarding the usage of permissions as a feature in combination with other features. Based on the information depicted by Table 5, we can conclude that in combination with permissions, API calls and intents are the most used features by researchers for Android malware analysis/detection. Figure 3 presents a statistical summary of Table 5. As already discussed above and highlighted by the figure too, API calls [125]-[136] and intents [96]-[100] account for 65.38% and 41.53% usage respectively, i.e., out of the total research papers used in this review, majority of them utilized API calls or intents in combination with permissions. The third most commonly used feature is another static feature called hardware components [194]-[195], and it accounts for 16.15% of the total. Some dynamic features such as network traffic [76]-[79] and systems calls [47]-[49] are also seldom used by researchers and practitioners in combination with permissions and their shares are 11.53% and 8.46% respectively. Apart from the static features mentioned above, many other components present in the manifest file of an apk prove to be a popular partner of permissions while building an analysis/detection model, as they account for a surprising 36.92% of the total. We note that intent and hardware components are also a part of the manifest file only, but since the researchers have extensively used them in combination with permissions, we display their usage separately instead of combining them with other rarely used Manifest file components. These features include App descriptions [109], version numbers [112], app components [116], meta information [126] and many more. Other features commonly used in combination with permissions are small file size [65], dex file [96], URLs [90], code-related information [130] etc., and they form 20.76% of the share. **Hence, based on the results presented above, we answer the third Research Question that the researchers have preferred using API calls and intents the most in combination with permissions for Android malware analysis and detection.**

5 Model used

In response to RQ4, this section describes the various ML and DL models [214] used by researchers and practitioners to conduct their research over the chosen range of reviews.

5.1 Machine Learning

Machine learning is a subset of artificial intelligence and a popular technology enabling machines to learn from past data and perform a given task automatically. Machine learning can be broadly divided into two types -

1. Supervised Machine learning
2. Unsupervised Machine learning

5.1.1 Supervised Machine learning

Just like a student who learns a concept under the supervision of a teacher, the machines are used to predict an output correctly with the help of the training data working as a supervisor that teaches the machines. The type of machine learning in which the devices are trained using well

Table 5: Features Used

Related works	Permissions	API calls	Hardware Components	Intents	Others
Zhang et al. [19]	✓	✓			
Enck et al. [14]	✓			✓	
Yerima et al. [47]	✓	✓			System calls
Yerima et al. [49]	✓	✓			System calls
Wu et al. [53]	✓	✓		✓	
Shrivastava et al. [59]	✓			✓	
Upadhayay et al. [60]	✓				Network traffic
Surendran et al. [62]	✓	✓			System calls
A. T. Kabakus [63]	✓	✓		✓	
Wang et al. [64]	✓	✓			Opcode sequences
Akbar et al. [65]	✓				Permission rate, smali file size
Zhu et al. [66]	✓	✓			Permission rate, system events.
Wang et al. [67]	✓	✓			
N. McLauglin [68]	✓				Opcode sequences
Grace et al. [70]	✓				Log file for app activities.
Liu et al. [71]	✓	✓			
Bayazit et al. [72]	✓			✓	
Lee et al. [73]	✓				
Zhu et al. [74]	✓	✓			
Almahmoud et al. [75]	✓	✓			Permission rates, system calls
Feng et al. [76]	✓			✓	Network traffic
Kandukuru et al. [77]	✓				Network traffic
Arora et al. [78]	✓				Network traffic
Ding et al. [79]	✓			✓	Network traffic
Idrees et al. [81]	✓			✓	
Khariwal et al. [82]	✓			✓	
Idrees et al. [83]	✓			✓	
Zhu et al. [15]	✓	✓	✓		
Bai et al. [84]	✓				Opcode sequences
Taheri et al. [85]	✓	✓		✓	
Alazab et al. [86]	✓	✓			
Imtiaz et al. [88]	✓	✓		✓	Network traffic
Liu et al. [89]	✓	✓		✓	System commands, opcodes, Package and FlowDroid's features [175], Network traffic
Chen et al. [90]	✓			✓	URL's, data flow features
Guan et al. [91]	✓	✓			
Mohamed et al. [92]	✓	✓			
Peng et al. [96]	✓			✓	Dex file headers, power spectrum density information of dex file structure entropy
Ashwini et al. [97]	✓	✓	✓	✓	
Jiang et al. [98]	✓			✓	
Wang et al. [99]	✓	✓		✓	
Rana et al. [100]	✓		✓	✓	
Lu et al. [101]	✓	✓			Resource, semantic and dynamic features
Millar et al. [102]	✓	✓			Opcode sequences
Shabtai et al. [104]	✓				Dex features, Xml features and apk features
Felt et al. [105]	✓	✓			
Erickson et al. [106]	✓	✓			
Jhu et al. [109]	✓				App Descriptions
Peiravian et al. [110]	✓	✓			
Sanz et al. [111]	✓		✓		
Feldman et al. [112]	✓			✓	Low version numbers, network traffic
Arp et al. [116]	✓	✓	✓	✓	App components, Network addresses
Yerima et al. [117]	✓	✓			Linux/Android commands
Kang et al. [118]	✓	✓		✓	File hash, serial number, System commands
Zhao et al. [119]	✓	✓			Action features and IP, URL features
Qiao et al. [120]	✓	✓			
Chen et al. [121]	✓	✓			Network traffic, newly defined feature *Sequence* based upon API calls
Demertzis et al. [122]	✓				Battery, Memory, CPU, Network

Verma et al. [123]	✓	✓	✓	✓	App components, network addresses
Wang et al. [124]	✓		✓		Receiver actions
Tangil et al. [125]	✓	✓		✓	Meta information, new features from app's resource files
Wang et al. [126]	✓	✓	✓	✓	Components, protected strings, IP addresses, and URL, commands
Li et al. [127]	✓	✓		✓	App components
Xie et al. [129]	✓	✓			
Xie et al. [130]	✓	✓	✓	✓	Code related patterns
Ren et al. [131]	✓	✓			Opcode sequence, hardware components
Tao et al. [132]	✓	✓			
Qiu et al. [135]	✓	✓	✓	✓	Network addresses
Zhu et al. [136]	✓	✓			Permission rate, monitoring system events, data flows
Feng et al. [137]	✓				App description
Aonzo et al. [138]	✓	✓			
Urooj et al. [139]	✓	✓		✓	App components, Packages, Receivers, services
Wang et al. [140]	✓				API docs, user reviews, app descriptions
Wang et al. [141]	✓	✓	✓	✓	Code patterns, functional call graphs
Ibrahim et al. [144]	✓	✓			Opcode sequences, application size, services, receivers, fuzzy hash
Arshad et al. [145]	✓	✓	✓	✓	Application components, system call logs, network addresses
Yuan et al. [146]	✓	✓			Dynamic App actions
Zhou et al. [147]	✓	✓			Network addresses
Cilleruelo et al. [148]	✓		✓		Information published on the Google Play Store
Firdaus et al. [149]	✓			✓	Code based features, system commands, directory path features
Wang et al. [150]	✓				
Singh et al. [151]	✓	✓			
Rafiq et al. [152]	✓	✓		✓	
Mahdavifar et al. [153]	✓	✓		✓	Packages, receivers, system calls, basic binders, composite behavior
Mahindru et al. [155]	✓	✓			User rating, number of user download apps
Anupama et al. [156]	✓				System calls
Chen et al. [157]	✓	✓			
Mahindru et al. [158]	✓	✓			
Tchakounté et al. [159]	✓				Resource names, .dex codes, source codes, package name certificate
Nissim et al. [160]	✓				Metadataa, number of activities, services, receivers, permissions, providers
Peynirci et al. [161]	✓	✓			Strings
Bao et al.[164]	✓	✓			App description
Shatnawi et al. [166]	✓	✓			
Smmarwar et al. [167]	✓	✓		✓	
Manzanares et al. [169]	✓			✓	System calls, hardware-software features etc.
Bhat et al. [170]	✓	✓	✓	✓	Network addresses
Elayan et al.[171]	✓	✓			
Syrris et al. [172]	✓	✓		✓	Actions, services, Broadcast, categories
Idrees et al. [173]	✓			✓	
Rehman et al. [174]	✓			✓	Process name
Martin et al. [175]	✓	✓		✓	Opcode sequences, system commands, Network data etc.
Navarro et al. [176]	✓				App components
Milosevic et al. [177]	✓				Source code
Alzaylaee et al. [178]	✓	✓		✓	
Cai et al. [179]	✓	✓		✓	App components, shell commands
Badhani et al. [180]	✓	✓			
Hijawi et al. [181]	✓		✓		Software components, URL, broadcast receivers features

Sheen et al. [182]	✓	✓			
Nisha et al. [183]	✓				Components and interfaces
Song et al. [184]	✓				MD5 blacklist database, permission intention
Zhang et al. [185]	✓	✓		✓	
Yang et al. [186]	✓	✓			
Qaisar et al. [188]	✓			✓	Services provider, new APK attributes
Appice et al. [189]	✓	✓		✓	Network addresses
Zhu et al. [190]	✓	✓			Permission rate, system events
Su et al. [192]	✓	✓	✓	✓	Strings, certificate-payload info, code pattern
Mahindru et al. [193]	✓	✓			Number of user download an app, rating of an app
Dehkordy et al. [194]	✓	✓	✓	✓	URLs, activity, service receiver, provider
Nguyen et al. [195]	✓	✓	✓	✓	Provider, activity, service, URLs
Taheri et al. [196]	✓	✓			
Mahesh et al. [197]	✓			✓	
Firdaus et al. [198]	✓				Directory path, telephony
Shrivastava et al. [199]	✓			✓	
Varsha et al. [200]	✓		✓	✓	Opcodes, strings, app components
Mahindru et al. [202]	✓	✓			Number of user download an app, rating of an app
Keyvanpour et al. [203]	✓	✓	✓	✓	
Razak et al. [204]	✓				
Xie et al. [205]	✓	✓	✓		
Mahindru et al. [206]	✓	✓			Number of user download an app, rating of an app
Alecakir et al. [18]	✓				App description
Ali et al. [207]	✓				Rating, comments and number of downloads related to an app
Sun et al. [208]	✓	✓		✓	Package Name
AlJarrah et al. [209]	✓	✓			Contextual information
Gharib et al. [210]	✓	✓			Logos, strings of notification messages, system call sequences
Sun et al. [211]	✓	✓			IP address, requested URLs

"labeled" training data, i.e., some input data is already tagged with the correct output, to find a mapping function between the input variable with the output variable is called supervised machine learning. Practical usage of supervised machine learning includes Risk Assessment, Image classification, Fraud Detection, and spam filtering. Supervised learning can be divided into two types -

1. Classification
2. Regression

Classification - This type of learning is used when the output variable is categorical, i.e., Yes-No, Male-Female, etc. type. Some commonly used classification algorithms are discussed in detail below -

1. Decision Tree (DT) Algorithm - Generally considered a classification problem; however, a Decision tree (DT) is a supervised learning technique that can be used both for classification and regression problems. As the name suggests, it is a tree-structured classifier utilizing the Classification and Regression Tree (CART) algorithm, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. The main step while implementing a decision tree is to select the best attribute for the root and sub-nodes, and for that, the DT algorithm tries to maximize the information gain value, and iteratively the node having the highest information gain is chosen first. DT algorithm is further divided into three types -
   - ID3 DT (Iterative Dichotomiser 3)
   - C4.5 DT
   - J48 DT
2. Random Forest (RF) Algorithm - This type of supervised learning algorithm works on the ensemble learning concept, which includes combining multiple classifiers or *decision trees* to solve a complex problem and avoid the problem of overfitting. RF works in two parts; the first part revolves around creating the random forest by combining n number of DTs, whereas, in the second, the prediction for each tree is made.

RF and Bagging classifier (BC) come under the category of Bagging algorithms that work on the criteria of combining the results of multiple models.

3. Logistic Regression (LR) algorithm - LR predicts the output of a categorical dependent variable, but instead of giving the exact value as 0 and 1, it produces the probabilistic values between 0 and 1. In the case of LR, instead of fitting a regression line, a logistic function, also known as the sigmoid function, maps the predicted values to probabilities.
4. K-Nearest Neighbor(KNN) Algorithm - KNN works on the hypothesis that a similarity exists between the new and available cases. After analyzing the data, it puts the new case into the category that is most similar to the available ones, and this whole process takes place by calculating the Euclidean distance of K number of neighbors.
5. Support Vector Machine (SVM) - The major aim of the SVM algorithm is to build the best line or decision boundary or hyperplane capable of segregating the n-dimensional space into classes so that the new data point can be easily fitted in the correct category for future cases. SVM uses extreme points, also known as support vectors, to help create a hyperplane; hence the algorithm is called Support Vector Machine. Sequential Minimal Optimization (SMO) is an algorithm for solving the quadratic programming (QP) problem that arises during the training of support vector machines (SVM). SVMs use the Kernel Trick and various kernel methods to transform linearly inseparable data into linearly separable data, thus finding an optimal boundary for possible outputs. Some of them are given below -
   - Linear kernel
   - Polynomial kernel (Poly)
   - Radial Basis Function (RBF)
   - Kernel change detection algorithm (KCD)
6. Naïve Bayes (NB) Algorithm - It is one of the fastest machine learning models based upon the Bayes theorem or Bayes' Rule, often used to determine the probability of a hypothesis with prior knowledge depending upon the conditional probability. The NB algorithm is mainly used in text classification problems with a high-dimensional training dataset. Three types of Naive Bayes model are commonly used in the literature -
   - Gaussian
   - Multinomial
   - Bernoulli
7. Bayesian networks (Bayesnet) - These are probabilistic graphical models that utilize Bayesian inference for probability computations. It is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable.

Regression This type of learning is used when a relationship exists between the input and the output variable and the prediction is of continuous variables. Some commonly used regression algorithms are linear regression, regression trees, nonlinear regression, Bayesian linear regression, polynomial regression [214] etc.

1. Linear Regression - With the concept of performing predictive analysis and the assumption of variables having a linear relationship, linear regression finds how the value of the dependent variable changes according to the value of the independent variable. Mathematically, linear regression finds the best-fit line with the least error, which means that the Mean Squared Error (MSE) between the predicted and actual values should be minimized.

5.1.2 Unsupervised Machine Learning

In the absence of labeled data, the models are trained using the unlabelled data set and are allowed to act without supervision, and the correct corresponding technique is called unsupervised machine learning. Unlike supervised learning, unsupervised learning cannot be used directly for a regression or classification problem because the related output data doesn't exist. The main aim of unsupervised learning is to analyze and locate the underlying structure of data, categorize that data according to similarities, and present that data in a compact format. Unsupervised learning is more suited for real-world problems as we often do not have input data with the corresponding output data. Unsupervised learning can be further divided into two types -

1. Clustering
2. Association

Clustering In this type of unsupervised learning, the data points are grouped into clusters of similar data points, and this is done by finding the presence or absence of similar patterns in the unlabelled data set. Commonly used for statistical data analysis, this technique provides each cluster with a cluster ID which is further utilized by machine learning algorithms to simplify the processing of huge and complex problems. One of the most commonly used clustering algorithms is the K- Means algorithm, capable of efficiently dividing the samples into different clusters of equal variances with the linear complexity of $\mathbf{O(n)}$. Other clustering methods used by researchers over the years include Agglomerative Hierarchical Clustering (AHC) [51] Farthest First (FF) clustering, Filtered clustering (FC), Density-Based clustering (DB) [202], Hidden Markov Model (HMM), Google Distance (GD) Clustering [208] etc.

Association Rule Learning Association rule learning aims to find relations or associations among data variables using the concept of *If and Else* statements. Association learning relies upon support, confidence, and lift to compute the associations between thousands of items. Some of the commonly used association rule learning algorithms are discussed in brief below -

1. Apriori algorithm - The algorithm relies upon breadth-first search and Hash Tree to efficiently compute and find the itemset associations after performing iterations on the large dataset.
2. F-P Growth algorithm - Frequent Pattern Growth (F-P) Algorithm revolves around finding frequent patterns without candidate generation. Unlike Apriori's functioning, which includes the generate and test strategy, it constructs an FP Tree to fragment the items' paths and generate frequent patterns.

5.1.3 Boosting in Machine learning

Combining various weak classifiers to build strong classifiers is known as the boosting technique in machine learning, and it is one of the most popular learning ensemble modeling techniques. Iterations to combine old ones and build new models are performed until training data capable of producing optimum predictions is prepared. Adaptive boosting (AdaBoost) was the first algorithm to combine various weak classifiers into a single strong classifier in the history of machine learning. Some of the other major boosting algorithms used in the literature are as follows -

- Gradient Boosting Machine (GBM)
- Extreme Gradient Boosting Machine (XGBM)
- Light GBM
- CatBoost

5.2 Deep Learning

Deep learning can be defined as a subset of machine learning as it has similar working but different capabilities and approaches. Deep learning, also known as Deep Neural Networks (DNN), is inspired by the functioning of the human brain cells/ neurons and leads to the concept of artificial neural networks capable of learning and discovering insights from data. Some popular deep learning models used by researchers over the years are discussed in brief below -

1. Convolutional Neural Network (CNN) - These networks take images as input, then assign importance, i.e., learnable weights and biases, to parts of the input to differentiate from one another. Through the help of relevant filters, a CNN successfully captures the Spatial and Temporal dependencies in an image.
2. Recurrent Neural Network (RNN) - Often used in NLP or language translation, these networks use sequential or time series data as inputs. Unlike other neural networks, the output of RNN depends on the prior elements within the sequence as it takes information from prior inputs too. Some popular variants of RNN often used in the literature are given below -
   - Long short-term memory (LSTM) - LSTM was proposed to solve the vanishing gradient and short-term memory problems. LSTM has "cells" type structures in the hidden layers of the neural network, which have three gates–an input gate, an output gate, and a forget gate capable of controlling the flow of information.
   - Gated recurrent units (GRUs) - Similar to LSTM, even GRUs are capable of dealing with the short-term memory problem of RNN models, but unlike LSTM, GRUs use hidden states and have two gates: a reset and an update gate.

3. Artificial Neural Network (ANN) - The working of ANN is similar to the working of nerve cells in the human brain. An artificial neural network has three or more layers that are interconnected. The first layer consists of input neurons. Those neurons send data to the deeper layers and then transfer the final output data to the last output layer. A fully connected multi-layer neural network is called a Multi-layer Perceptron (MLP).
4. Deep Belief Neural (DBN) Networks - These are the classifiers that use layers of stochastic latent variables, which make up the network. The top couple of layers in the DBN have no direction, but the layers above them have direct links to lower layers. DBNs have an edge over traditional neural networks as they can be generative and discriminative models.
5. Deep Autoencoders (DAE) - The working of autoencoders is a bit different as they function as a special type of feedforward neural network in which the input is the same as the output. They tend to convert the information into a lower dimensional code, also known as latent-space representation, which is further used to reconstruct the result. The structure of an autoencoder comprises three parts: an encoding method, a decoding method, and a loss function.
6. Self-Organising Map (SOM) - As depicted by the name, the map organizes itself without any supervision from anyone or anything. The training of SOM is done through a fierce neural network, a single-layer feedforward network that resembles the brain's functioning. Apart from being used as a classifier, it can also be used as a non-linear dimensionality reduction technique.

Table 6 summarizes researchers' choice between ML, DL, or any other algorithm. In response to RQ4, we conclude by the information in Table 6 that most researchers and practitioners opted for traditional supervised machine learning algorithms like DT, SVM, RF, and NB due to their convenient and simple working. But nowadays, researchers have realized that machine learning has some limitations too. It relies too much on human intervention, so it might not work well with complex feature engineering problems like Natural Language Processing (NLP),

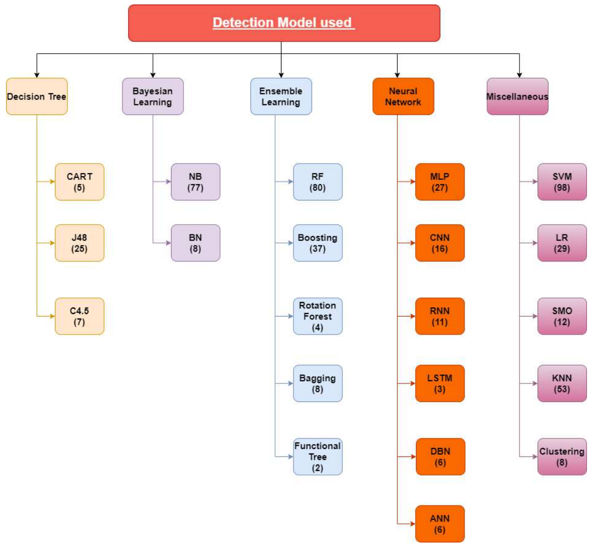

Fig. 4: Statistics depicting the usage of various detection models

Image recognition, etc. Hence, as seen in Table 6, researchers have explored a new area of Deep Neural networks and models like CNN, ANN, DAE, etc., for the past few years.

Figure 4 gives us a closer understanding of Table 6. We classify the majorly used detection models from Table 6 into five categories: Decision Trees, Bayesian Learning, Ensemble Learning, Neural Networks, and Miscellaneous to analyze the usage of different classifiers. For instance, Figure 4 highlights that five, 25, and seven papers out of the 200 related work reviewed in our work involved the usage of CART, J48, and C4.5 Decision trees, respectively. However, note that 46.5% of the total work utilized the DT classifier without mentioning the particular type. With the help of the information highlighted by the figure, we can conclude that the five most used detection models include SVM [64]-[66], DT [110]-[115], RF [124]-[131] from Ensemble learning, NB [165]-[167] from Bayesian Learning and KNN [73]-[75] with 98 (49%), 93 (46.5%), 80 (40%), 77 (38.5%) and 53 (26.5%) corresponding number (percentage) of studies that involved the classifiers mentioned respectively. It is worth noting that due to the growing popularity of deep learning, researchers have extensively started using neural network classifiers in their work. Neural networks [72]-[76] have been involved in more than 35% of the studies mentioned in our review. Apart from the ML/DL

Table 6: Model used

Related works	Machine Learning (ML)	Deep Learning (DL)	Others
Zhang et al. [19]			Permission behavioural pattern monitoring model
Li et al. [20]	SVM, Rotation Forest, PART, FT, Random Committee, RF.		
Sahin et al. [21]	NB, SMO, RF, C4.5 DT, LR, KNN	MLP	
Talha et al. [22]			Detection algorithm based upon permission score.
Varma et al. [23]	NB, J48 DT, RF	MLP, Multi-class classifier	
Mahindru et al. [24]	NB, J48 DT, RF, Simple Logistic (SL), kstar		
Dogru et al. [25]			Detection algorithm based upon permission score
Rathore et al. [26]			
Shang et al. [27]	Improved NB classification algorithm		
Tchakounte et al. [28]			Algorithm based upon permission patterns
Ju et al. [16]			Manual pattern recognition to existing malware permissions patterns
Ilham et al. [29]	RF, SVM, J48 DT		
Sahin et al. [30]	NB, KNN		
Angelo et al. [31]	J48 DT, NB	MLP	
Xiong et al. [32]			Developed a classifier name *ENCLAMALD* based upon permission patterns
Lu at el. [33]	RF, DT, Apriori algorithm		
Kavitha et al. [34]			
E. Amer [35]	RF, AdaBoost, SVM, DT	MLP	
Chakravarty et al. [36]	J48 DT, Random Committee, SMO, Randomizable filtered classifiers	MLP	
Pondugula et al. [37]		DNN	
Sahal et al. [38]	J48 DT, NB, SVM, KNN		
Tuan Mat et al. [39]	NB		
Wang et al. [40]	SVM, DT, RF	DCN	
Park et al. [41]	RF		
Liang et al. [42]			Developed their own rule-based algorithm
Enck et al. [14]			Analysis of the newly produced Android OS in 2009
Enck et al. [17]			Defined a new Application working upon its own Syntax Language
Wang et al. [43]			Algorithm based upon calculating the risk score of each app
Peng et al. [44]	NB and its modifications		
Pandita et al. [45]			NLP techniques to read application descriptions
Samra et al. [46]	k-means		
Yerima et al. [47]	Bayesian Classifier		
Aung et al. [48]	K-means clustering, J48 DT, RF, CART		
Yerima et al. [49]	Bayesian Classifier		
Sanz et al. [50]	SL, NB, Bayesnet, SMO, IBK, J48 DT, RandomTree		
Moonsamy et al. [51]	AHC		

Backes et al. [52]			Algorithm to assess and rewrite application policies
Wu et al. [53]	NB, KNN		
Kato et al. [54]	Stacking Ensemble Learning (SEL), RF		
Arora et al. [55]			Algorithm to form graphs based upon permission patterns and frequency
Alsoghyer et al. [56]	RF, J48 DT, SMO, NB		
Saleem et al. [57]			
Ghasempour et al. [58]	DT, SVM		
Shrivastava et al. [59]			Algorithm to calculate the risk score of an application using the feature's frequency
Upadhayay et al. [60]	RF, SVM , NB		
Lee et al. [61]	KNN, SVM, AdaBoost, Extra Tree, RF		
Surendran et al. [62]	Tree augmented NB		
A. T. Kabakus [63]		CNN	
Wang et al. [64]	SVM, GBDT, XGBM, LightGBM, CatBoost, LR		
Akbar et al. [65]	SVM, Rotation Forest, RF, NB		
Zhu et al. [66]	SVM, An ensemble taking DT as a base and involving PCA to form Rotation Forest		
Wang et al. [67]		Graph convolutional network	
N. McLauglin [68]		CNN	
Wang et al. [69]	SVM, RF, DT		
Grace et al. [70]			Risk level based detection of own algorithm
Liu et al. [71]	Multi-layered gradient boosting DT algorithm, SVM, DT, RF, XGBM		
Bayazit et al. [72]		RNN-based LSTM, BiLSTM, GRU	
Lee et al. [73]	J48 DT, RF, Decision Table, NB, SMO, SVM, LR, IBK, KNN, AdaBoost	MLP	
Zhu et al. [74]	SVM, KNN	Merged Sparse Auto-Encoder (MSAE), Stacked Denoising Auto-encoders (SDAE).	
Almahmoud et al. [75]	SVM, KNN, NB, RF, DT	RNN	
Feng et al. [76]		CNN	
Kandukuru et al. [77]	DT		
Arora et al. [78]	Apriori and FP-Growth algorithms		
Ding et al. [79]	KNN, multinomial NB, DT, RF, SVC, NuSVC, LinearSVC, LR, GBDT, XGBM	Combination of a residual network (ResNet) and LSTM	
Sahin et al. [80]	KNN, NB, SVM, DT, Bagging-DT algorithms		Linear regression-based and bagging method ensemble algorithms
Idrees et al. [81]			Algorithm based on the identification of distinct usage patterns of permissions and intents
Khariwal et al. [82]	RF, SVM, NB		
Idrees et al. [83]	NB, Kstar, Prism		
Zhu et al. [15]		CNN based Multi-Head Squeeze-and-Excitation Residual block	
Bai et al. [84]	CatBoost		
Taheri et al. [85]	Modified versions of KNN		
Alazab et al. [86]	RF, J48, RT, KNN and NB		

Mathur et al. [87]	KNN, SVM, LR, RF, ET, XGBM, AdaBoost, BG		
Imtiaz et al. [88]	DT, SMO, NB	MLP, ANN	
Liu et al. [89]	AdaBoost, Bagging, ExtraTrees, GBM, RF, Voting		Subspace based Restricted Boltzmann Machines (SBRBM)
Chen et al. [90]	LightGBM SVM, GBM, AdaBoost, Extra-TreesClassifier, RF, KNN, LR	Neural Networks	
Guan et al. [91]	RF, KNN, NB, SVM		Class Imbalanced Learning (CIL), Synthetic Minority Over-Sampling Technique (SMOTE), Random under-sampling (RUS), Balance Cascade
Mohamed et al. [92]	KNN, NB, SVM, DT		
Varma et al. [93]	RF, SVM, KNN, DT, Nearest Centroid (NC)		Bat Optimization Algorithm for Wrapper-based Feature Selection (BOAWFS)
Gyunka et al. [94]	NB, SL, RF, Partial DT, KNN, SVM		
Taha et al. [95]	LightGBM, SVM, KNN, LR, NB, DT.		
Peng et al. [96]		MLP, CNN	
Ashwini et al. [97]	Ridge Classifier, XGBM, RF, SVC		
Jiang et al. [98]	J48, KNN, NB, SVM		
Wang et al. [99]	LR, DT, SVM	DBN	
Rana et al. [100]	DT, RF, Extremely randomized trees (ERT), GB Tree, SVM, LR		
Lu et al. [101]		DBN, GRU	
Millar et al. [102]		CNN	
Barrera et al. [103]		SOM	
Shabtai et al. [104]	DT, NB, BN, BBN, BDT, PART, RF		
Felt et al. [105]			Own approach Stowaway to compare permissions required to invoke API methods and actually requested permissions
Erickson et al. [106]			
Sarma et al. [107]	SVM		
Frank et al. [108]			
Jhu et al. [109]	Naive Bayes with Multinomial Event Model		
Peiravian et al. [110]	SVM, DT, Bagging predictor		
Sanz et al. [111]	KNN, DT, Bayesnet, SVM		
Feldman et al. [112]	NB, SVM, KNN, C4.5 DT		
Pehlivan et al. [113]	Bayesian Classification, CART, DT, RF, SMO		
Rahman et al. [114]	NB, KNN, DT, RF, Decision Forest		
Rovelli et al. [115]	C4.5 DT, kstar Lazy Learning, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), NB, AdaBoost		
Arp et al. [116]	SVM		
Yerima et al. [117]	RF, NB, DT, random trees, SL		
Kang et al. [118]	NB		
Zhao et al. [119]	SVM, KNN, J48, NB		
Qiao et al. [120]	SVM, RF	ANN	
Chen et al. [121]	SVM, C4.5 DT, NB, KNN, Bagging predictor	MLP	
Demertzis et al. [122]		Evolving Spiking Neural Network (eSNN), MLP, Radial Basis functions, GMDH PNN, eSNN	Extreme learning machine (ELM)

Verma et al. [123]	SVM, Minimal complexity machine		
Wang et al. [124]	NB, LR, DT, Bayesnet, RF, SVM		
Tangil et al. [125]	Extra Trees, SVM, RF, XGBoost		
Wang et al. [126]	KNN, J48 DT, RF		
Li et al. [127]	NB, DT, LR, SVM, RF		
Bhattacharya et al. [128]	Bayesnet, DT, IBK, J48 DT, JRip, Kstar, LR, RF, SMO, NB, etc.	MLP	
Xie et al. [129]	DT, RF, SVM, KNN, NB		
Xie et al. [130]	SVM, LR, RF	CNN	
Ren et al. [131]	SVM, KNN, AdaBoost , RF, GBM		
Tao et al. [132]	RF, SVM, DT		
Namrud et al. [133]	SVM, K means, SOM		
Alswaina et al. [134]	Extremely Randomized Trees (ET), SVM, ID3 DT, RF, BC, KNN	NN	
Qiu et al. [135]	SVM, DT	DNN	
Zhu et al. [136]	SVM, KNN, DT, LR, RF, ET, Adaboost, GBDT	MLP	
Feng et al. [137]	LR, NB-SVM	GRU , CNN	
Aonzo et al. [138]			Proposed their own linear and non linear classifiers
Urooj et al. [139]	AdaBoost, SVM, RF, KNN, NB, RBF, DT, SVM		
Wang et al. [140]			Evaluated their own text semantic-based approach
Wang et al. [141]	SVM, KNN, RF		
Zhang et al. [142]			Own algorithm to detect bugs and Implicitly Malicious Behavior
Kesswani et al. [143]	NB		
Ibrahim et al. [144]	Stochastic Gradient Descent (SGD), DT, RF, SVC, KNN, XGBoost, GaussianNB	GRU	
Arshad et al. [145]	SVM, RF, NB, DT		
Yuan et al. [146]	NB, C4.5 DT, LR, SVM	DBN, CNN, MLP	
Zhou et al. [147]	RF, KNN, SVM	GRU	
Cilleruelo et al. [148]	SVM, Stochastic gradient descent(SGD), RF, XGBM		
Firdaus et al. [149]	NB, FT, J48 DT, RF	MLP	
Wang et al. [150]	KNN, SVM, NB, J48 DT		
Singh et al. [151]	SVM(Linear)		
Rafiq et al. [152]	SVM, LR, DT, XGBM, RF, KNN		
Mahdavifar et al. [153]	RF, SVM, KNN	Pseudo-Label Stacked Auto-Encoder (PLSAE), Pseudo-Label Deep Neural Network (PLDNN), Label Programming (LP)	
Seraj et al. [154]	RF, Gaussian NB, SVC, KNN	MLP	
Mahindru et al. [155]	DT, LR, NB, SVM with linear kernel, polynomial kernel, and RBF kernel	SOM, NN	
Sahin et al. [21]	KNN, NB, RF, C4.5 DT, LR	SMO, MLP	
Anupama et al. [156]	LR, CART, RF, SVM	DNN, CNN	
Chen et al. [157]	RF	CNN	
Mahindru et al. [158]	LSSVM with distinct kernel functions (polynomial, RBF and linear), LR, NB, DT	NN	
Tchakounté et al. [159]			Algorithm based on fuzzy hash processed static features and computed similarity score

Nissim et al. [160]	SVM		Active Learning methods
Peynirci et al. [161]	ID3, CART, J48 DT, RF, NB, RBF, SLogReg, Prism, MODLEM, OneR, NNGE, SVM	MLP	
Nauman et al. [162]	Bayesian Statistical Inference	Neural networks, DAE , DBN, CNN , LSTM	
Bhattacharya et al. [163]	Bayesnet, NB, Decision Table, Random Tree, RF, J48 DT	SMO, MLP	Quick Reduct algorithm
Bao et al.[164]	NB, Multinomial classification		Algorithms based upon predicting permissions from API and text classification
Medrano et al. [165]	-	-	
Mat et al. [165]	NB		
Shatnawi et al. [166]	LR, SVM, KNN, NB, GaussianNB		
Smmarwar et al. [167]	OEL, fine tree, NB, cubic SVM, Weighted KNN, Bagging, Adaboost, GentleBoost		
Arif et al. [168]			Information Gain
Manzanares et al. [169]			Extracted applications to form a comprehensive dataset
Bhat et al. [170]	DT, LR, SVM, NB, RF		
Elayan et al.[171]	SVM, KNN, NB, DT, RF	GRU	
Syrris et al. [172]	NB, RF, SVM	ANN	
Idrees et al. [173]	NB, DT, Decision Table, RF	SMO, MLP	
Rehman et al. [174]	SVM, KNN, DT, Linear Discriminant Analysis		
Martin et al. [175]	AdaBoost, Bagging (with RF estimators), ExtraTrees, GBM, RF, Voting classifier combining an RF, KNN, DT		
Navarro et al. [176]	RF, CART, Bagging		
Milosevic et al. [177]	C4.5 DT, NB, SVM with SMO, RF, JRIP, LR, AdaBoost, Farthest First, Simple K-means and Expectation maximization (EM) algorithms		
Alzaylaee et al. [178]	SVM Linear and RBF, NB, SL, DT, PART, J48 DT	MLP	
Cai et al. [179]	SVM, LR, RF, KNN, Gaussian NB, multinomial NB, LDA, QDA, RF	MLP	
Badhani et al. [180]	DT, ELM, LR, RIPPER, SVM-linear		
Hijawi et al. [181]	RF, J48 DT, NB, SVM, KNN	MLP	
Sheen et al. [182]	Adaboost, IBK and J48 DT, Decision Stump, Random tree, NB, SVM		
Nisha et al. [183]	KNN, SVM, RF, GBT	CNN	
Song et al. [184]			Dangerous permissions matching model to compute a threat degree threshold
Zhang et al. [185]			Algorithm to identify explicit and implicit permission use points to further create permission use graphs to analyze the permissions behavioral pattern
Yang et al. [186]			Algorithm to construct State Transition Graphs (STG) from permissions to analyze permission-related behavior only under the combinations of the relevant permissions
Thiyagarajan et al. [187]	DT		

Qaisar et al. [188]	SVM, DT		Case-based learning approach to store and learn permissions like features
Appice et al. [189]	RF, SVM, stacking, ensemble	Deep-Net	Algorithm involving clustering-based sampling with the specific multi-view learning approach
Zhu et al. [190]	RF, SVM		
A. Altaher [191]		Neuro-fuzzy inference system	
Su et al. [192]		DBN, SVM	
Mahindru et al. [193]	SVM, NB, RF, LR, K-Means, FF, FC, DB, Bayesnet, Adaboost, DT, KNN, J48 DT	MLP, DNN, SOM, Y-J48, Y-SMO, Y-MLP, best training ensemble approach (BTE), majority voting ensemble approach (MVE) and nonlinear ensemble decision tree forest approach (NDTF)	
Dehkordy et al. [194]	SVM, KNN, ID3 DT		
Nguyen et al. [195]	SVM	DNN	
Taheri et al. [196]	SVM	Robust NN, C4N CNN	
Mahesh et al. [197]		CNN, ANN , DBN, Method level behavioral semantic approach (MLBS) DE-CNN	CNN-Adaptive Red Fox Optimization (CNN-ARFO)
Firdaus et al. [198]	RF, Bayesnet, SVM	MLP, Voted perceptron (VP), Radial basis function network (RBFN) ANN	
Shrivastava et al. [199]			Frequency-based rules defined over permissions and intents to predict malicious risk level of an application
Varsha et al. [200]	RF, Rotation forest, SVM		
M. Deypir [201]			Entropy-based method to calculate information gain score for a permission and correspondingly risk score for an application
Mahindru et al. [202]	K-mean, SVM(linear, poly, RBF), DT, LR, NB	NN, SOM, FF, FC and DB	
Keyvanpour et al. [203]	KNN, SL, NB, J48 DT, RT, RF		
Razak et al. [204]	RF, J48 DT, KNN, AdaBoost	MLP	
Xie et al. [205]	SVM		
Mahindru et al. [206]	LOGR, SVM, NB, DT	Radial Basis Function Neural Network RBFM-KCM, RBFN-FCM, RBFN-RAN, ANN, with three different ensemble methods (i.e., BTE, MVE, and NDTF	
Alecakir et al. [18]		RNN GRU, MLP	
Ali et al. [207]	SVM (Linear, RBF, Poly), LR		
Sun et al. [208]	HMM, GD, SVM (Linear, RBF, Poly, KCD)		
AlJarrah et al. [209]	RF, SVM, LR, NB, KNN, DT		
Gharib et al. [210]	SVM, RF, NB, AdaBoost	DNN, DAE	
Sun et al. [211]	RF, DT, SVM		Positive and Unlabeled learning

classifiers mentioned in Figure 4, researchers have developed their algorithm, too, instead of using the traditional classifiers. For instance, in [55], Arora et al. proposed their algorithm to form graphs based on permission patterns and frequency as weights. **Hence, based on the results of the empirical evidence, we answer the fourth Research Question that the most commonly used ML models are Support Vector Machines, Decision Trees, and Random Forest due to their convenient and simple working but at the same time, due to the limitations of traditional supervised learning methods, one can assume that the popularity of neural network models will surely rise.**

6 Datasets used

Researchers and practitioners have adopted numerous malicious datasets in the past decade to carry out their research on Android malware. The datasets can be broadly divided into two types based on their data source: *in-the-lab* datasets and *in-the-wild* datasets. Traditional datasets such as Drebin[5] and Genome[6] are considered as baselines in the field of Android malware detection and fall under the category of *in-the-lab* datasets. On the other hand, datasets such as Virustotal[7], Virusshare[8], Contagio[9], AMD [215], McAfee[10] and Androzoo[11] are called *in-the-wild* datasets due to their continuously updating nature, and that is the reason why it's obvious that the performance of *in-the-wild* datasets is found to be more credible than that of *in-the-lab* datasets and in addition to the datasets mentioned above some researchers also utilized kaggle[12] and Anzhi[13] type datastores too. Apart from the datasets mentioned above, the datasets provided by the Canadian Institute of Cybersecurity[14] such as the CICInvesAndMal2019, CICAndMal2017, and CIC2020 have also been utilized by the researchers.

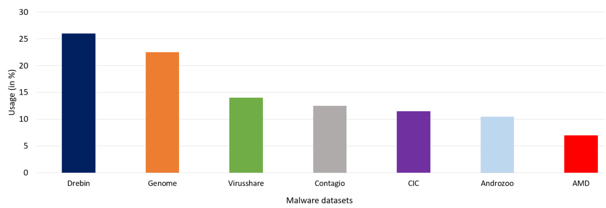

Fig. 5: Statistics depicting the most commonly used malware datasets

In response to RQ5, we present Table 7 as it summarizes the malware datasets used by researchers and practitioners to form their analysis/detection model. As seen from the table, most researchers have used traditional malware

Table 7: Dataset used

Related works	**Malware datasets used**
Zhang et al. [19]	Genome
Li et al. [20]	Anzhi Store
Sahin et al. [21]	[216]
Talha et al. [22]	Contagio, Drebin, Genome
Varma et al. [23]	[217]
Mahindru et al. [24]	[218], [219], Genome, and AndroMalShare[15]
Dogru et al. [25]	Drebin
Rathore et al. [26]	Drebin
Shang et al. [27]	-
Tchakounte et al. [28]	CIC-AndMal2017
Ju et al. [16]	-
Ilham et al. [29]	AMD
Sahin et al. [30]	Kaggle
Angelo et al. [31]	Unisa malware dataset (UMD)[16]
Xiong et al. [32]	[217]
Lu at el. [33]	Baidu application market[17], Genome, Virusshare
Kavitha et al. [34]	-
E. Amer [35]	Drebin, Genome
Chakravarty et al. [36]	AMD
Pondugula et al. [37]	For training data from IEEE DataPort, and sites like droidbench
Sahal et al. [38]	Contagio, Genome
Tuan Mat et al. [39]	Androzoo and Drebin
Wang et al. [40]	Virusshare, Genome
Park et al. [41]	AndroZoo
Liang et al. [42]	Genome
Enck et al. [14]	-
Enck et al. [17]	Android Market
Wang et al. [43]	Genome
Peng et al. [44]	Genome
Pandita et al. [45]	-
Samra et al. [46]	-
Yerima et al. [47]	Genome
Aung et al. [48]	-
Yerima et al. [49]	Genome
Sanz et al. [50]	VirusTotal
Moonsamy et al. [51]	Genome
Backes et al. [52]	-
Wu et al. [53]	Contagio
Kato et al. [54]	Drebin, Androzoo, VirusShare
Arora et al. [55]	Genome, Drebin, Koodous
Alsoghyer et al. [56]	HelDroid [220], RansomProber project[212], VirusTotal, Koodous
Saleem et al. [57]	Drebin
Ghasempour et al. [58]	Androzoo
Shrivastava et al. [59]	Drebin
Upadhayay et al. [60]	Genome, Drebin, Koodous
Lee et al. [61]	CICAndMal 2017
Surendran et al. [62]	Drebin, AMD, AndroZoo, external repositories
A. T. Kabakus [63]	Drebin ,KuafuDet[18] ,AndroZoo, VirusShare
Wang et al. [64]	Huawei application market[19], Xiaomi application market[20], 360 application market[21], Wandoujia application market[22]
Akbar et al. [65]	VirusShare
Zhu et al. [66]	VirusShare
Wang et al. [67]	Andro_dumpsys

[5] http://www.sec.cs.tu-bs.de/ danarp/Drebin/
[6] http://www.malgenomeproject.org/
[7] https://www.virustotal.com
[8] https://virusshare.com/
[9] http://contagiominidump.blogspot.hk/
[10] https://home.mcafee.com/Default.aspx?rfhs=1
[11] https://androzoo.uni.lu/
[12] https://www.kaggle.com
[13] http://www.anzhi.com/
[14] https://www.unb.ca/cic/datasets/index.html

Related works	Malware datasets used
N. McLaugling [68]	McAfee, vendor's internal dataset
Wang et al. [69]	Mal_com1, Mal_com2 and Mal_Zhou [220]
Grace et al. [70]	Github
Liu et al. [71]	VirusShare
Bayazit et al. [72]	CICInvesAndMal2019
Lee et al. [73]	Andro-AutoPsy Dataset [221]
Zhu et al. [74]	MUDFLOW [222], VirusShare
Almahmoud et al. [75]	CIC-AndMal2017, CIC-InvesAndMal2019, CIC-MalDroid2020
Feng et al. [76]	CICAndMal2017
Kandu et al.[77]	Genome
Arora et al. [78]	Genome
Ding et al. [79]	CICInvesAndMal2019
Sahin et al. [80]	M0Droid[223], AMD, Kaggle, [224]
Idrees et al. [81]	Contagio, Drebin, Genome, Virus Total, theZoo, MalShare, VirusShare
Khariwal et al.[82]	Genome, Drebin, Koodous
Idrees et al. [83]	Contagio, VirusTotal, appsapk, Android-mob
Zhu et al. [15]	VirusShare
Bai et al. [84]	Drebin
Taheri et al. [85]	Drebin, Contagio, Genome
Alazab et al. [86]	AndroZoo, Contagio, MalShare, VirusShare, VirusTotal
Mathur et al. [87]	Androzoo, AMD
Imtiaz et al. [88]	CICInves AndMal2019
Liu et al. [89]	OmniDroid, CIC2019, CIC2020
Chen et al. [90]	VirusShare
Guan et al. [91]	VirusShare
Mohamed et al. [92]	Genome, Maldroid
Varma et al. [93]	CICInvesAnd Mal2019
Gyunka et al.[94]	Genome, Contagio
Taha et al. [95]	Drebin
Peng et al. [96]	CICMalDroid 2020, CIC-InvesAndMal 2019, Drebin
Ashwini et al. [97]	Drebin
Jiang et al. [98]	Genome, Andro MalShare
Wang et al. [99]	Information Security Lab of Peking University
Rana et al. [100]	Drebin
Lu et al. [101]	VirusShare, Genome, Contagio
Millar et al. [102]	Genome, Drebin, vendor's internal repository, AMD
Barrera et al. [103]	No malware
Shabtai et al. [104]	No malware
Felt et al. [105]	No malware
Erickson et al. [106]	No malware
Sarma et al. [107]	Contagio
Frank et al. [108]	No malware
Jhu et al. [109]	Steamy Window[23]
Peiravian et al. [110]	[217]
Sanz et al. [111]	VirusTotal
Feldman et al. [112]	Contagio
Pehlivan et al. [113]	COMODO Security Solutions, Inc., a private security company[24]
Rahman et al. [114]	Kharon Malware Dataset[25], CICInvesAndMal2019, MalwareBazaar[26], Github, other platforms

Related works	Malware datasets used
Rovelli et al. [115]	Genome, Contagio
Arp et al. [116]	Genome,FakeInstaller, GoldDream[27], GingerMaster[28], DroidKungFu[29]
Yerima et al. [117]	McAfee
Kang et al. [118]	VirusShare, Contagio, Malware.lu
Zhao et al. [119]	Drebin
Qiao et al. [120]	Genome
Chen et al. [121]	360 APKs[30], MobiSec Lab Website[31], [217]
Demertzis et al. [122]	Magnum- Research[32]
Verma et al. [123]	Contagio, malware forums , security blogs, Genome
Wang et al. [124]	VirusTotal
Tang et al. [125]	Genome, Drebin
Wang et al. [126]	Drebin, Genome
Li et al. [127]	Drebin
Bhattacharya et al. [128]	Contagio
Xie et al. [129]	Genome, VirusShare, Drebin
Xie et al. [130]	Genome, VirusShare, Drebin, antivirus companies
Ren et al. [131]	Anzhi, AndroTotal, Drebin
Tao et al. [132]	VirusShare, Contagio
Namrud et al. [133]	AndroZoo
Alswaina et al. [134]	-
Qiu et al. [135]	-
Zhu et al. [136]	ViruShare
Feng et al. [137]	No Malware
Aonzo et al. [138]	AndroZoo
Urooj et al. [139]	MalDroid [225], DefenseDroid[33] and a small own generated dataset
Wang et al. [140]	No malware
Wang et al. [141]	FakeInst, Opfake, FakeInstaller, DroidKungFu, GinMaster, Plankton
Zhang et al. [142]	No malware
Kesswani et al. [143]	No malware
Ibrahim et al. [144]	CICMalDroid 2020
Arshad et al. [145]	Drebin
Yuan et al. [146]	Genome, Contagio
Zhou et al. [147]	Genome
Cilleruelo et al. [148]	Malware selected on the basis of lifespan criteria from Google Play Store
Firdaus et al. [149]	Drebin
Wang et al. [150]	[217]
Singh et al. [151]	Drebin, polymorphic and metamorphic malware dataset
Rafiq et al. [152]	Drebin, AMD, Androzoo
Mahdavifar et al. [153]	VirusTotal, Contagio, AMD, CICMalDroid202
Seraj et al. [154]	Applications classified as harmful via VirusTotal and downloaded from Google Play Store
Mahindru et al. [155]	Didn't mention properly
Sahin et al. [21]	VirusShare
Anupama et al. [156]	Drebin
Chen et al. [157]	AndroZoo
Mahindru et al. [158]	Genome, [218]
Tchakounté et al. [159]	AMD
Nissim et al. [160]	Contagio, Genome, third party application stores based in Russia, China, and Europe
Peynirci et al. [161]	Genome, Drebin, AndroZoo
Nauman et al. [162]	Drebin, VirusShare
Bhattacharya et al. [163]	Wang's repository dataset[34], Contagio, KEEL/UCI repository
Bao et al.[164]	No malware
Medrano et al. [165]	No malware
Mat et al. [165]	AndroZoo, Drebin
Shatnawi et al. [166]	CIC InvesAndMal2019

Related works	Malware datasets used
Smmarwar et al. [167]	CICInvesAndMal2019
Arif et al. [168]	AndroZoo, Drebin
Manzanares et al. [169]	Drebin, AMD, VirusTotal, VirusShare
Bhat et al. [170]	Virustotal, VirusShare, Drebin
Elayan et al.[171]	CICAndMal2017
Syrris et al. [172]	Drebin
Idrees et al. [173]	Contagiodump, Genome, Virus Total, theZoo, MalShare, VirusShare
Rehman et al. [174]	M0DROID
Martin et al. [175]	Koodous, AndroZoo
Navarro et al. [176]	AndroZoo, [217] , VirusShare, AndroMalShare
Milosevic et al. [177]	M0Droid
Alzaylaee et al. [178]	McAfee
Cai et al. [179]	AMD, Drebin
Badhani et al. [180]	Andro- DumpSys, AndroZoo, Contagio , AndroMalShare , AMD, VirusTotal
Hijawi et al. [181]	[218]
Sheen et al. [182]	Genome
Nisha et al. [183]	AndroZoo, VirusShare, AndroMalShare, PRAGuard[226]
Song et al. [184]	Not mentioned
Zhang et al. [185]	Genome
Yang et al. [186]	No malware
Thiyagarajan et al. [187]	AndroZoo
Qaisar et al. [188]	Android PRAGuard, Drebin, Open-source apps, Kharon, Androzoo, CICAAGM
Appice et al. [189]	Alternative Chinese and Russian Markets, Android websites, malware forums, security blogs, Genome
Zhu et al. [190]	VirusShare
A. Altaher [191]	Genome
Su et al. [192]	Drebin, Genome, Contagio
Mahindru et al. [193]	Genome , AndroMalShare, [218]
Dehkordy et al. [194]	Drebin, AMD
Nguyen et al. [195]	AMD, Drebin
Taheri et al. [196]	Drebin, Contagio, Genome
Mahesh et al. [197]	Private companies
Firdaus et al. [198]	Drebin, Genome
Shrivastava et al. [199]	Third-party applications
Varsha et al. [200]	Drebin
M. Deypir [201]	[217], Security expert and World Wide Web
Mahindru et al. [202]	Genome, AndroMalShare, [218]
Keyvanpour et al. [203]	Drebin
Razak et al. [204]	Drebin
Xie et al. [205]	Anzhi
Mahindru et al. [206]	[218], [219], Sanddroid
Alecakir et al. [18]	No malware
Ali et al. [207]	No malware
Sun et al. [208]	Contagio
AlJarrah et al. [209]	CICMalDroid2020
Gharib et al. [210]	R-PackDroid[35], HellDroid[220], Contagio, Koodous
Sun et al. [211]	Drebin

datasets in their studies, such as Genome and Drebin, whereas some have shifted their focus to other alternatives from the *in-the-wild* dataset category, namely Contagio. Figure 5 gives us a better understanding of Table 7 by plotting the top seven malware datasets in terms of usage in %, i.e., the percentage of studies that have utilized the corresponding malware dataset from the total lot of research papers used in our review. The statistics depict the preference of researchers towards the traditional datasets as Drebin [95]-[97] stands at the top with 26% usage, followed by Genome [103]-[104] at 22.5%. *In-the-wild* datasets such as Virusshare [90]-[91], Contagio [94] and Androzoo [86]-[87] stand at third, fourth and sixth rank with 14%, 12.5% and 10.5% share respectively. As discussed above, the popularity of the datasets provided by the Canadian Institute of Cybersecurity [166]-[167] due to the concise pattern of information in datasets, and they take up almost 11.5% of the comprehensive studies taken up in this review. **Hence, based on the results presented above, we answer the fifth Research Question that the *In-the-lab* datasets such as Drebin and Genome occupy the largest proportion of the most preferred malware dataset by researchers.**

7 Challenges and Limitations

We answer our sixth Research Question in this and the subsequent section, wherein we describe a few limitations of proposed models mainly associated with permissions as a feature in the field of Android security followed by proposing some future research directions.

7.1 Drawbacks of using permissions as a feature

Static techniques are generally inexpensive in terms of complexity compared to dynamic approaches, as static features can be more easily extracted than dynamic ones. However, static methods have a few disadvantages, and consequently, models built over Android permissions are generally found incapable of recognizing the stealthier behavior of code obfuscation and dynamic code loading. As a result, some malicious apps may incorporate stealthy behavior and evade permissions-based detection. Additionally, the inability of many detection models to track behaviors that do not cause permission checks is another limitation of using permissions as a feature [19], [185].

7.2 Faulty Research dataset

The training and validation dataset is the foundation for any Android malware analysis/detection model. Faulty research datasets, biased to one class, i.e., either benign or malware, outdated, or insufficient in size or distribution, can produce inaccurate results in terms of meaning and poor detection accuracy. It has been noticed that app developers often make careless spelling mistakes. Applications are found with no permission requests or faulty formatting of manifest files. These reasons surely cause hindrances to research. Results with insufficient data are less efficient and accurate because of the lack of training data. Even if the detector performs well in the experimental tests, it might still prove to be inefficient in real life. Hence, building a good and informative dataset of Android applications that request the actual and essentially needed permissions or features is an important pre-requisite step for any model. One of the main reasons for accumulating faulty applications in one's dataset is the open-source nature of the Android operating system itself. Users are given the freedom to download applications not only from the mainstream market but also from informal third-party markets and unknown malicious websites, which might result in the mixing of benign and malware application sets.

7.3 Feature Extraction and Processing

Gathering the right application samples is important; however, knowing how and which features to extract is itself a very extensive research topic. As described in the previous section, despite the numerous advantages of using static features, static analysis and consequently permission-based techniques are also found to have some disadvantages At the same time, there are also many challenges to feature extraction with dynamic analysis. For example, how to ensure that all malicious behaviors can be triggered during dynamic analysis and which emulator to use to produce real-time usage patterns of a user are issues that need to be solved.

The summary and statistics presented in section V indicate that most of the research is still based on traditional analysis features, such as permissions, because of their strong training effect, vast size, and prominent relationship with class labels and behavior.

7.4 Application of Machine Learning

Machine learning methods are used in the field of Android malware detection to improve the effectiveness and efficiency of detection. However, because of the phenomena such as concept drift, the performance of traditionally trained classifiers has started to decline. Concept drift, which is a problem that arises with the continuous development and evolution of Android malware, has caused researchers and practitioners to shift their focus toward neural network-based detection methods. For example, Bayazit et al. [72] chose static analysis to build their various RNN-based classifiers. The features that they used were permissions and intents. In the end, they compared the detection results of their various RNN-based algorithms to determine that the Bidirectional Long Short-Term Memory (BiLSTM) model outperforms all. Moreover, neural networks can be applied to bridge the gap between the semantic representation of APKs and Android malware detection. Hence, we conclude that neural networks have potential in the field of Android malware detection using static analysis. The preference of researchers and practitioners is starting to tilt more towards deep learning classifiers while building their analysis/detection models because of the added advantages compared with the traditional machine learning classifiers. However, deep learning algorithms can be computationally expensive and may require many resources to be successfully trained. This can be seen as a major barrier, especially for people who want access to high-performance computing services but lack the necessary resources.

7.5 Unreasonable reliability over users

The majority of the proposed models in the field of Android malware detection lie under the category of *Off-Device* models, i.e., they rely upon the user to provide the application data and then return the malware report of the corresponding applications after performing their proposed approach which is, in fact, a tedious and long method. Instead, *On-Device* models are the need of the hour, capable of performing malware analysis/detection using the user's smartphone to generate instant malware reports.

7.6 Application Collusion

Application collusion is an emerging threat to Android-based devices that seem almost immune to permissions-based detection systems. In app collusion, two or more Android apps somehow collude to perform a malicious action that they cannot accomplish independently [55]. In this way, they perform malicious tasks without showing malware behavior.

7.7 Zero day malware

Zero-day malware is the kind of threat for which no patch exists yet and therefore seems immune to traditional signature-based analysis that identifies the dangerous permissions usage patterns. In simple words, permission-based malware detection strategies are based on knowledge of the vulnerability, exploit in question, or analyzing behavioral patterns of application classes, which obviously is not available for zero-day threats. As a result, certain methods for mitigating these threats are ineffective.

8 Future Directions

In this section, we provide some directions for future work in the Android malware analysis/detection field using Android permissions based on our investigation and review.

1. The Android OS has more than 150 pre-defined permissions. Moreover, as applications become more complex with time, this number is bound to grow only. Permissions are one of the key components present within any Android application's manifest file and have been widely used in the literature for Android malware detection. However, the usage pattern of permissions between normal and malware applications is generally quite similar. For instance, we have collected 77,000 normal apps and an equal number of malware apps from Androzoo. Furthermore, we have extracted permissions from the manifest files of corresponding applications. Table 8 shows that 13 of the top 20 permissions are common in normal and malware datasets. If all these permissions are considered features while building an analysis or detection model, poor detection accuracy results are inevitable. Table 9 helps us to further prove our point by summarizing the detection results considering all features, i.e., the total lot of 129 permissions, for detection without performing any feature selection/ ranking/ other technique. As seen from the Table, the highest detection accuracy achieved by using the BC classifier is merely 78.64%, which is too low in Android security for any malware detection model. Hence, priority should be given to feature selection or ranking to filter out only the most relevant and influential set of features in the future.
2. Combining multiple static analysis techniques or features can always be beneficial when it comes to comparison with using just a single feature set. Permissions are known to be manually designed and generally are coarse-grained in nature. Intents possess similar characteristics like permissions, whereas opcode-based methods are found to be capable of capturing contextual semantics of applications but they both fail to produce structural semantics. Functional call graphs do the needful by extracting structural semantics. Hence, we can say that using two or more types of static features in combination might further improve the experimental results and thus a model's detection accuracy.
3. Since stealthier malware may evade permissions-based static detection, several works in the literature aim to use dynamic features such as network traffic and system calls to detect such stealthier malware. However, analyzing dynamic features is complex and time-consuming; therefore, hybrid detection models serve the purpose as they combine both

Table 8: Top 20 most frequently requested permissions from both normal and malware datasets with their corresponding frequency.

PERMISSIONS	**Normal Frequency**	**PERMISSIONS**	**Malware Frequency**
INTERNET	55063	INTERNET	55684
ACCESS_NETWORK_STATE	52391	ACCESS_NETWORK_STATE	55252
WRITE_EXTERNAL_STORAGE	38934	WRITE_EXTERNAL_STORAGE	54759
WAKE_LOCK	32527	ACCESS_WIFI_STATE	53886
ACCESS_WIFI_STATE	28554	READ_PHONE_STATE	53586
RECEIVE	23875	READ_EXTERNAL_STORAGE	46646
READ_EXTERNAL_STORAGE	22516	WAKE_LOCK	44003
VIBRATE	20472	GET_TASKS	43399
ACCESS_FINE_LOCATION	16968	CHANGE_WIFI_STATE	43165
ACCESS_COARSE_LOCATION	16650	ACCESS_COARSE_LOCATION	42425
RECEIVE_BOOT_COMPLETED	16519	VIBRATE	42325
CAMERA	14993	MOUNT_UNMOUNT_FILESYSTEMS	41324
READ_PHONE_STATE	14176	ACCESS_FINE_LOCATION	40720
C2D_MESSAGE	12342	WRITE_SETTINGS	39497
BIND_GET_INSTALL_REFERRER_SERVICE	10593	SYSTEM_ALERT_WINDOW	38594
BILLING	9905	CAMERA	36115
FOREGROUND_SERVICE	9587	CHANGE_NETWORK_STATE	30874
GET_ACCOUNTS	7806	RECEIVE_BOOT_COMPLETED	29441
WRITE_SETTINGS	7258	READ_LOGS	29112
BLUETOOTH	5820	RECORD_AUDIO	27010

Table 9: Detection results considering all features for detection without performing any feature selection/ ranking/ other technique

FEATURES used	**Detection accuracy using various machine learning and deep learning classifiers (in %)**					
	DT	**RF**	**ANN**	**BC**	**NB**	**LR**
All Permissions (129)	74.64	74.64	69.55	78.64	69.60	69.60

static and dynamic elements. Because of static features like permissions in hybrid models, some malware samples can be easily detected. And dynamic features can help detect stealthier samples. Hence, hybrid models may be better for effective Android malware detection.

4. Moreover, as discussed in [143], the malware authors tend to request more permissions than their application requires for its functioning to cover up or hide their malicious intent. Due to the absence of any robust permission check system that provides an in-depth comparison of the permissions requested and required, malware authors exploit it as much as possible. Hence, more research is required on this to fulfill the absence of such a permission check system by the OS shortly to reduce the malware attacks at the initial level only.
5. Currently, repositories like Androzoo and CIC provide standardized application dataset samples and libraries with comprehensive background information and essential filters that solve the faulty dataset issue up to some extent.

9 Conclusion

This paper summarizes the state-of-the-art studies and comprehensively reviews the research trend in Android malware analysis/detection using permissions as a key feature. Concretely, this Comprehensive Literature Review (CLR) is conducted with the close observation and study of 200 research papers ranging from the advent of Android in 2009 to the current research scenario in 2023, covering almost 14 years of research history.

Based on the six research questions put forward by us to understand the relevance of permissions as a means to analyze/detect Android malware, this CLR investigates the purpose of research (analysis or analysis combined with detection), the technique used to select or utilize or rank the extracted features, most commonly used features alongside with permissions, the malware datasets utilized by the researchers/ practitioners and lastly, the limitations, challenges in the field of Android malware analysis/detection in order to propose some future research directions.

According to this CLR, we answered the six research questions mentioned above and conclude that 1) Majority of researchers and practitioners have chosen the path of building Android malware detection models instead of merely analysing the malware; 2) The most popular techniques to handle the extracted features are the ML/DL classifiers as their hyper parameters can be easily tuned to reduce, rank, select even the unlabelled or unbalanced feature set, followed by statistical test such as Chi-square, Mann-Whitney test and many more; 3) Researchers have preferred using API calls and intents the most in combination with permissions while forming their Android malware analysis/detection models; 4) Based on the results of empirical evidence, we conclude that the most commonly used ML models are SVM, DT and RF due to their convenient and simple working but at the same time, due to the limitations of traditional supervised learning methods, one can assume that the popularity of neural network models will surely rise; 5) *In-the-lab* datasets such as Drebin and Genome occupy the largest proportion of the most preferred malware dataset by researchers.

Compliance with ethical standards

Conflict of interest - The authors declare that they have no conflict of interest.
Ethical approval - This article does not contain any studies with human participants or animals performed by any of the authors.
Research Data Policy and Data Availability - The authors declare that the data supporting the findings of this study are available within the paper.

References

[1] Lubuva, H., Huang, Q., Msonde, G.C.: A review of static malware detection for android apps permission based on deep learning. International Journal of Computer Networks and Applications **6**(5), 80–91 (2019)

[2] Alqahtani, E.J., Zagrouba, R., Almuhaideb, A.: A survey on android malware detection techniques using machine learning algorithms. In: 2019 Sixth International Conference on Software Defined Systems (SDS), pp. 110–117 (2019). IEEE

[3] Naway, A., Li, Y.: A review on the use of deep learning in android malware detection. arXiv preprint arXiv:1812.10360 (2018)

[4] Hahn, S., Protsenko, M., Müller, T.: Comparative evaluation of machine learning-based malware detection on android. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit (2016)

[5] Kouliaridis, V., Barmpatsalou, K., Kambourakis, G., Chen, S.: A survey on mobile malware detection techniques. IEICE Transactions on Information and Systems **103**(2), 204–211 (2020)

[6] Qamar, A., Karim, A., Chang, V.: Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems **97**, 887–909 (2019)

[7] Hamed, Y.S.I., AbdulKader, S.N.A., Mostafa, M.-S.M.: Mobile malware detection: A survey. International Journal of Computer Science and Information Security (IJCSIS) **17**(1) (2019)

[8] Yan, P., Yan, Z.: A survey on dynamic mobile malware detection. Software Quality Journal **26**(3), 891–919 (2018)

[9] Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R., Maskeliunas, R.: Android malware detection: A survey. In: Applied Informatics: First International Conference, ICAI 2018, Bogotá,

Colombia, November 1-3, 2018, Proceedings 1, pp. 255–266 (2018). Springer

[10] BalaGanesh, D., Chakrabarti, A., Midhun-chakkaravarthy, D.: Smart devices threats, vulnerabilities and malware detection approaches: a survey. European Journal of Engineering and Technology Research **3**(2), 7–12 (2018)

[11] Gyamfi, N.K., Owusu, E.: Survey of mobile malware analysis, detection techniques and tool. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 1101–1107 (2018). IEEE

[12] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., Liu, H.: A review of android malware detection approaches based on machine learning. IEEE Access **8**, 124579–124607 (2020)

[13] Pan, Y., Ge, X., Fang, C., Fan, Y.: A systematic literature review of android malware detection using static analysis. IEEE Access **8**, 116363–116379 (2020)

[14] Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE security & privacy **7**(1), 50–57 (2009)

[15] Zhu, H.-j., Gu, W., Wang, L.-m., Xu, Z.-c., Sheng, V.S.: Android malware detection based on multi-head squeeze-and-excitation residual network. Expert Systems with Applications **212**, 118705 (2023)

[16] Ju, S.-h., Seo, H.-s., Kwak, J.: Research on android malware permission pattern using permission monitoring system. Multimedia Tools and Applications **75**, 14807–14817 (2016)

[17] Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 235–245 (2009)

[18] Alecakir, H., Can, B., Sen, S.: Attention: there is an inconsistency between android permissions and application metadata! International Journal of Information Security, 1–19 (2021)

[19] Zhang, Y., Yang, M., Yang, Z., Gu, G., Ning, P., Zang, B.: Permission use analysis for vetting undesirable behaviors in android apps. IEEE transactions on information forensics and security **9**(11), 1828–1842 (2014)

[20] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., Ye, H.: Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics **14**(7), 3216–3225 (2018)

[21] Şahin, D.Ö., Kural, O.E., Akleylek, S., Kılıç, E.: A novel permission-based android malware detection system using feature selection based on linear regression. Neural Computing and Applications, 1–16 (2021)

[22] Talha, K.A., Alper, D.I., Aydin, C.: Apk auditor: Permission-based android malware detection system. Digital Investigation **13**, 1–14 (2015)

[23] Varma, P.R.K., Raj, K.P., Raju, K.S.: Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms. In: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 294–299 (2017). IEEE

[24] Mahindru, A., Singh, P.: Dynamic permissions based android malware detection using machine learning techniques. In: Proceedings of the 10th Innovations in Software Engineering Conference, pp. 202–210 (2017)

[25] Doğru, İ.A., Önder, M.: Appperm analyzer: malware detection system based on android permissions and permission groups. International Journal of Software Engineering and Knowledge Engineering **30**(03), 427–450 (2020)

[26] Rathore, H., Sahay, S.K., Rajvanshi, R.,

Sewak, M.: Identification of significant permissions for efficient android malware detection. In: International Conference on Broadband Communications, Networks and Systems, pp. 33–52 (2020). Springer

[27] Shang, F., Li, Y., Deng, X., He, D.: Android malware detection method based on naive bayes and permission correlation algorithm. Cluster Computing **21**(1), 955–966 (2018)

[28] Tchakounté, F., Wandala, A.D., Tiguiane, Y.: Detection of android malware based on sequence alignment of permissions. Int. J. Comput.(IJC) **35**(1), 26–36 (2019)

[29] Ilham, S., Abderrahim, G., Abdelhakim, B.A.: Permission based malware detection in android devices. In: Proceedings of the 3rd International Conference on Smart City Applications, pp. 1–6 (2018)

[30] Şahın, D.Ö., Kural, O.E., Akleylek, S., Kiliç, E.: New results on permission based static analysis for android malware. In: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), pp. 1–4 (2018). IEEE

[31] D'Angelo, G., Palmieri, F., Robustelli, A.: A federated approach to android malware classification through perm-maps. Cluster Computing **25**(4), 2487–2500 (2022)

[32] Xiong, P., Wang, X., Niu, W., Zhu, T., Li, G.: Android malware detection with contrasting permission patterns. China Communications **11**(8), 1–14 (2014)

[33] Lu, T., Hou, S.: A two-layered malware detection model based on permission for android. In: 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET), pp. 239–243 (2018). IEEE

[34] Kavitha, K., Salini, P., Ilamathy, V.: Exploring the malicious android applications and reducing risk using static analysis. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 1316–1319 (2016). IEEE

[35] Amer, E.: Permission-based approach for android malware analysis through ensemble-based voting model. In: 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pp. 135–139 (2021). IEEE

[36] Chakravarty, S., *et al.*: Feature selection and evaluation of permission-based android malware detection. In: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), pp. 795–799 (2020). IEEE

[37] Sirisha, P., Anuradha, T., *et al.*: Detection of permission driven malware in android using deep learning techniques. In: 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 941–945 (2019). IEEE

[38] Sahal, A.A., Alam, S., Soğukpinar, I.: Mining and detection of android malware based on permissions. In: 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 264–268 (2018). IEEE

[39] Mat, S.R.T., Ab Razak, M.F., Kahar, M.N.M., Arif, J.M., Zabidi, A.: Applying bayesian probability for android malware detection using permission features. In: 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), pp. 574–579 (2021). IEEE

[40] Wang, Z., Li, K., Hu, Y., Fukuda, A., Kong, W.: Multilevel permission extraction in android applications for malware detection. In: 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1–5 (2019). IEEE

[41] Park, J., Kang, M., Cho, S.-j., Han, H., Suh, K.: Analysis of permission selection techniques in machine learning-based malicious app detection. In: 2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE),

pp. 92–99 (2020). IEEE

[42] Liang, S., Du, X.: Permission-combination-based scheme for android mobile malware detection. In: 2014 IEEE International Conference on Communications (ICC), pp. 2301–2306 (2014). IEEE

[43] Wang, Y., Zheng, J., Sun, C., Mukkamala, S.: Quantitative security risk assessment of android permissions and applications. In: Data and Applications Security and Privacy XXVII: 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Newark, NJ, USA, July 15-17, 2013. Proceedings 27, pp. 226–241 (2013). Springer

[44] Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Using probabilistic generative models for ranking risks of android apps. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 241–252 (2012)

[45] Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T.: {WHYPER}: Towards automating risk assessment of mobile applications. In: 22nd USENIX Security Symposium (USENIX Security 13), pp. 527–542 (2013)

[46] Samra, A.A.A., Yim, K., Ghanem, O.A.: Analysis of clustering technique in android malware detection. In: 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 729–733 (2013). IEEE

[47] Yerima, S.Y., Sezer, S., McWilliams, G., Muttik, I.: A new android malware detection approach using bayesian classification. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 121–128 (2013). IEEE

[48] Zarni Aung, W.Z.: Permission-based android malware detection. International Journal of Scientific & Technology Research **2**(3), 228–234 (2013)

[49] Yerima, S.Y., Sezer, S., McWilliams, G.: Analysis of bayesian classification-based approaches for android malware detection. IET Information Security **8**(1), 25–36 (2014)

[50] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P.G., Álvarez, G.: Puma: Permission usage to detect malware in android. In: International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions, pp. 289–298 (2013). Springer

[51] Moonsamy, V., Rong, J., Liu, S.: Mining permission patterns for contrasting clean and malicious android applications. Future Generation Computer Systems **36**, 122–132 (2014)

[52] Backes, M., Gerling, S., Hammer, C., Maffei, M., Styp-Rekowsky, P.: Appguard–enforcing user requirements on android apps. In: Tools and Algorithms for the Construction and Analysis of Systems: 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings 19, pp. 543–548 (2013). Springer

[53] Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P.: Droidmat: Android malware detection through manifest and api calls tracing. In: 2012 Seventh Asia Joint Conference on Information Security, pp. 62–69 (2012). IEEE

[54] Kato, H., Sasaki, T., Sasase, I.: Android malware detection based on composition ratio of permission pairs. IEEE Access **9**, 130006–130019 (2021)

[55] Arora, A., Peddoju, S.K., Conti, M.: Permpair: Android malware detection using permission pairs. IEEE Transactions on Information Forensics and Security **15**, 1968–1982 (2019)

[56] Alsoghyer, S., Almomani, I.: On the effectiveness of application permissions for android ransomware detection. In: 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), pp. 94–99

(2020). IEEE

[57] Saleem, M.S., Mišić, J., Mišić, V.B.: Examining permission patterns in android apps using kernel density estimation. In: 2020 International Conference on Computing, Networking and Communications (ICNC), pp. 719–724 (2020). IEEE

[58] Ghasempour, A., Sani, N.F.M., Ovye, J.A.: Permission extraction framework for android malware detection. International Journal of Advanced Computer Science and Applications **11**(11) (2020)

[59] Shrivastava, G., Kumar, P.: Sensdroid: analysis for malicious activity risk of android application. Multimedia Tools and Applications **78**(24), 35713–35731 (2019)

[60] Upadhayay, M., Sharma, A., Garg, G., Arora, A.: Rpndroid: android malware detection using ranked permissions and network traffic. In: 2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4), pp. 19–24 (2021). IEEE

[61] Lee, S.-A., Yoon, A.-R., Lee, J.-W., Lee, K.: An android malware detection system using a knowledge-based permission counting method. JOIV: International Journal on Informatics Visualization **6**(1), 138–144 (2022)

[62] Surendran, R., Thomas, T., Emmanuel, S.: A tan based hybrid model for android malware detection. Journal of Information Security and Applications **54**, 102483 (2020)

[63] Kabakus, A.T.: Droidmalwaredetector: A novel android malware detection framework based on convolutional neural network. Expert Systems with Applications **206**, 117833 (2022)

[64] Wang, X., Zhang, L., Zhao, K., Ding, X., Yu, M.: Mfdroid: A stacking ensemble learning framework for android malware detection. Sensors **22**(7), 2597 (2022)

[65] Akbar, F., Hussain, M., Mumtaz, R., Riaz, Q., Wahab, A.W.A., Jung, K.-H.: Permissions-based detection of android malware using machine learning. Symmetry **14**(4), 718 (2022)

[66] Zhu, H.-J., You, Z.-H., Zhu, Z.-X., Shi, W.-L., Chen, X., Cheng, L.: Droiddet: effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing **272**, 638–646 (2018)

[67] Wang, W., Ren, C., Song, H., Zhang, S., Liu, P., et al.: Fgl_droid: an efficient android malware detection method based on hybrid analysis. Security and Communication Networks **2022** (2022)

[68] McLaughlin, N.: Malceiver: Perceiver with hierarchical and multi-modal features for android malware detection. arXiv preprint arXiv:2204.05994 (2022)

[69] Wang, W., Wang, X., Feng, D., Liu, J., Han, Z., Zhang, X.: Exploring permission-induced risk in android applications for malicious application detection. IEEE Transactions on Information Forensics and Security **9**(11), 1869–1882 (2014)

[70] Grace, M., Sughasiny, M.: Behaviour analysis of inter-app communication using a lightweight monitoring app for malware detection. Expert Systems with Applications **210**, 118404 (2022)

[71] Liu, K., Zhang, G., Chen, X., Liu, Q., Peng, L., Yurui, L.: Android malware detection based on sensitive patterns. Telecommunication Systems **82**(4), 435–449 (2023)

[72] Bayazit, E.C., Sahingoz, O.K., Dogan, B.: A deep learning based android malware detection system with static analysis. In: 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp. 1–6 (2022). IEEE

[73] Lee, J., Jang, H., Ha, S., Yoon, Y.: Android malware detection using machine learning with feature selection based on the genetic

algorithm. Mathematics **9**(21), 2813 (2021)

[74] Zhu, H.-J., Wang, L.-M., Zhong, S., Li, Y., Sheng, V.S.: A hybrid deep network framework for android malware detection. IEEE Transactions on Knowledge and Data Engineering **34**(12), 5558–5570 (2021)

[75] Almahmoud, M., Alzu'bi, D., Yaseen, Q.: Redroiddet: android malware detection based on recurrent neural network. Procedia Computer Science **184**, 841–846 (2021)

[76] Feng, J., Shen, L., Chen, Z., Wang, Y., Li, H.: A two-layer deep learning method for android malware detection using network traffic. IEEE Access **8**, 125786–125796 (2020)

[77] Kandukuru, S., Sharma, R.: Android malicious application detection using permission vector and network traffic analysis. In: 2017 2nd International Conference for Convergence in Technology (I2CT), pp. 1126–1132 (2017). IEEE

[78] Arora, A., Peddoju, S.K.: Ntpdroid: a hybrid android malware detector using network traffic and system permissions. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 808–813 (2018). IEEE

[79] Ding, C., Luktarhan, N., Lu, B., Zhang, W.: A hybrid analysis-based approach to android malware family classification. Entropy **23**(8), 1009 (2021)

[80] Şahın, D.Ö., Akleylek, S., Kiliç, E.: Linregdroid: Detection of android malware using multiple linear regression models-based classifiers. IEEE Access **10**, 14246–14259 (2022)

[81] Idrees, F., Rajarajan, M., Chen, T.M., Rahulamathavan, Y., Naureen, A.: Andropin: Correlating android permissions and intents for malware detection. In: 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 394–399 (2017). IEEE

[82] Khariwal, K., Singh, J., Arora, A.: Ipdroid: Android malware detection using intents and permissions. In: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 197–202 (2020). IEEE

[83] Idrees, F., Rajarajan, M.: Investigating the android intents and permissions for malware detection. In: 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 354–358 (2014). IEEE

[84] Bai, H., Xie, N., Di, X., Ye, Q.: Famd: A fast multifeature android malware detection framework, design, and implementation. IEEE Access **8**, 194729–194740 (2020)

[85] Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., Conti, M.: Similarity-based android malware detection using hamming distance of static binary features. Future Generation Computer Systems **105**, 230–247 (2020)

[86] Alazab, M., Alazab, M., Shalaginov, A., Mesleh, A., Awajan, A.: Intelligent mobile malware detection using permission requests and api calls. Future Generation Computer Systems **107**, 509–521 (2020)

[87] Mathur, A., Podila, L.M., Kulkarni, K., Niyaz, Q., Javaid, A.Y.: Naticusdroid: A malware detection framework for android using native and custom permissions. Journal of Information Security and Applications **58**, 102696 (2021)

[88] Imtiaz, S.I., Rehman, S., Javed, A.R., Jalil, Z., Liu, X., Alnumay, W.S.: Deepamd: Detection and identification of android malware using high-efficient deep artificial neural network. Future Generation computer systems **115**, 844–856 (2021)

[89] Liu, Z., Wang, R., Japkowicz, N., Tang,

D., Zhang, W., Zhao, J.: Research on unsupervised feature learning for android malware detection based on restricted boltzmann machines. Future Generation Computer Systems **120**, 91–108 (2021)

[90] Chen, X., Shi, S., Xie, C., Yang, Z., Guo, Y., Fang, Y., Wen, W.: Suip: An android malware detection method based on data flow features. In: Journal of Physics: Conference Series, vol. 1812, p. 012010 (2021). IOP Publishing

[91] Guan, J., Jiang, X., Mao, B.: A method for class-imbalance learning in android malware detection. Electronics **10**(24), 3124 (2021)

[92] Mohamed, S.E., Ashaf, M., Ehab, A., Shereef, O., Metwaie, H., Amer, E.: Detecting malicious android applications based on api calls and permissions using machine learning algorithms. In: 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pp. 1–6 (2021). IEEE

[93] Pushpa Latha, D.: Bat optimization algorithm for wrapper-based feature selection and performance improvement of android malware detection (2021)

[94] Gyunka, B.A., Abikoye, O.C., Adekunle, A.S.: Anomaly android malware detection: A comparative analysis of six classifiers. In: International Conference on Information and Communication Technology and Applications, pp. 145–157 (2020). Springer

[95] Taha, A.A., Malebary, S.J.: Hybrid classification of android malware based on fuzzy clustering and the gradient boosting machine. Neural Computing and Applications **33**, 6721–6732 (2021)

[96] Peng, T., Hu, B., Liu, J., Huang, J., Zhang, Z., He, R., Hu, X.: A lightweight multi-source fast android malware detection model. Applied Sciences **12**(11), 5394 (2022)

[97] Ashwini, S., Pai, M., Sangeetha, J.: Android malware classification based on static features of an application. In: Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 2, pp. 567–581 (2021). Springer

[98] Jiang, X., Mao, B., Guan, J., Huang, X.: Android malware detection using fine-grained features. Scientific Programming **2020**, 1–13 (2020)

[99] Wang, J., Jing, Q., Gao, J., Qiu, X.: Sedroid: A robust android malware detector using selective ensemble learning. In: 2020 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–5 (2020). IEEE

[100] Rana, M.S., Sung, A.H.: Evaluation of advanced ensemble learning techniques for android malware detection. Vietnam Journal of Computer Science **7**(02), 145–159 (2020)

[101] Lu, T., Du, Y., Ouyang, L., Chen, Q., Wang, X.: Android malware detection based on a hybrid deep learning model. Security and Communication Networks **2020**, 1–11 (2020)

[102] Millar, S., McLaughlin, N., Rincon, J.M., Miller, P.: Multi-view deep learning for zero-day android malware detection. Journal of Information Security and Applications **58**, 102718 (2021)

[103] Barrera, D., Kayacik, H.G., Van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to android. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 73–84 (2010)

[104] Shabtai, A., Fledel, Y., Elovici, Y.: Automated static code analysis for classifying android applications using machine learning. In: 2010 International Conference on Computational Intelligence and Security, pp. 329–333 (2010). IEEE

[105] Felt, A.P., Chin, E., Hanna, S., Song, D.,

Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 627–638 (2011)

[106] Erickson, J.L., Gibler, C.D., Chen, H.: Androidleaks: Detecting privacy leaks in android applications. Technical report, Sandia National Lab.(SNL-CA), Livermore, CA (United States) (2011)

[107] Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pp. 13–22 (2012)

[108] Frank, M., Dong, B., Felt, A.P., Song, D.: Mining permission request patterns from android and facebook applications. In: 2012 IEEE 12th International Conference on Data Mining, pp. 870–875 (2012). IEEE

[109] Zhu, J., Guan, Z., Yang, Y., Yu, L., Sun, H., Chen, Z.: Permission-based abnormal application detection for android. In: Information and Communications Security: 14th International Conference, ICICS 2012, Hong Kong, China, October 29-31, 2012. Proceedings 14, pp. 228–239 (2012). Springer

[110] Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and api calls. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300–305 (2013). IEEE

[111] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Nieves, J., Bringas, P.G., Álvarez Marañón, G.: Mama: manifest analysis for malware detection in android. Cybernetics and Systems **44**(6-7), 469–488 (2013)

[112] Feldman, S., Stadther, D., Wang, B.: Manilyzer: automated android malware detection through manifest analysis. In: 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, pp. 767–772 (2014). IEEE

[113] Pehlivan, U., Baltaci, N., Acartürk, C., Baykal, N.: The analysis of feature selection methods and classification algorithms in permission based android malware detection. In: 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp. 1–8 (2014). IEEE

[114] Rahman, R., Islam, M.R., Ahmed, A., Hasan, M.K., Mahmud, H.: A study of permission-based malware detection using machine learning. In: 2022 15th International Conference on Security of Information and Networks (SIN), pp. 01–06 (2022). IEEE

[115] Rovelli, P., Vigfússon, Ý.: Pmds: permission-based malware detection system. In: Information Systems Security: 10th International Conference, ICISS 2014, Hyderabad, India, December 16-20, 2014, Proceedings 10, pp. 338–357 (2014). Springer

[116] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C.: Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss, vol. 14, pp. 23–26 (2014)

[117] Yerima, S.Y., Sezer, S., Muttik, I.: High accuracy android malware detection using ensemble learning. IET Information Security **9**(6), 313–320 (2015)

[118] Kang, H., Jang, J.-w., Mohaisen, A., Kim, H.K.: Detecting and classifying android malware using static analysis along with creator information. International Journal of Distributed Sensor Networks **11**(6), 479174 (2015)

[119] Zhao, K., Zhang, D., Su, X., Li, W.: Fest: A feature extraction and selection tool for android malware detection. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 714–720 (2015). IEEE

[120] Qiao, M., Sung, A.H., Liu, Q.: Merging permission and api features for android malware detection. In: 2016 5th IIAI International Congress on Advanced Applied

Informatics (IIAI-AAI), pp. 566–571 (2016). IEEE

[121] Chen, S., Xue, M., Tang, Z., Xu, L., Zhu, H.: Stormdroid: A streaminglized machine learning-based system for detecting android malware. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 377–388 (2016)

[122] Demertzis, K., Iliadis, L.: Bio-inspired hybrid intelligent method for detecting android malware. In: Knowledge, Information and Creativity Support Systems: Selected Papers from KICSS'2014-9th International Conference, Held in Limassol, Cyprus, on November 6-8, 2014, pp. 289–304 (2016). Springer

[123] Verma, S., Muttoo, S.K., Pal, S.: Mdroid: android based malware detection using mcm classifier. International Journal of Engineering Applied Sciences and Technology **1**(8), 206–215 (2016)

[124] Wang, K., Song, T., Liang, A.: Mmda: Metadata based malware detection on android. In: 2016 12th International Conference on Computational Intelligence and Security (CIS), pp. 598–602 (2016). IEEE

[125] Suarez-Tangil, G., Dash, S.K., Ahmadi, M., Kinder, J., Giacinto, G., Cavallaro, L.: Droidsieve: Fast and accurate classification of obfuscated android malware. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 309–320 (2017)

[126] Wang, X., Zhang, D., Su, X., Li, W., et al.: Mlifdect: android malware detection based on parallel machine learning and information fusion. Security and Communication Networks **2017** (2017)

[127] Li, D., Wang, Z., Li, L., Wang, Z., Wang, Y., Xue, Y.: Fgdetector: fine-grained android malware detection. In: 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), pp. 311–318 (2017). IEEE

[128] Bhattacharya, A., Goswami, R.T.: Dmdam: data mining based detection of android malware. In: Proceedings of the First International Conference on Intelligent Computing and Communication, pp. 187–194 (2017). Springer

[129] Xie, N., Zeng, F., Qin, X., Zhang, Y., Zhou, M., Lv, C.: Repassdroid: Automatic detection of android malware based on essential permissions and semantic features of sensitive apis. In: 2018 International Symposium on Theoretical Aspects of Software Engineering (TASE), pp. 52–59 (2018). IEEE

[130] Xie, N., Di, X., Wang, X., Zhao, J.: Andro_md: android malware detection based on convolutional neural networks. International Journal of Performability Engineering **14**(3), 547 (2018)

[131] Ren, B., Liu, C., Cheng, B., Guo, J., Chen, J.: Mobisentry: Towards easy and effective detection of android malware on smartphones. Mobile Information Systems **2018**, 1–14 (2018)

[132] Tao, G., Zheng, Z., Guo, Z., Lyu, M.R.: Malpat: Mining patterns of malicious and benign android apps via permission-related apis. IEEE Transactions on Reliability **67**(1), 355–369 (2017)

[133] Namrud, Z., Kpodjedo, S., Bali, A., Talhi, C.: Deep-layer clustering to identify permission usage patterns of android app categories. IEEE Access **10**, 24240–24254 (2022)

[134] Alswaina, F., Elleithy, K.: Android malware permission-based multi-class classification using extremely randomized trees. IEEE Access **6**, 76217–76227 (2018)

[135] Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., Wang, Y., Xiang, Y.: A3cm: automatic capability annotation for android malware. IEEE Access **7**, 147156–147168 (2019)

[136] Zhu, H., Li, Y., Li, R., Li, J., You, Z.,

Song, H.: Sedmdroid: An enhanced stacking ensemble framework for android malware detection. IEEE Transactions on Network Science and Engineering **8**(2), 984–994 (2020)

[137] Feng, Y., Chen, L., Zheng, A., Gao, C., Zheng, Z.: Ac-net: Assessing the consistency of description and permission in android apps. IEEE Access **7**, 57829–57842 (2019)

[138] Aonzo, S., Merlo, A., Migliardi, M., Oneto, L., Palmieri, F.: Low-resource footprint, data-driven malware detection on android. IEEE Transactions on Sustainable Computing **5**(2), 213–222 (2017)

[139] Urooj, B., Shah, M.A., Maple, C., Abbasi, M.K., Riasat, S.: Malware detection: a framework for reverse engineered android applications through machine learning algorithms. IEEE Access **10**, 89031–89050 (2022)

[140] Wang, R., Wang, Z., Tang, B., Zhao, L., Wang, L.: Smartpi: Understanding permission implications of android apps from user reviews. IEEE Transactions on Mobile Computing **19**(12), 2933–2945 (2019)

[141] Wang, W., Gao, Z., Zhao, M., Li, Y., Liu, J., Zhang, X.: Droidensemble: Detecting android malicious applications with ensemble of string and structural static features. IEEE Access **6**, 31798–31807 (2018)

[142] Zhang, J., Tian, C., Duan, Z., Zhao, L.: Rtpdroid: Detecting implicitly malicious behaviors under runtime permission model. IEEE Transactions on Reliability **70**(3), 1295–1308 (2021)

[143] Kesswani, N., Lyu, H., Zhang, Z.: Analyzing android app privacy with gp-pp model. Ieee Access **6**, 39541–39546 (2018)

[144] İbrahim, M., Issa, B., Jasser, M.B.: A method for automatic android malware detection based on static analysis and deep learning. IEEE Access **10**, 117334–117352 (2022)

[145] Arshad, S., Shah, M.A., Wahid, A., Mehmood, A., Song, H., Yu, H.: Samadroid: a novel 3-level hybrid malware detection model for android operating system. IEEE Access **6**, 4321–4339 (2018)

[146] Yuan, Z., Lu, Y., Xue, Y.: Droiddetector: android malware characterization and detection using deep learning. Tsinghua Science and Technology **21**(1), 114–123 (2016)

[147] Zhou, H., Yang, X., Pan, H., Guo, W.: An android malware detection approach based on simgru. IEEE Access **8**, 148404–148410 (2020)

[148] Cilleruelo, C., De-Marcos, L., Martinez-Herráiz, J.-J., *et al.*: Malware detection inside app stores based on lifespan measurements. IEEE Access **9**, 119967–119976 (2021)

[149] Firdaus, A., Anuar, N.B., Karim, A., Razak, M.F.A.: Discovering optimal features using static analysis and a genetic search based method for android malware detection. Frontiers of Information Technology & Electronic Engineering **19**(6), 712–736 (2018)

[150] Wang, C., Xu, Q., Lin, X., Liu, S.: Research on data mining of permissions mode for android malware detection. Cluster Computing **22**, 13337–13350 (2019)

[151] Singh, A.K., Jaidhar, C., Kumara, M.A.: Experimental analysis of android malware detection based on combinations of permissions and api-calls. Journal of Computer Virology and Hacking Techniques **15**, 209–218 (2019)

[152] Rafiq, H., Aslam, N., Aleem, M., Issac, B., Randhawa, R.H.: Andromalpack: enhancing the ml-based malware classification by detection and removal of repacked apps for android systems. Scientific Reports **12**(1), 19534 (2022)

[153] Mahdavifar, S., Alhadidi, D., Ghorbani, A.A.: Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. Journal of network

and systems management **30**, 1–34 (2022)

[154] Seraj, S., Khodambashi, S., Pavlidis, M., Polatidis, N.: Hamdroid: permission-based harmful android anti-malware detection using neural networks. Neural Computing and Applications **34**(18), 15165–15174 (2022)

[155] Mahindru, A., Sangal, A.: Somdroid: Android malware detection by artificial neural network trained using unsupervised learning. Evolutionary Intelligence **15**(1), 407–437 (2022)

[156] Anupama, M., Vinod, P., Visaggio, C.A., Arya, M., Philomina, J., Raphael, R., Pinhero, A., Ajith, K., Mathiyalagan, P.: Detection and robustness evaluation of android malware classifiers. Journal of Computer Virology and Hacking Techniques **18**(3), 147–170 (2022)

[157] Chen, X., Yu, H., Yu, D., Chen, J., Sun, X.: Predicting android malware combining permissions and api call sequences. Software Quality Journal **31**(3), 655–685 (2023)

[158] Mahindru, A., Sangal, A.: Fsdroid:-a feature selection technique to detect malware from android using machine learning techniques: Fsdroid. Multimedia Tools and Applications **80**, 13271–13323 (2021)

[159] Tchakounté, F., Ngassi, R.C.N., Kamla, V.C., Udagepola, K.P.: Limondroid: a system coupling three signature-based schemes for profiling android malware. Iran Journal of Computer Science **4**, 95–114 (2021)

[160] Nissim, N., Moskovitch, R., BarAd, O., Rokach, L., Elovici, Y.: Aldroid: efficient update of android anti-virus software using designated active learning methods. Knowledge and Information Systems **49**, 795–833 (2016)

[161] Peynirci, G., Eminağaoğlu, M., Karabulut, K.: Feature selection for malware detection on the android platform based on differences of idf values. Journal of Computer Science and Technology **35**, 946–962 (2020)

[162] Nauman, M., Tanveer, T.A., Khan, S., Syed, T.A.: Deep neural architectures for large scale android malware analysis. Cluster Computing **21**, 569–588 (2018)

[163] Bhattacharya, A., Goswami, R.T., Mukherjee, K.: A feature selection technique based on rough set and improvised pso algorithm (psors-fs) for permission based detection of android malwares. International journal of machine learning and cybernetics **10**, 1893–1907 (2019)

[164] Bao, L., Lo, D., Xia, X., Li, S.: Automated android application permission recommendation. Science China Information Sciences **60**, 1–17 (2017)

[165] Rubio-Medrano, C.E., Soundrapandian, P.K.D., Hill, M., Claramunt, L., Baek, J., Ahn, G.-J.: Dypoldroid: Protecting against permission-abuse attacks in android. Information Systems Frontiers **25**(2), 529–548 (2023)

[166] Shatnawi, A.S., Yassen, Q., Yateem, A.: An android malware detection approach based on static feature analysis using machine learning algorithms. Procedia Computer Science **201**, 653–658 (2022)

[167] Smmarwar, S.K., Gupta, G.P., Kumar, S., Kumar, P.: An optimized and efficient android malware detection framework for future sustainable computing. Sustainable Energy Technologies and Assessments **54**, 102852 (2022)

[168] Arif, J.M., Ab Razak, M.F., Mat, S.R.T., Awang, S., Ismail, N.S.N., Firdaus, A.: Android mobile malware detection using fuzzy ahp. Journal of Information Security and Applications **61**, 102929 (2021)

[169] Guerra-Manzanares, A., Bahsi, H., Nõmm, S.: Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization. Computers & Security **110**, 102399 (2021)

[170] Bhat, P., Dutta, K.: A multi-tiered feature selection model for android malware

detection based on feature discrimination and information gain. Journal of King Saud University-Computer and Information Sciences **34**(10), 9464–9477 (2022)

[171] Elayan, O.N., Mustafa, A.M.: Android malware detection using deep learning. Procedia Computer Science **184**, 847–852 (2021)

[172] Syrris, V., Geneiatakis, D.: On machine learning effectiveness for malware detection in android os using static analysis data. Journal of information security and applications **59**, 102794 (2021)

[173] Idrees, F., Rajarajan, M., Conti, M., Chen, T.M., Rahulamathavan, Y.: Pindroid: A novel android malware detection system using ensemble learning methods. Computers & Security **68**, 36–46 (2017)

[174] Rehman, Z.-U., Khan, S.N., Muhammad, K., Lee, J.W., Lv, Z., Baik, S.W., Shah, P.A., Awan, K., Mehmood, I.: Machine learning-assisted signature and heuristic-based detection of malwares in android devices. Computers & Electrical Engineering **69**, 828–841 (2018)

[175] Martín, A., Lara-Cabrera, R., Camacho, D.: Android malware detection through hybrid features fusion and ensemble classifiers: The andropytool framework and the omnidroid dataset. Information Fusion **52**, 128–142 (2019)

[176] Navarro, L.C., Navarro, A.K., Grégio, A., Rocha, A., Dahab, R.: Leveraging ontologies and machine-learning techniques for malware analysis into android permissions ecosystems. Computers & Security **78**, 429–453 (2018)

[177] Milosevic, N., Dehghantanha, A., Choo, K.-K.R.: Machine learning aided android malware classification. Computers & Electrical Engineering **61**, 266–274 (2017)

[178] Alzaylaee, M.K., Yerima, S.Y., Sezer, S.: Dl-droid: Deep learning based android malware detection using real devices. Computers & Security **89**, 101663 (2020)

[179] Cai, L., Li, Y., Xiong, Z.: Jowmdroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. Computers & Security **100**, 102086 (2021)

[180] Badhani, S., Muttoo, S.K.: Cendroid—a cluster-ensemble classifier for detecting malicious android applications. Computers & Security **85**, 25–40 (2019)

[181] Alqatawna, J., Ala'M, A.-Z., Hassonah, M.A., Faris, H., *et al.*: Android botnet detection using machine learning models based on a comprehensive static analysis approach. Journal of Information Security and Applications **58**, 102735 (2021)

[182] Sheen, S., Anitha, R., Natarajan, V.: Android based malware detection using a multifeature collaborative decision fusion approach. Neurocomputing **151**, 905–912 (2015)

[183] OS, J.N., *et al.*: Detection of malicious android applications using ontology-based intelligent model in mobile cloud environment. Journal of Information Security and Applications **58**, 102751 (2021)

[184] Song, J., Han, C., Wang, K., Zhao, J., Ranjan, R., Wang, L.: An integrated static detection and analysis framework for android. Pervasive and Mobile Computing **32**, 15–25 (2016)

[185] Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., Wang, X.S., Zang, B.: Vetting undesirable behaviors in android apps with permission use analysis. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 611–622 (2013)

[186] Yang, S., Zeng, Z., Song, W.: Permdroid: automatically testing permission-related behaviour of android applications. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 593–604 (2022)

[187] Thiyagarajan, J., Akash, A., Murugan, B.:

Improved real-time permission based malware detection and clustering approach using model independent pruning. IET Information Security **14**(5), 531–541 (2020)

[188] Qaisar, Z.H., Li, R.: Multimodal information fusion for android malware detection using lazy learning. Multimedia Tools and Applications, 1–15 (2022)

[189] Appice, A., Andresini, G., Malerba, D.: Clustering-aided multi-view classification: a case study on android malware detection. Journal of intelligent information systems **55**, 1–26 (2020)

[190] Zhu, H.-J., Jiang, T.-H., Ma, B., You, Z.-H., Shi, W.-L., Cheng, L.: Hemd: a highly efficient random forest-based malware detection framework for android. Neural Computing and Applications **30**, 3353–3361 (2018)

[191] Altaher, A.: An improved android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (ehnfc) and permission-based features. Neural Computing and Applications **28**, 4147–4157 (2017)

[192] Su, X., Shi, W., Qu, X., Zheng, Y., Liu, X.: Droiddeep: using deep belief network to characterize and detect android malware. Soft Computing **24**(8), 6017–6030 (2020)

[193] Mahindru, A., Sangal, A.: Mldroid—framework for android malware detection using machine learning techniques. Neural Computing and Applications **33**(10), 5183–5240 (2021)

[194] Dehkordy, D.T., Rasoolzadegan, A.: A new machine learning-based method for android malware detection on imbalanced dataset. Multimedia Tools and Applications **80**, 24533–24554 (2021)

[195] Nguyen, D.V., Nguyen, G.L., Nguyen, T.T., Ngo, A.H., Pham, G.T.: Minad: Multi-inputs neural network based on application structure for android malware detection. Peer-to-Peer Networking and Applications, 1–15 (2022)

[196] Taheri, R., Javidan, R., Pooranian, Z.: Adversarial android malware detection for mobile multimedia applications in iot environments. Multimedia Tools and Applications **80**, 16713–16729 (2021)

[197] Mahesh, P.S., Hemalatha, S.: An efficient android malware detection using adaptive red fox optimization based cnn. Wireless Personal Communications **126**(1), 679–700 (2022)

[198] Firdaus, A., Anuar, N.B., Razak, M.F.A., Sangaiah, A.K.: Bio-inspired computational paradigm for feature investigation and malware detection: interactive analytics. Multimedia Tools and Applications **77**, 17519–17555 (2018)

[199] Shrivastava, G., Kumar, P.: Intent and permission modeling for privacy leakage detection in android. Energy Systems **13**(3), 567–580 (2022)

[200] Varsha, M., Vinod, P., Dhanya, K.: Identification of malicious android app using manifest and opcode features. Journal of Computer Virology and Hacking Techniques **13**, 125–138 (2017)

[201] Deypir, M.: Entropy-based security risk measurement for android mobile applications. Soft Computing **23**(16), 7303–7319 (2019)

[202] Mahindru, A., Sangal, A.: Semidroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches. International Journal of Machine Learning and Cybernetics **12**, 1369–1411 (2021)

[203] Keyvanpour, M.R., Barani Shirzad, M., Heydarian, F.: Android malware detection applying feature selection techniques and machine learning. Multimedia Tools and Applications **82**(6), 9517–9531 (2023)

[204] Razak, M.F.A., Anuar, N.B., Othman, F., Firdaus, A., Afifi, F., Salleh, R.: Bio-inspired for features optimization and malware detection. Arabian Journal for Science

and Engineering **43**, 6963–6979 (2018)

[205] Xie, N., Wang, X., Wang, W., Liu, J.: Fingerprinting android malware families. Frontiers of Computer Science **13**, 637–646 (2019)

[206] Mahindru, A., Sangal, A.: Hybridroid: an empirical analysis on effective malware detection model developed using ensemble methods. The Journal of Supercomputing **77**, 8209–8251 (2021)

[207] Ali, T., Khan, Y., Ali, T., Faizullah, S., Alghamdi, T., Anwar, S.: An automated permission selection framework for android platform. Journal of Grid Computing **18**, 547–561 (2020)

[208] Sun, J., Yan, K., Liu, X., Yang, C., Fu, Y.: Malware detection on android smartphones using keywords vector and svm. In: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), pp. 833–838 (2017). IEEE

[209] AlJarrah, M.N., Yaseen, Q.M., Mustafa, A.M.: A context-aware android malware detection approach using machine learning. Information **13**(12), 563 (2022)

[210] Gharib, A., Ghorbani, A.: Dna-droid: A real-time android ransomware detection framework. In: Network and System Security: 11th International Conference, NSS 2017, Helsinki, Finland, August 21–23, 2017, Proceedings 11, pp. 184–198 (2017). Springer

[211] Sun, L., Wei, X., Zhang, J., He, L., Philip, S.Y., Srisa-an, W.: Contaminant removal for android malware detection systems. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1053–1062 (2017). IEEE

[212] Chen, J., Wang, C., Zhao, Z., Chen, K., Du, R., Ahn, G.-J.: Uncovering the face of android ransomware: Characterization and real-time detection. IEEE Transactions on Information Forensics and Security **13**(5), 1286–1300 (2017)

[213] Senliol, B., Gulgezen, G., Yu, L., Cataltepe, Z.: Fast correlation based filter (fcbf) with a different search strategy. In: 2008 23rd International Symposium on Computer and Information Sciences, pp. 1–4 (2008). IEEE

[214] Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with java implementations. Acm Sigmod Record **31**(1), 76–77 (2002)

[215] Fan, M., Liu, J., Luo, X., Chen, K., Tian, Z., Zheng, Q., Liu, T.: Android malware familial classification and representative sample selection via frequent subgraph analysis. IEEE Transactions on Information Forensics and Security **13**(8), 1890–1905 (2018)

[216] Wei, F., Li, Y., Roy, S., Ou, X., Zhou, W.: Deep ground truth analysis of current android malware. In: Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14, pp. 252–276 (2017). Springer

[217] Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: 2012 IEEE Symposium on Security and Privacy, pp. 95–109 (2012). IEEE

[218] Abdul Kadir, A.F., Stakhanova, N., Ghorbani, A.A.: Android botnets: What urls are telling us. In: Network and System Security: 9th International Conference, NSS 2015, New York, NY, USA, November 3-5, 2015, Proceedings 9, pp. 78–91 (2015). Springer

[219] Lashkari, A.H., Kadir, A.F.A., Taheri, L., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST), pp. 1–7 (2018). IEEE

[220] Andronio, N., Zanero, S., Maggi, F.: Heldroid: Dissecting and detecting mobile ransomware. In: Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015. Proceedings 18, pp. 382–404 (2015). Springer

[221] Jang, J.-w., Kang, H., Woo, J., Mohaisen, A., Kim, H.K.: Andro-autopsy: Anti-malware system based on similarity matching of malware and malware creator-centric information. Digital Investigation **14**, 17–35 (2015)

[222] Avdiienko, V., Kuznetsov, K., Gorla, A., Zeller, A., Arzt, S., Rasthofer, S., Bodden, E.: Mining apps for abnormal usage of sensitive data. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 1, pp. 426–436 (2015). IEEE

[223] Damshenas, M., Dehghantanha, A., Choo, K.-K.R., Mahmud, R.: M0droid: An android behavioral-based malware detection model. Journal of Information Privacy and Security **11**(3), 141–157 (2015)

[224] Arslan, R.S.: Androanalyzer: android malicious software detection based on deep learning. PeerJ Computer Science **7**, 533 (2021)

[225] Mahdavifar, S., Kadir, A.F.A., Fatemi, R., Alhadidi, D., Ghorbani, A.A.: Dynamic android malware category classification using semi-supervised deep learning. In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 515–522 (2020). IEEE

[226] Maiorca, D., Ariu, D., Corona, I., Aresu, M., Giacinto, G.: Stealth attacks: An extended insight into the obfuscation effects on android malware. Computers & Security **51**, 16–31 (2015)