IJIST
P ISSN : 2618-1630
E ISSN : 2909-6130

Pragmatic Evidence on Android Malware Analysis Techniques: A Systematic Literature Review

Review Article

Mian Muhammad Bilal [1], Ghulam Rasool [1], Sajid Ibrahim Hashmi[1], Zaigham Mushtaq [2]
[1]Department of Computer Science (COMSATS University Islamabad, Lahore Campus).
[2]Department of Computer Science (The Islamia University of Bahawalpur).
* **Correspondence**: Main Muhammad Bilal (e-mail: bilalcomsis3@gmail.com).

Citation | Bilal. M. M, Rasool. G, Hashmi. I. S, Mushtaq. Z., "Pragmatic Evidence on Android Malware Analysis Techniques: A Systematic Literature Review", IJIST, vol. 5, no. 1, pp. 1-19, Jan 2023.
DOI: https://doi.org/10.33411/IJIST/2023050101
Received | Nov 02,2022; **Revised** | Dec 21,2022 **Accepted** | Dec 26,2022; **Published** | Jan 01,2023.

A large number of studies including research articles and surveys on android malware detection and analysis techniques have been presented during the last one and a half decades. The authors proposed different systems and frameworks to identify malware from software applications. However, there is no recent and comprehensive systematic literature review on the detection and analysis of android malware methods, systems, and frameworks. We present a systematic review of literature on android malware detection and analysis techniques and tools by following standard guidelines for Systematic Literature Review methodology from 2010 to 2021. We selected 75 most relevant studies out of 3343 published studies. We found that the prominent malicious datasets are Genome (39%) and Drebin (36%) used by different researchers for the detection of malware. The static, dynamic, and hybrid source code analysis methods are applied by android malware detection techniques. We also identified the limitations and future research directions of existing techniques as research gaps for the community. Based on the pragmatic evidence of this research, we have proposed a hybrid analysis-based multiple feature analysis framework. This framework will not only address the limitations of static and dynamic-based approaches, but it also analyzes evolving android malware datasets using deep neural network and machine learning techniques and improve the accuracy of evolving malware samples.

Keywords: Android Malware; Systematic Literature Review; Static Analysis; Dynamic Analysis and Reverse Engineering.

Conflict of interest. The authors declare that there exists no conflict of interest for publishing this manuscript in IJIST. **Project details.** Nil

Author's Contribution. All authors have contributed equally

Introduction

Android is a mobile-based open-source operating system from Google. It is developed for smartphones and tablets. The increasing and improved state of Android malware has dramatically reduced the effectiveness of security measures, leaving platforms like Android vulnerable to anonymous and novel types of malware [1]. The recognition of Android is continuously increasing due to its evolving powerful application features. Android applications are directly installed on mobile phones as opposed to download and run-on personal computers [2][3]. Android the most popular operating system has a market share of 72.2% of the smartphone market in the second quarter of 2021. In 2021, mobile phones powered by Android were placed first, reckoned for 83.8% of mobile phone sales in an organized manner. According to statistics, 17% of Android applications contain malware [4]. On the other hand, 97% of smartphone malware is reported to be on Android [5]. The first android malware emerged in 2010 and since then the platform has been under attack by diverse and unique sets of malware. Unlike a conventional program, malware continuously grows; eventually, it exploits security flaws and vulnerabilities to infect, propagate and continue [6]. To detect harmful behavior, static analysis methods investigate malware source code. Moreover, features are extracted from .apk files for the static analysis (e.g., Android App package installation). However, finding these features is very difficult when such files are unintelligible [7]. Bae et al. [8] investigated the possibility of employing a dynamic analysis method at a low cost to mobile phones. Subsequently, authors established that linking different features did not add an overhead, it rather produced appropriate results after detection. Android Malware has been evolving continuously and there is always room for enhancement and betterment of its detection and prevention opportunities. To get a clear picture of android malware detection and analysis techniques considering static, hybrid, and dynamic, we systematically perform a literature review after selecting rigorously connected studies. The major contributions of this SLR are given below:

- We present this SLR considering the major aspects of Android malware detection techniques and tools.
- We propose a hybrid analysis framework based on the comprehensive systematic literature review.
- We identify research gaps that may be used for researchers in future work on this topic.

The remainder of this SLR is organized as follows. Materials and Methods questions with the overall review protocol for this study are presented in Section 2. Results and Discussion on research questions are reported in Section 3. Finally, this research work concludes and highlights future research directions in Section 4.

Investigation Site.

The research community proposed several methods using dynamic, static, or hybrid analysis to detect and analyze evolving android malware [9] [7][8] [10] [11] [12] [13] [14]. To date, researchers presented very few literature reviews and surveys on android malware detection techniques. Y. Pan et al [15] presented a systematic literature review considering static analysis on Android malware detection. This SLR covers the period of January 2014 to March 2020 and extracted studies based on four categories such as opcode, characteristic, symbolic execution, and program techniques considering static analysis. Another survey performed by K. Liu et al [16] reviewed machine learning-based Android malware detection methods. Key aspects such as feature selection, machine learning models, data processing, and algorithms were summarized and analyzed in this review. Another review presented by M. Odusami et al discussed android malware detection methods for unknown types of malware. There exist no work that presents a systematic literature review on android

malware detection techniques and tools from 2010 to 2021. Moreover, to date, no state-of-the-art literature review has been presented considering source code analysis in terms of static, dynamic and hybrid android malware detection methods. Existing survey and literature reviews do not consider tools, datasets, limitations, and future aspects of android malware detection methods. With the advent of time, there is an evolution in android malware development and different techniques are developed to analyse and detect these evolving malware samples. There is a need to perform a systematic literature review to understand the different techniques and tools used to detect and analyse android malware. There exists a literature review [17] on android malware detection techniques which is based on static analysis only. A few review studies exist but they do not provide a significant amount of information. A recent and comprehensive state of artwork on the topic is presented to provide directions for the people working in this domain.

Material and Methods.

This SLR (Systematic Literature Review) represents the advancements made in Android malware identification and prevention since its origin in the smart mobile device market. We follow the Planning, Conducting, and Reporting steps suggested by Kitchenham et al. [17]. Figure 1 represents the process followed in our SLR.

A. Review Protocol:

Three steps that were followed for this review are mentioned below:

- Plan Review: This step distinguished the overall objectives and helped to construct the study protocols.
- Conduct Review: This step included research questions, search strategy, selection criteria of the study, quality assessment criteria, data extraction, and data synthesis.
- Report Review: This is the final step of the SLR in which a report on the review results was created.

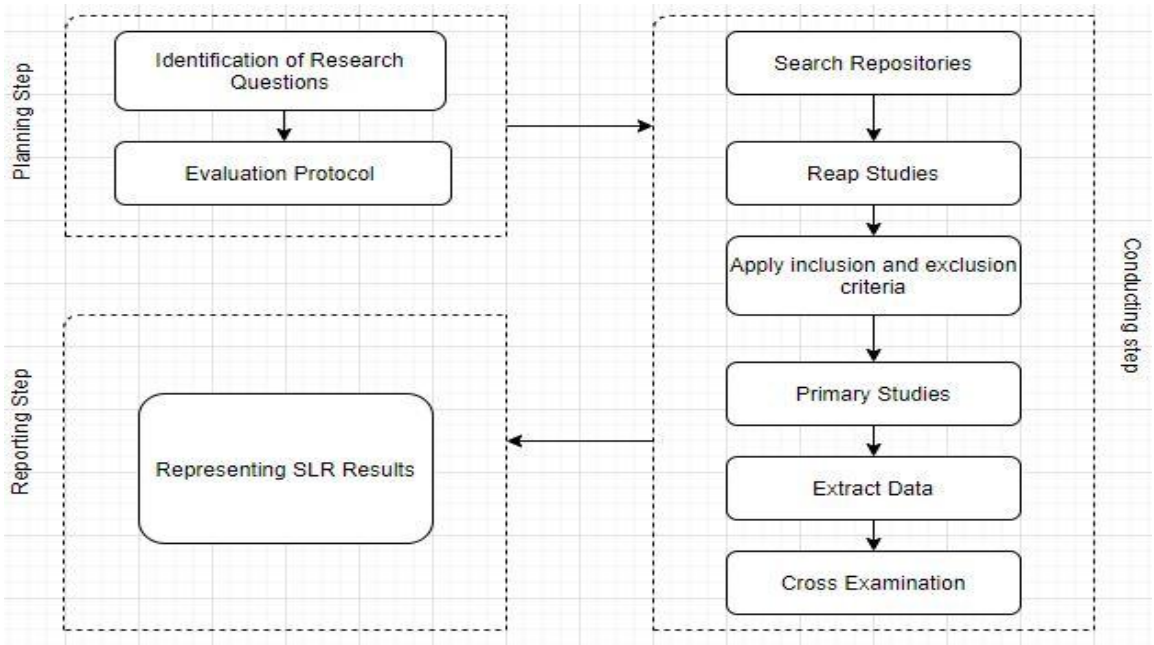

Figure 1 Overview of Systematic Literature Review Process

B. Research Questions:

Table 1 presents five Research Questions (RQ) relevant to Android malware detection techniques and tools based on static, hybrid, and dynamic analysis.

Table 1: Research Questions

RQ No.	Research Questions	Motivation
1	RQ1: What are different techniques and tools applied for the analysis and detection of Android Malware?	To identify Android malware detection techniques and tools used by these techniques
2	RQ2: What are the limitations of prevalent research techniques and tools being used for the analysis and detection of Android Malware?	To determine the limitations highlighted in the existing state of the art on Android malware analysis and detection
3	RQ3: Which datasets are used for the evaluation of Android Malware analysis and detection techniques?	To identify the datasets used for evaluation and experimental purposes by the Android malware analysis and detection methods
4	RQ4: Which source code Analysis methods are used by Android Malware techniques and tools?	To identify source code analysis methods in terms of static, dynamic, and hybrid
5	RQ5: What are the research gaps and future research directions for Android Malware analysis techniques and tools?	To identify research gaps and future work highlighted in Android malware analysis and detection techniques

C. Search Strategy:

To formulate an effective search string aligned with the Metasearch sentence, the finalized keywords were concatenated with Boolean operators (`AND' and `OR') and wildcard characters ('*'). Finally, the search string with the combination of Boolean operators and the wildcard is mentioned. ('android malware' **OR** 'history of android malware' **OR** 'android malware evolution' **OR** 'evolution of android malware' **OR** 'android malware framework' **OR** 'android malware system' **AND** ('analyze*' **OR** 'detect*' **OR** 'software*' **OR** 'revers*')

We investigated five electronic databases through Google Scholar and Google, which are listed as follows.

- IEEE Xplore Digital Library
- Science Direct
- ACM Digital Library
- Wiley Online Library
- Springer Link

All studies related to search string are considered and the search range is from August 2010 to January 2021. Table 2 presents the attributes specified for the extraction of information from primary studies.

Table 2: Attributes of Research Studies

Attributes	Sub Attributes	Description
General Elements	Paper No., Authors, Title, Paper Type, Year, Country, Features, Source Code Analysis method, Sub SCA, Proposed tools/techniques, Metrics, Emails, Datasets used, dataset types, Algorithm domain, Case study, Accuracy	Report-related details corresponding to the research publications
Research Publications	Paper No., Paper Title, Publication Type, Publications, Year, Domain of Research, Origin	Describe full information about relevant research publications

Android Malware Detection Tools	Paper No., Paper Title, Tools, Domain/Techniques/Mechanism, Language, Algorithm Domain	Report related details on tools
Study assessment criteria	Paper No., Paper Title, Datasets, Dataset Type, Source Code Analysis Type, Limitations, Future Work	The selected research studies are evaluated based on different attributes
Publication details with Origin	Paper No. Publisher Name, Total Papers, Journals, Conferences, Workshops, References	Report details on research publications
Selected Publications Status	Paper Name, Search Criteria for Paper selection, Study selection at step-1, Study selection at Step 2, Study Selection at Step 3	Indicate the number of research papers after applying each step
Final Publications Selected	No. of Total Papers, Journals, Conferences, Workshops	Represent the total number of research

D. Selection Criteria of the Study:

To ensure the most relevant research, the publication period from January 2010 to January 2021 was considered. We applied Kitchenham [17] guidelines to apply inclusion and exclusion criteria for selecting the most relevant studies. The inclusion criteria are:

- The allowed period for the selection is from January 2010 to January 2021.
- The research must be published in the English language.
- Only full-length papers must be considered.
- The selection process follows the search steps
- The assessment mechanism must be following the specified criteria.
- To filter out irrelevant studies, a series of exclusion criteria were formed as follows.
- Slides", "tutorials", "editorials", "posters" and other non-peer reviews were discarded.
- Books and various blogs were excluded.
- Publications other than the English language were excluded.
- Studies with fewer than 8 pages were excluded.
- Due to the presence of the search keyword 'Android', studies other than malware were excluded.

The quality assessment criteria are defined in Table 3. After this level, 68 papers were left on the list.

Table 3: Error! No text of specified style in document.**:** Quality Assessment Criteria and Primary Study Results

QA	Quality Criteria Question	Score	Y (1)	P (0.5)	N (0)
1	Did the study discuss any technique and tool for Android malware detection or analysis?	"Yes = 1, Partial = 0.5, No= 0"	81	6	5
2	Did the study report any key research limitations of prevalent techniques and tools used for Android malware detection or analysis?	"Yes = 1, Partial = 0.5, No= 0"	38	11	43
3	Are the datasets mentioned in the primary studies?	"Yes = 1, Partial = 0.5, No= 0"	79	5	8
4	Did the study discuss source code analysis methods used by Android malware techniques and tools?	"Yes = 1, Partial = 0.5, No= 0"	78	4	10
5	Did the study discuss any future work of prevalent research techniques used for Android malware detection or analysis?	"Yes = 1, Partial = 0.5, No= 0"	42	19	34

E. Study Selection Process:

To obtain independent assessments, a six-level selection process was conducted, as shown in Figure 2.

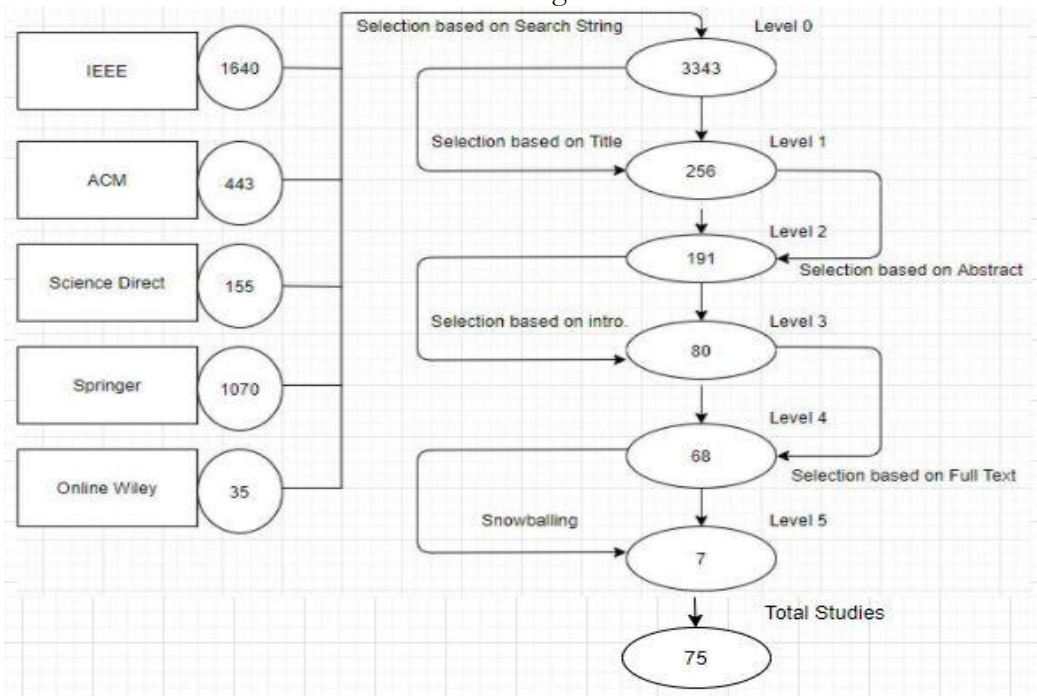

Figure 2 Study Selection Process

After forming all the steps, we applied snowball tracking by searching through the reference list of each finalized study and ensured that no important study was missed. 7 more studies [18][19][20][21][22][23][24] were identified to make it seventy-five in total.

Result and Discussion.

The objective of this section is to present the results obtained from primary studies. We first discuss generic facts about primary studies. Secondly, according to facts, we present the results of this SLR considering the research questions. Finally, based on research evidence results, a conclusion and future suggestions are presented.

A. Quality Assessment Criteria Evaluation:

The overall quality assessment score of the finalized primary studies is mentioned in Figure 3 below.

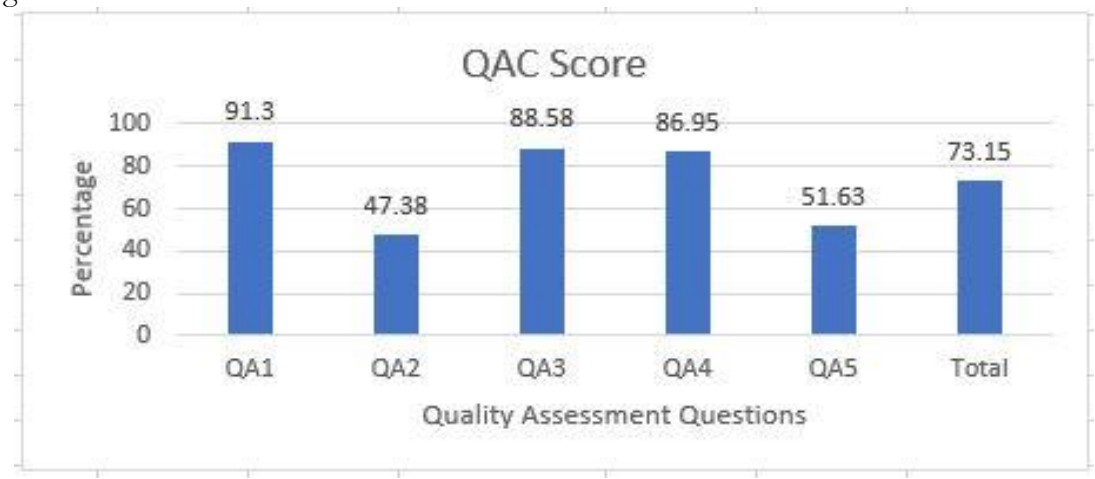

Figure 3 Selected Studies Quality Assessment Score

B. Description of Search Results:

We finalized 80 papers in level III out of 3343 searched papers published between the years 2010 and 2021. The status of the papers along with the ratio of selection at different levels is mentioned in Table 4. Figure 4 indicates the percentage of the selected primary studies from each selected digital library mentioned in Section 3 (Methodology).

Table 4*:* Error! No text of specified style in document.**:** Publication Status

Dataset	Level0	Level1	Level2	Level3	Level4	SNB	Total	%
ACM	443	30	19	11	10	2	12	16
Wiley	35	9	9	9	9	0	9	12
Sci Direct	155	40	32	26	21	0	21	28
Springer	1070	65	41	7	6	2	8	11
IEEE	1640	112	90	27	22	3	25	35
Total	3343	256	191	80	68	7	75	

SNB: Snow Balling

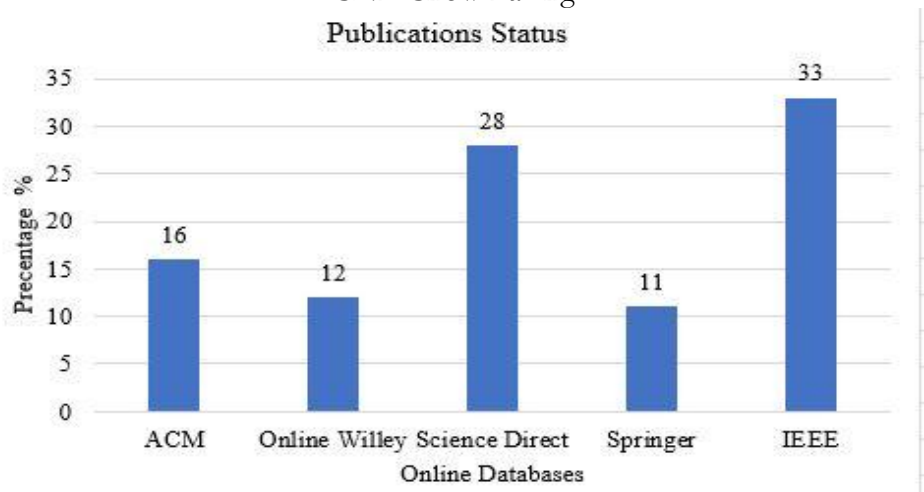

Figure 4: Error! No text of specified style in document. Frequency of the selected primary publication and proceedings

Table 5: Summary of the selected primary studies

Year	Journal	Conference	Count
2020	5	0	5
2019	11	1	12
2018	13	2	15
2017	11	3	14
2016	9	4	13
2015	6	0	6
2014	3	2	5
2013	1	1	2
2012	0	1	1
2011	1	1	2
2010	0	0	0
Total	**60**	**15**	**75**
%	80	20	100

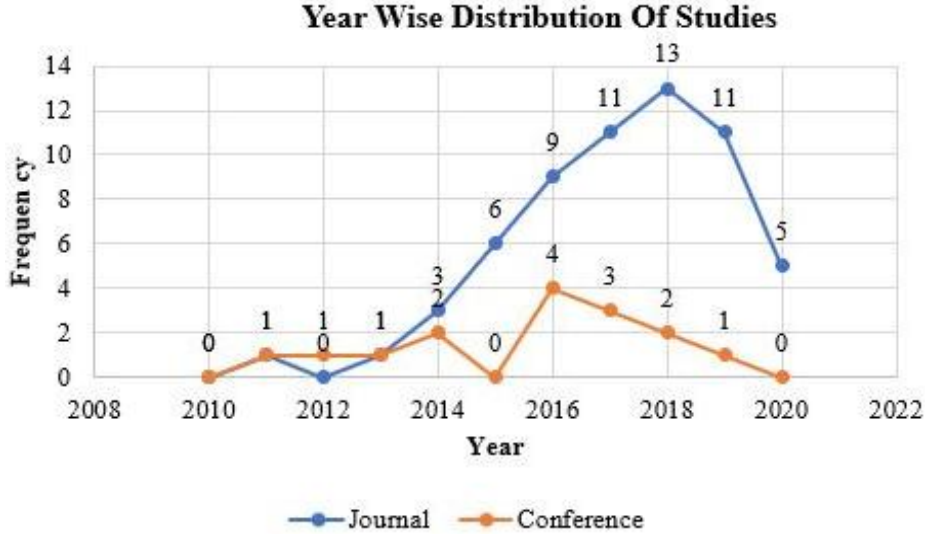

Figure 5 Year Wise Distribution of the primary studies

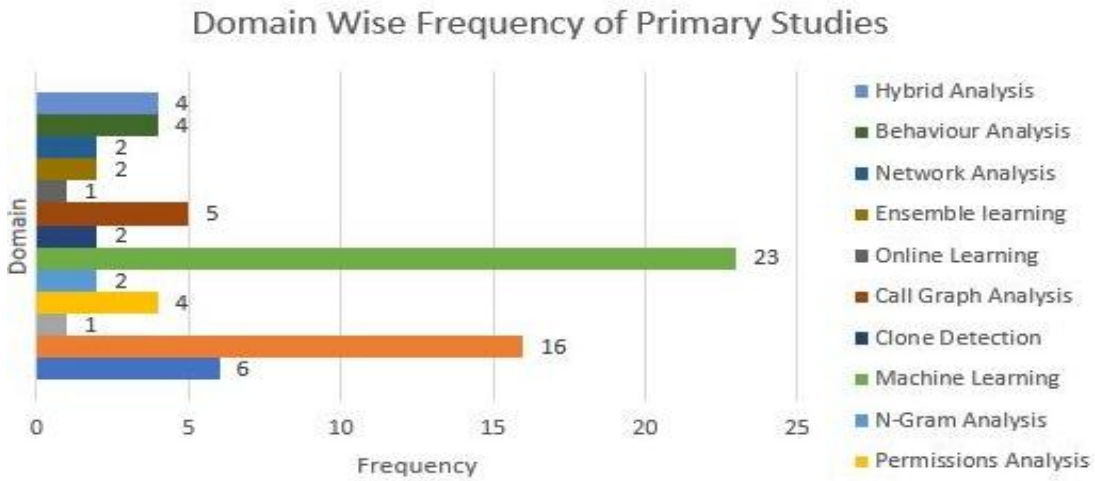

Figure 6 Domain-wise frequency of the papers

Table 5 represents the number of publications concerning the year of publication. Figure 5 indicates the year-wise distribution of the selected primary studies. Figure 6 indicates the domain-wise distribution of the selected primary studies.

C. Results and Analysis of Research Questions:

RQ1: What are Different Techniques and Tools Applied for the Analysis and Detection of Android Malware?

Android Malware Analysis and Detection Techniques

We discuss different Android malware analysis and detection techniques that are presented. The analysis mechanism contains Static analysis, Dynamic analysis, and Hybrid analysis as the major along with network analysis.

a. Permission-Based Static Analysis:

This category focuses on Android malware detection techniques considering permissions as a feature-based static analysis. Therefore, to analyze and detect Android malware using permissions, different techniques based on static analysis have been presented to date [3][25][12][26][27][28][29].

b. Graph-Based Static Analysis:

In this category, those static Android malware detection techniques have been focused on, which consider call graph and subgraph analysis. There are many studies [5][7][29][30][31][32] that adopted graph-based techniques to analyze and detect Android malware.

c. Android Feature-Based Static Analysis:

This category mainly focuses on features collected from android binary files based on byte code and binary configuration files. Therefore, techniques [33][34][35][27][36][37][19][38][39][40] based on the features have been proposed by different authors.

d. Opcode-Based Static Analysis:

Android malware analysis and detection techniques are based on the opcode of an application, which is obtained from a binary file in APK. In literature, authors have proposed several techniques [1][2][34][41][42][43][20] considering opcode sequences based on static analysis.

b. Dynamic Analysis Based Android Malware Analysis and Detection Techniques:

In dynamic analysis, the run time behavior of the malware is monitored to identify the malicious activity, while in static analysis source code of the malware app is analyzed to detect malicious behavior. Different dynamic analysis [9][6][10][44][18][13][45][46][47][48][49][50][51][52] techniques have been presented in the literature which targets the run time behavior of the app.

c. Hybrid Analysis Based Android Malware Analysis and Detection Techniques:

Several techniques [53][14][54][55][56][57][58][59][60][61][62][63][64] have been presented to date that first use static analysis to get static information of the app and then apply machine learning techniques to analyze the behavior of the app and detect malicious applications. Table 6 presents summarized information about a few android malware analysis and detection techniques.

Table 6: Android malware analysis and detection techniques

Sr.#	Techniques	Mechanism	Experimental Evaluation
1	Software Engineering	Static Analysis	Machine Learning, Signature Matching Algorithms
2	Reverse Engineering	Static Analysis, Hybrid Analysis	Chi-Square, Fisher Score, Information Gain, SVM, Fast Greedy Algorithms
3	Machine Learning	Static Analysis, Dynamic Analysis, Hybrid Analysis	Random Forest, Decision Trees, Naïve Bay's, KNN, Conforamal Prediction
4	Deep Learning	Static Analysis, Dynamic Analysis, Hybrid Analysis	Deep Learning Algorithms
5	Call Graph Analysis	Static Analysis	The batch learning algorithm, SVM, the signature matching algorithm

Android Malware Analysis and Detection Tools

Table 7 depicts a few static analysis tools with a few related examples.

a. Static Analysis Tools for Android Malware Analysis and Detection:

Table 7: Android Malware Static Analysis Tools and Support

Sr. #	Static Analysis Tool	Examples	Type	Domain	
1	APKTool	[2][5][6][34][35][65][19][38][39]	Industrial App	Reverse Engineering	Java
2	Matplotlib	[1] [34][66]	Library	Deep Learning, Machine Learning	Python
3	Numpy	[1] [34],[66]	Library	Deep Learning, Machine Learning	Python
4	Pandas	[1] [34][67]	Library	Deep Learning, Machine	Python, C

				Learning	
5	Sklearn	[1][34][35][31][19]	Library	Machine Learning	Python,C++,C
6	Androguard	[29][30][32]	Industrial App	API Call Graph Analysis	Python

b. Dynamic Analysis Tools for Android Malware Analysis and Detection:

Different dynamic analysis tools few examples are presented in Table 8.

Table 8: Android Malware Dynamic Analysis Tools and Support

Sr.#	Dynamic Analysis Tool	Ref	Type	Domain	Language
1	Monkey	[9],[8],[11],[13],[53],[14],[51],[21],[2],[22],[54],[55]	Emulator	Machine Learning	Java, C, C++, Python
2	Droid Box	[68],[56],[64],[23],[13]	Emulator	Machine Learning	Python
3	Strace	[13],[22],[54],[58]	CLI	Behaviour Analysis	C
4	Libsvm	[8]	Library	Support Vector Machine	Java, C++
5	Flowdroid	[33],[18],[53]	Industrial App	Software Engineering, called Graph	Java

RQ2: What are The Limitations of Prevalent Research Techniques and Tools Being Used for the Analysis and Detection of Android Malware?

In this subsection, the limitations of various methodologies are highlighted. 43 issues are perceived independently from chosen research contributions which are marked with Limitation Number (#). These few limitations are separated into 10 software engineering domains. The detail is presented in Table 9.

Table 9: Limitations of Android Malware Analysis Techniques and Tools

L NO	Limitation	Ref no
L1	the method is vulnerable to false positives when detecting information leak attack	[69]
L2	the system has external dependencies that could be replaced with a standalone app	[25]
L3	control flow and string encryption can affect user trigger dependencies method	[36]
L4	non-malicious apps are detected as malicious, and some samples remain undetected	[27]
L5	only known malware families can be detected with this approach	[70]
L6	dependence relation could not be detected with this method for repacked malware	[65]
L7	generalizability of the datasets used by Reveal Droid affects the results	[41]
L8	call graph extraction for all apps in the dataset, failed by the soot tool	[7]
L9	malicious behavior of the native code could not be detected with the proposed approach	[48]
L10	runtime malicious behavior detection may fail	[38]

RQ3: Which Datasets are used for the Evaluation of Android Malware Analysis and Detection Techniques?

In Figure 7, the generic dataset types and frequency of the studies are presented. In Figure 8, the percentage of the datasets used in different selected primary studies is presented.

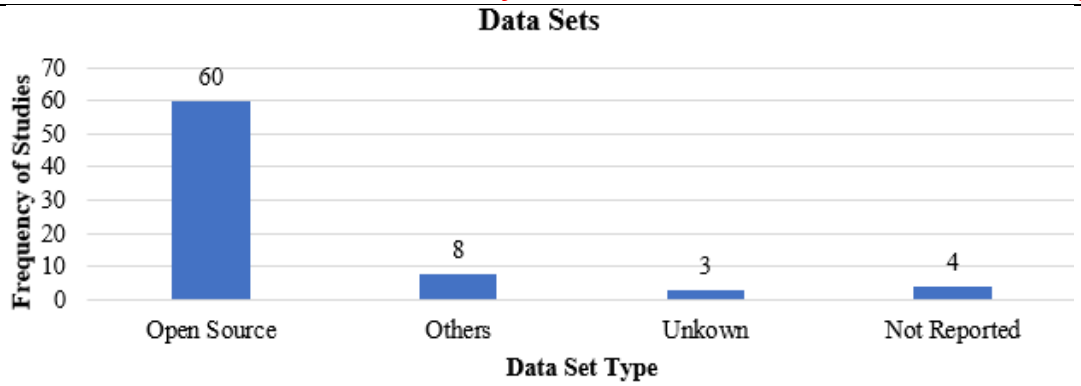

Figure 7 Dataset Type with Frequency of the selected Primary Studies

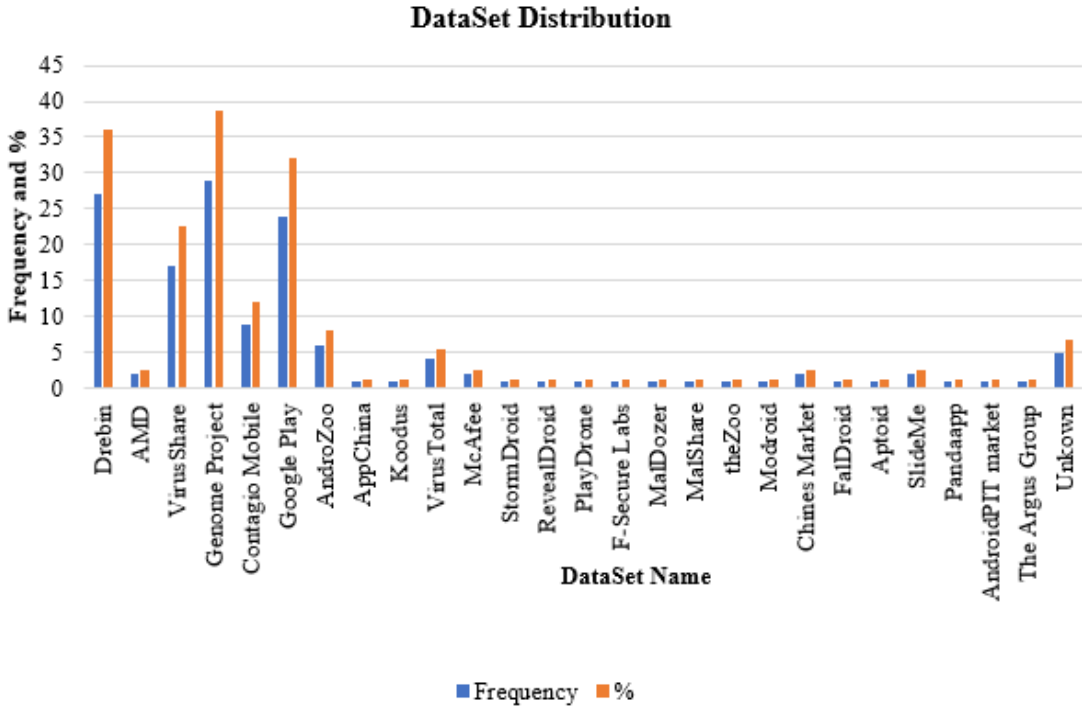

Figure 8 Frequency of the Datasets Used

RQ4: Which Source Code Analysis Methods are used by Android Malware Analysis Techniques and Tools?

We further divide the group of static source code analyses into four subcategories as shown in Figure 9.

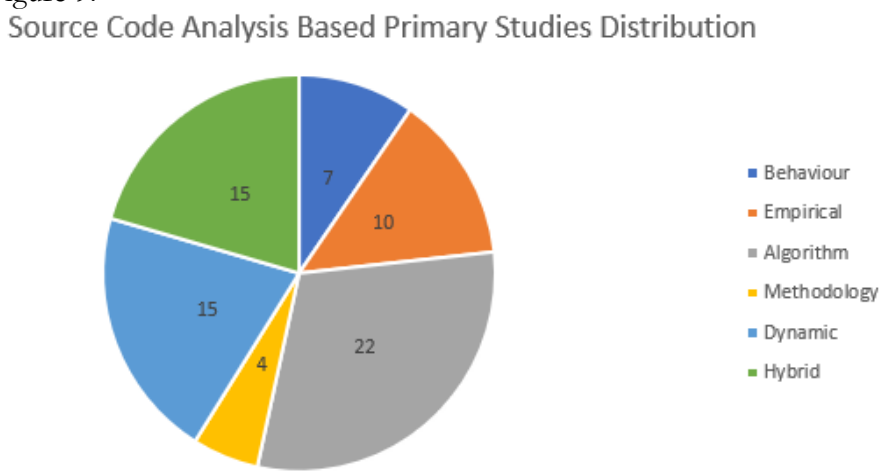

Figure 9 Source Code Analysis Based Primary Studies Distribution

In Table 10 conditions mentioned used by primary studies for Android malware detection and analysis. In Table 11, ten studies based on the extracted information rely on Empirical Source Code Analysis Methods. In Table 12, studies are summarized based on information that uses different Algorithms for malware classification.

Table 10: Behavioural Source Code Analysis-Based Studies

Pre-Condition	Post-Condition	Ref
The multi-view context-aware approach is adopted for malicious code localization	Source Code Metrics	[69]
Three major components signature database, Android client, and central server are used for analysis	A web-based portal is developed	[25]

Table 11 Empirical Source Code Analysis-Based Studies

Pre-Condition	Post-Condition	Ref
Java source code is extracted using dex2jar	The clone detection method is applied	[70]
Intercomponent call graph analysis is applied and evaluated using the ProGuard[29] tool	Static taint analysis is used	[29]
For clustering of permissions, the k-means algorithm is applied	Apps are tested using Blue Stack App player	[3]
Different features are extracted using the Contextual Weisfeiler-Lehman kernel algorithm [71]	CASANDER an online learning-based framework is developed	[72]

Table 12: Algorithm-based Source Code Analysis Studies

Pre-Condition	Post-Condition	Ref
Extracted Opcodes from android package, Source code metrics are used	A generative Adversarial Network is used	[1]
Extracted Opcodes from android package, Source code metrics are used	K-max pooling method is used on different datasets	[34]
Source code metrics results are used	SVM Classifier is applied	[26]
N-gram analysis is applied to extract a multilevel fingerprint	The performance metric is applied to the evaluation	[73]

Android malware detection methods based on dynamic source code analysis is reported in Table 13. In Table 14, information on the studies based on hybrid source code analysis is reported.

Table 13: Dynamic Source Code Analysis Studies

Pre-Condition	Post-Condition	Ref
Analysis and interception modules are implemented based on malware behavior analysis	Feature-weighted SVM is used with accuracy, precision, and recall metrics for measuring better performance	[45]
Logistic regression [74], KNN [75], Arrow [76], SVM, and Naïve Bays algorithms are used on frequency and co-occurrences	KNN Classifier is implemented using the WEKA[25] tool	[46]
The behavior-based malware detection system is implemented called AMDS	AMDS [77] is evaluated on Nexus 4 based on jellybean 4.2.2	[47]

Table 14: Hybrid Source Code Analysis Studies

Pre-Condition	Post-Condition	Ref

Different types of misbehaviors of android malware are noticed using Madam Detection	Four datasets are used to determine the effectiveness of Madam	[55]
Source Code Metrics	The accuracy of the classification is enhanced by using the OKNN Machine Learning algorithm	[56]
Malware features are extracted from APK files using Androguard	The classification model is generated using the lazy association learning (LAC) algorithm	[57]

RQ5: **What are the Research Gaps and Future Research Directions for Android Malware Analysis Techniques and Tools?**

We summarize the following research gaps that require the attention of the research community.

- Integration of different existing and new analysis methods for the detection of malware.
- Generic frameworks for evaluation of malware detection techniques.
- APIs Call analysis for detection of malware.

D. Pragmatic Evidence-Based Proposed Theoretical Framework:

a) Motivation:

The techniques which are based on permission analysis may not detect malicious application which has no permission or single permission [12]. So, detection techniques based on permission as a feature set may not solely detect the malware samples. [14]. Therefore, we need to develop a system that does not only apply for permission as a feature set but also considers the other components of the Android APK files such as API Call logs, API system call analysis, intent filters, and hardware component analysis. The studies [35][77][20][56][61] apply multiple feature extraction for android malware analysis and detection. But these studies have the limitations of static or dynamic analysis techniques which are discussed in the introduction section. Moreover, hybrid analysis techniques use dynamic analysis in combination with static analysis. Therefore, dynamic analysis in a few cases depends on the communication with the server. In the case of congestion or the link of the system gets down, the android application may not communicate with the server and as a result performance of the system may be affected [53]. The techniques which use machine learning methods for the classification of android apps can be replaced with deep neural networks because large datasets can be analyzed with much better performance [28].

b) Proposed Framework:

Based on the above discussion in the introduction and motivation section, we are purposing a new framework shown in Figure 10. This framework will be based on the hybrid analysis of the Android malware, which will address the limitations of the static and dynamic analysis methods. We will also investigate Deep Neural Network methods that may improve performance for large datasets.

c) Datasets:

We have identified in our systematic literature review that the most used datasets are Genome [78] 39% and Drebin [79] 36%. We will use Koodous Virushare, Maldroid, or Androzoo datasets for malicious apps. The benign apps will be collected from the Google play store and the Chinese app market for testing and training purposes [80].

d) Performance Metrics:

We will use the following performance metrics in Table 15 to analyze the android malware detection framework.

Table 15: Android Malware Detection Performance Metrics

Metrics	Description
True Positive Rate (TPR)	True Positive / Total Positives
False Positive Rate (FPR)	False Positive / Total Negatives
Accuracy (ACC)	True Positives + True Negatives / Total Instances
True Positives (TP)	Number of malware applications classified as malware
False Positives (FP)	Number of legitimate applications classified as legitimate
Recall (RC)	TP / TP+FN
F1-Score	2*PR*RC / PR+RC

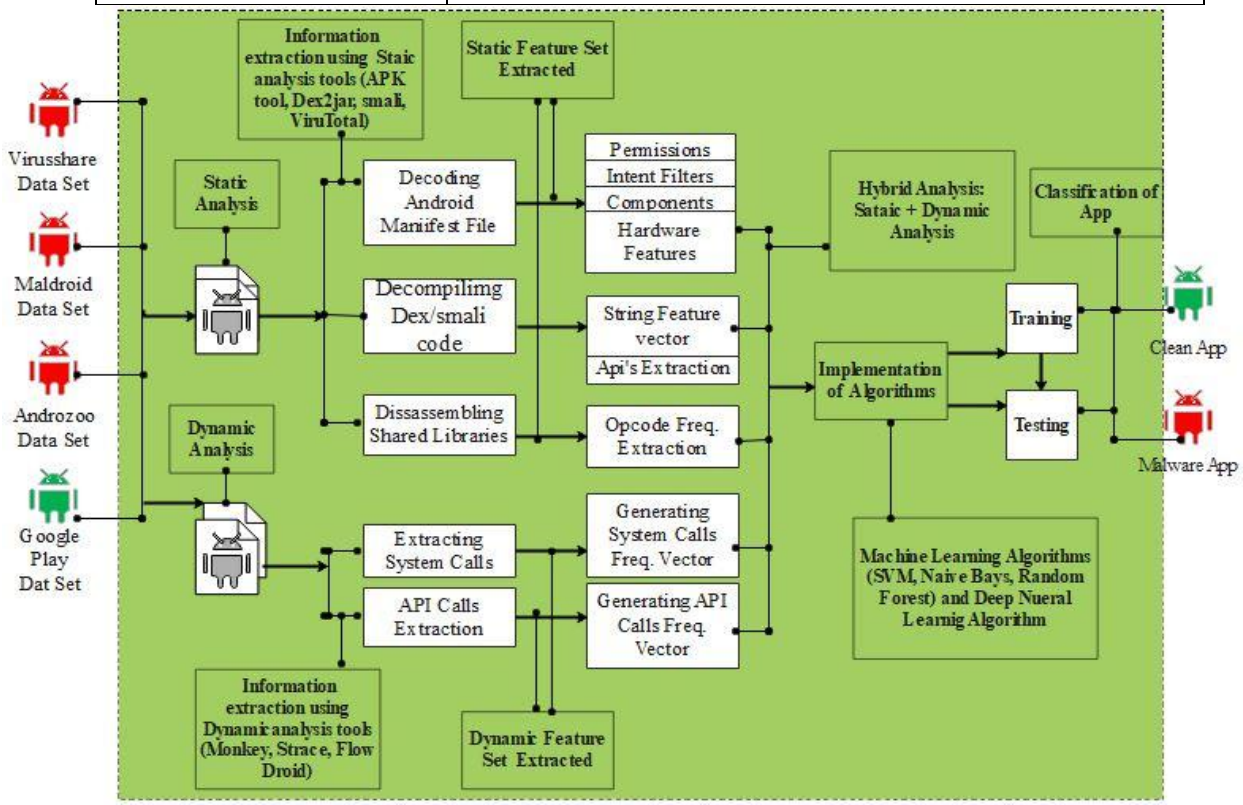

Figure 10 Proposed Framework

Conclusion and Future Work.

Authors are advised to submit concluding remarks about their findings with recommendations for further work. This work sums up the best-in-class methods and gives an extensive outline of Android malware identification techniques utilizing static, dynamic, and hybrid analysis. Solidly, this SLR is conducted by undertaking 75 primary studies from different online libraries published between August 2010 to January 2021 and analyzing five important research questions.

References

1. M. Amin, B. Shah, A. Sharif, T. Ali, K. L. Kim, and S. Anwar, "Android malware detection through generative adversarial networks," Transactions on Emerging Telecommunications Technologies, vol .30, pp. 1-29, issue. e3675, 2019.
2. P. Faruki, V. Laxmi, A. Bharmal, M. Gaur, and V. Ganmoor, "AndroSimilar: Robust signature for detecting variants of Android malware," Journal of Information Security and Applications, vol. 22, pp. 66–80, 2015.
3. S. B. Almin and M. Chatterjee, "A Novel Approach to Detect Android Malware," Procedia Computer Science, vol. 45, pp. 407–417, 2015.

4. Z. Ma, H. Ge, Y. Liu, M. Zhao and J. Ma, "A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms," in IEEE Access, vol. 7, pp. 21235-21245, 2019.
5. M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, and T. Liu, "Android Malware Familial Classification and Representative Sample Selection via Frequent Subgraph Analysis," IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 1890–1905, 2018.
6. H. Papadopoulos, N. Georgiou, C. Eliades, and A. Konstantinidis, "Android malware detection with unbiased confidence guarantees," Neurocomputing, vol. 280, pp. 3–12, 2018.
7. P. Palumbo, L. Sayfullina, D. Komashinskiy, E. Eirola, and J. Karhunen, "A pragmatic android malware detection procedure," Computers & Security, vol. 70, pp. 689–701, 2017.
8. L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version)," ACM Transactions on Privacy and Security, vol. 22, no. 2, pp. 1–34, Oct. 2019.
9. C. Bae and S. Shin, "A collaborative approach on host and network level android malware detection," Security and Communication Networks, vol. 9, no. 18, pp. 5639–5650, 2016.
10. H. Zhang, S. Luo, Y. Zhang, and L. Pan, "An Efficient Android Malware Detection System Based on Method-Level Behavioral Semantic Analysis," in IEEE Access, vol. 7, pp. 69246-69256, 2019.
11. L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini and E. De Cristofaro, "A Family of Droids-Android Malware Detection via Behavioral Modeling: Static vs Dynamic Analysis," 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, pp. 1-10, 2018.
12. K. A. Talha, D. I. Alper, and C. Aydin, "APK Auditor: Permission-based Android malware detection system," Digital Investigation, vol. 13, pp. 1–14, 2015.
13. K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, "DroidEvolver: Self-Evolving Android Malware Detection System," 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, pp. 47-62, 2019.
14. D. Li, L. Zhao, Q. Cheng, N. Lu, and W. Shi, "Opcode sequence analysis of Android malware by a convolutional neural network," Concurrency and Computation: Practice and Experience; e5308, pp. 1-18, Sep. 2019.
15. A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "A multi-view context-aware approach to Android malware detection and malicious code localization," Empirical Software Engineering, vol. 23, no. 3, pp. 1222–1274, 2018.
16. A. Narayanan, G. Meng, L. Yang, J. Liu and L. Chen, "Contextual Weisfeiler-Lehman graph kernel for malware detection," 2016 International Joint Conference on Neural Networks (IJCNN), pp. 4701-4708, , 2016,.
17. A. Arora, S. K. Peddoju and M. Conti, "PermPair: Android Malware Detection Using Permission Pairs," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1968-1982, 2020.
18. Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in Proc. IEEE Symp. Secur. Privacy, pp. 95–109, , May 2012.
19. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," in Proc. NDSS, pp. 23–26, 2014.

20. C. Li, K. Mills, D. Niu, R. Zhu, H. Zhang and H. Kinawi, "Android Malware Detection Based on Factorization Machine," in IEEE Access, vol. 7, pp. 184008-184019, 2019.
21. K. Xu, Y. Li and R. H. Deng, "ICCDetector: ICC-Based Malware Detection on Android," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252-1264, June 2016.
22. V. Vovk , A. Gammerman , G. Shafer , "Algorithmic Learning in a Random World", Springer, New York, 2005 .
23. Y. Zhang, W. Ren, T. Zhu, and Y. Ren, "SaaS: A situational awareness and analysis system for massive android malware detection," Future Generation Computer Systems, vol. 95, pp. 548–559, 2019.
24. R. Sartea, A. Farinelli, and M. Murari, "SECUR-AMA: Active Malware Analysis Based on Monte Carlo Tree Search for Android Systems," Engineering Applications of Artificial Intelligence, vol. 87, p. 103303, Jan. 2020.
25. P. Feng, J. Ma, C. Sun, X. Xu and Y. Ma, "A Novel Dynamic Android Malware Detection System with Ensemble Learning," in IEEE Access, vol. 6, pp. 30996-31011, 2018.
26. S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," in IEEE Access, vol. 6, pp. 4321-4339, 2018.
27. T. Kabakus and I. A. Dogru, "An in-depth analysis of Android malware using hybrid techniques," Digital Investigation, vol. 24, pp. 25–33, 2018.
28. P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," Journal of Systems and Software, vol. 80, no. 4, pp. 571–583, 2007.
29. K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," in IEEE Access, vol. 8, pp. 124579-124607, 2020.
30. H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen, and L. Cheng, "DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model," Neurocomputing, vol. 272, pp. 638–646, 2018.
31. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided Android malware classification," Comput. Electr. Eng., vol. 61, pp. 266–274, 2017.
32. F. Alswaina and K. Elleithy, "Android Malware Permission-Based Multi-Class Classification Using Extremely Randomized Trees," in IEEE Access, vol. 6, pp. 76217-76227, 2018.
33. A. Skovoroda and D. Gamayunov, "Automated Static Analysis and Classification of Android Malware using Permission and API Calls Models," 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, pp. 243-24309, , 2017.
34. Z. Wang, C. Li, Z. Yuan, Y. Guan, and Y. Xue, "DroidChain: A novel Android malware detection method based on behavior chains," Pervasive Mob. Comput., vol. 32, pp. 3–14, 2016.
35. K. Tian, D. Yao, B. G. Ryder, G. Tan and G. Peng, "Detection of Repackaged Android Malware with Code-Heterogeneity Features," in IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 1, pp. 64-77, 1 Jan.-Feb. 2020
36. T. Lei, Z. Qin, Z. Wang, Q. Li and D. Ye, "EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6668-6680, Aug. 2019.

37. H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in Proceedings of the 2013 ACM workshop on Artificial intelligence and security - AISec '13, pp. 45-54, , 2013.
38. K. O. Elish, X. Shu, D. (daphne) Yao, B. G. Ryder, and X. Jiang, "Profiling user-trigger dependence for Android malware detection," Comput. Secur., vol. 49, pp. 255–273, 2015.
39. S. Wu, P. Wang, X. Li, and Y. Zhang, "Effective detection of android malware based on the usage of data flow APIs and machine learning," Inf. Softw. Technol., vol. 75, pp. 17–25, 2016.
40. J. Chen, M. H. Alalfi, T. R. Dean, and Y. Zou, "Detecting android malware using clone detection," J. Comput. Sci. Technol., vol. 30, no. 5, pp. 942–956, 2015.
41. J. Garcia, M. Hammad, and S. Malek, "Lightweight, obfuscation-resilient detection and family identification of android malware," ACM Trans. Softw. Eng. Methodol., vol. 26, no. 3, pp. 1–29, 2018.
42. T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A Multimodal Deep Learning Method for Android Malware Detection Using Various Features," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773-788, March 2019.
43. K. Xu, Y. Li, R. H. Deng and K. Chen, "DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks," 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, pp. 473-487, 2018.
44. L. Deshotels, V. Notani, and A. Lakhotia, "DroidLegacy: Automated familial classification of android malware," in Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop 2014 - PPREW'14, pp. 1-12, 2014.
45. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and accurate zero-day android malware detection," in Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12, pp. 281-294, 2012.
46. G. Meng, Y. Xue, Z. Xu, Y. Liu, J. Zhang, and A. Narayanan, "Semantic modelling of Android malware for effective malware comprehension, detection, and classification," in Proceedings of the 25th International Symposium on Software Testing and Analysis - ISSTA, pp 306–317, 2016.
47. F. Shen, J. D. Vecchio, A. Mohaisen, S. Y. Ko and L. Ziarek, "Android Malware Detection Using Complex-Flows," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, pp. 2430-2437, 2017.
48. G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "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.
49. S. Zou, J. Zhang, and X. Lin, "An effective behavior-based Android malware detection system," Secur. Commun. Netw., vol. 8, no. 12, pp. 2079–2089, 2015.
50. X. Xiao, X. Xiao, Y. Jiang, X. Liu, and R. Ye, "Identifying Android malware with system call co-occurrence matrices," Trans. emerg. telecommun. technol., vol. 27, no. 5, pp. 675–684, 2016.
51. L. Gheorghe et al., "Smart malware detection on Android," Secur. Commun. Netw., vol. 8, no. 18, pp. 4254–4272, 2015.
52. X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM," Multimed. Tools Appl., vol. 78, no. 4, pp. 3979–3999, 2019.

53. P. Vinod and P. Viswalakshmi, "Empirical evaluation of a system call-based android malware detector," Arab. J. Sci. Eng., vol. 43, no. 12, pp. 6751–6770, 2018.
54. V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. de Geus, "Identifying Android malware using dynamically obtained features," J. comput. virol. hacking tech., vol. 11, no. 1, pp. 9–17, 2015.
55. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for Android," in Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices - SPSM '11. pp. 15-26, 2011.
56. H. Cai, N. Meng, B. Ryder and D. Yao, "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455-1470, June 2019.
57. S. K. Dash et al., "DroidScribe: Classifying Android Malware Based on Runtime Behavior," 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, pp. 252-261, 2016,
58. A.sShabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "'Andromaly': a behavioral malware detection framework for android devices," J. Intell. Inf. Syst., vol. 38, no. 1, pp. 161–190, 2012.
59. P. Teufl, M. Ferk, A. Fitzek, D. Hein, S. Kraxberger, and C. Orthacker, "Malware detection by applying knowledge discovery processes to application metadata on the Android Market (Google Play): Android Malware Detection based on Metadata Analysis," Secur. Commun. Netw., vol. 9, no. 5, pp. 389–419, 2016.
60. A. Saracino, D. Sgandurra, G. Dini and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 1, pp. 83-97, 1 Jan.-Feb. 2018
61. J. Lin, X. Zhao, and H. Li, "Target: Category-based android malware detection revisited," in Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '17, pp. 1-9, 2017.
62. J. Yu, Q. Huang, and C. Yian, "DroidScreening: a practical framework for real-world Android malware analysis: A practical framework for real-world Android malware analysis," Secur. Commun. Netw., vol. 9, no. 11, pp. 1435–1449, 2016.
63. J.-W. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim, "Detecting and classifying method based on similarity matching of Android malware behavior with profile," Springerplus, vol. 5, no. 1, p. 273, 2016.
64. A. Altaher, "An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features," Neural Comput. Appl., vol. 28, no. 12, pp. 4147–4157, 2017.
65. A. Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," Neurocomputing, vol. 396, pp. 599–608, 2020.
66. Z.-U. Rehman et al., "Machine learning-assisted signature and heuristic-based detection of malwares in Android devices," Comput. Electr. Eng., vol. 69, pp. 828–841, 2018.
67. A. I. Ali-Gombe, B. Saltaformaggio, J. "ram" Ramanujam, D. Xu, and G. G. Richard III, "Toward a more dependable hybrid analysis of android malware using aspect-oriented programming," Comput. Secur., vol. 73, pp. 235–248, 2018.
68. S. Y. Yerima and S. Sezer, "DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection," in IEEE Transactions on Cybernetics, vol. 49, no. 2, pp. 453-466, Feb. 2019

69. J. H. Abawajy and A. Kelarev, "Iterative Classifier Fusion System for the Detection of Android Malware," in IEEE Transactions on Big Data, vol. 5, no. 3, pp. 282-292, 1 Sept. 2019
70. S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu, "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.
71. E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," Digit. investig., vol. 24, pp. 48–59, 2018.
72. A. Narayanan, M. Chandramohan, L. Chen and Y. Liu, "Context-Aware, Adaptive, and Scalable Android Malware Detection Through Online Learning," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 1, no. 3, pp. 157-175, June 2017.
73. L. Zhang, V. L. L. Thing, and Y. Cheng, "A scalable and extensible framework for android malware detection and family attribution," Comput. Secur., vol. 80, pp. 120–133, 2019.
74. M. Nauman, T. A. Tanveer, S. Khan, and T. A. Syed, "Deep neural architectures for large scale android malware analysis," Cluster Comput., vol. 21, no. 1, pp. 569–588, 2018.
75. A. Martín, H. D. Menéndez, and D. Camacho, "MOCDroid: multi-objective evolutionary classifier for Android malware detection," Soft Comput., vol. 21, no. 24, pp. 7405–7415, 2017.
76. Y. Pan, X. Ge, C. Fang and Y. Fan, "A Systematic Literature Review of Android Malware Detection Using Static Analysis," in IEEE Access, vol. 8, pp. 116363-116379, 2020.
77. T. Cover, "Estimation by the nearest neighbor rule," in IEEE Transactions on Information Theory, vol. 14, no. 1, pp. 50-55, January 1968.
78. Hosmer Jr DW, Lemeshow S, Sturdivant RX. Logistic regression for matched case–control studies. Applied Logistic Regression, Third Edition:, pp. 243–268, 1989.
79. Crammer K, Kulesza A, Dredze M. Adaptive regularization of weight vectors. In Advances in Neural Information Processing Systems, Columbia, Canada, pp. 414–422, , 2009.
80. A. Veloso, W. Meira and M. J. Zaki, "Lazy Associative Classification," Sixth International Conference on Data Mining (ICDM'06), pp. 645-654, 2006.