Computer Science Review 40 (2021) 100373

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Malicious application detection in android — A systematic literature review

Tejpal Sharma *, Dhavleesh Rattan

Department of Computer Science and Engineering, Punjabi University Patiala, 147002, Punjab, India

ARTICLE INFO

Article history:
Received 29 April 2020
Received in revised form 3 January 2021
Accepted 27 January 2021
Available online 13 February 2021

Keywords:
Android malicious applications
Malware detection
Machine learning
Mobile malware
Smartphone's

ABSTRACT

Context: In last decade, due to tremendous usage of smart phones it seems that these gadgets became an essential necessity of day-to-day life. People are using new technologies and storing prominent data in their smartphones. Unfortunately, data related to privacy is center of attraction for attackers. Therefore, attackers are developing new techniques to steal the data from smartphones.
Objective: The objective of study is to report a systematic literature review regarding malicious application detection in android operating system.
Method: Standard systematic literature review method is used to carry out the research. In this, 380 research articles are studied which are published in various prominent international journals and conferences.
Results: The different techniques which are used to investigate malicious application are identified. Furthermore, features used in static and dynamic technique are classified according to their usage in recent approaches. Various hybrid methods are analyzed and mapped according to the combination of static and dynamic features used. A variety of machine learning techniques are also identified and categorized in different classes. The datasets are listed are taken from various previous research approaches.
Conclusion: This research will help to identify malicious applications in android operating system. New hybrid techniques must be implemented to investigate malware activities and recommendations are given for future research.

© 2021 Elsevier Inc. All rights reserved.

Contents

* Corresponding author.
E-mail addresses: tejpal3205@gmail.com (T. Sharma), dhavleesh@gmail.com (D. Rattan).

https://doi.org/10.1016/j.cosrev.2021.100373
1574-0137/© 2021 Elsevier Inc. All rights reserved.

1. Introduction and motivation

With the addition of smart phones in day-to-day life, its involvement in our life gets increased. It is involved in all daily activities of a person:

- Whether it is phone call to friend/relative.
- Capturing a precious life moment.
- Doing any important bank transaction.
- Or storing valuable data to a digital storage place.

Now in these days, the utilization of mobile phones gets increased rapidly and it becomes source of all the things which were implemented/ executed by telephone, camera, computers and storage devices.

It is obvious, if something is used very much by public then attacker focus on that very quickly because it will give them more benefits. They can target large amount of people. So, with the increase of mobile phone utilization, mobile malwares or malicious applications that are used in mobile phones get increased.

Researchers are working on it day to day, new malwares developed and then antivirus companies and other research organization (may be individual researcher) working on it. There are various kinds of techniques are developed to track these malwares/malicious applications. A number of literatures surveys are conducted regarding these techniques which are used to detect malicious applications. Our main focus is on android mobile applications because 75% mobile users in this world are using android operating system oriented mobile phones [1].

1.1. Motivation for work

This paper will report to various malware/malicious application detection methods and report findings in this area. Systematic literature review takes much time to complete but it provides all comprehensive information about the area and also leads towards the current research possibilities. This review paper will discuss various key areas of research in malware/malicious application in android mobile phones and the major motive for work is:

- About 75% people are using android mobile phones in this world and malware growth is increasing day by day [1]. This research work will find various malicious application detection methods which are used and infer results from then for future research.
- After analyzing, android malware detection research, we realized that some points which sill uncovered in the current literature surveys that we will include in our research paper and that will be helpful for further investigation in this research area.

2. Background

2.1. Android

2.1.1. Introduction

Mobile phones have come a long way from the times when they were first introduced. But began as a luxury commodity became main stream in about a couple of decades time. During early years, it primarily remained as a means of communication, mostly centered on making calls and the Short Messaging Service [2]. The advent of smartphones, however, completely transformed the scope of use of this electronic gadget in a person's life. Smart devices like smartphones, tablets etc carry immense portable computing power at the touch of a finger, which requires the storage and transfer of a lot of information and data. This includes a lot of sensitive and valuable data like contacts details, photographs, bank details, passwords, etc. Smartphone users are mostly unaware of the risk posed by the intruders who are always looking to acquire access to this data [3,4].

Android, developed by Google, is an operating system based on the Linux kernel. Mobile OS market is headed by Android OS, according to the recent research 85% of the smartphones in all over world are using Android operating System [5]. As per the reports Play store is growing 3 times the rate of Apple application store [5]. This growth makes it popular and which leaded it towards the target for the malwares. As the Android devices are increasing day by day in the market and it also breaches uses privacy, e.g., access to contacts and GPS coordinates, money making from premium SMS/Calls and other private information from mobile. Recent studies have mentioned that how mobile market have been abused to host malware or repackaged applications embedding malicious components [4,6,7].

2.1.2. Android market growth

Android was launched in 2008 and the smartphone market share of the operating system has risen from about 4% in 2009 to a staggering 75% in 2019 [1,8]. This stellar growth can be attributed to a number of advantages that it provides over the competitive operating systems. The primary reason for such an impressive

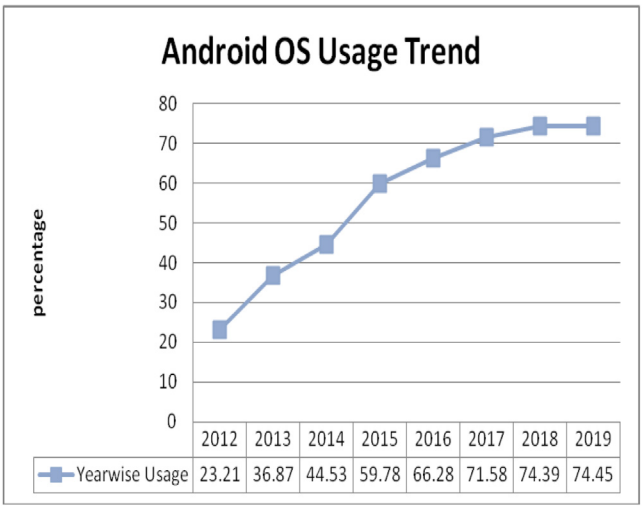

Fig. 1. Android market growth [8,10].

growth in Android's market share is the backing of the Open Handset Alliance (OHA). The OHA is a multinational consortium of device manufacturers, components manufacturers, software developers, network operators etc which aims to develop open standards for mobile devices. But it basically comes down to is that the development and success of Android is not dependent on a limited variety of devices. The range of hardware takes care of the requirements of varied consumers based on performance, affordability, aesthetics etc. [9] (see Fig. 1).

2.1.3. Android OS overview

Android was released in 2008 and was primarily developed as an open-source operating system for handheld devices with touch capabilities like smartphones and tablets. Android uses a Linux kernel to manage system resources and all the Java written applications run in isolated environment. What it means is that each application runs within its own Dalvik virtual machine. Starting Android 5.0, the Dalvik just-in-time compiler was replaced with an ART compiler which stands for ahead-of-time. The ART compiler is also compatible with just in time compiler in some cases. The Android hardware contains a base-band ARM processor, a separate processor for running applications, and various devices chipsets such as Bluetooth and GPS [2]. Fig. 2 shows an overview of the Android architecture. This architecture divides the system into 4 systematic layers, which contains application, application framework, libraries and kernel drivers. So as to get to the framework, permissions must be taken by the all Java applications from the Android Permission System during the process of installation. Kernel grants the permissions to android applications and it is only through well-defined API method calls that these applications are able to interact with each other and the system. The Android applications are divided into four main components comprising of activities, services, broadcast receivers and content providers. The role of the content providers is to manage the data between the applications such as sharing of data between applications. Activities represent the user interface with the application and on the other hand services take care about the background process running for the application. Intents in Android are necessary for the operation when one component requests another component to perform some task, and then it communicates though message to perform that task. As the broadcast receivers runs in the background, they allow the events to be passed to the applications by the system, outside of a regular user flow. This allows the application to respond to broadcast announcements originating anywhere in the system [2,11].

2.2. Android malware

2.2.1. Android malware introduction

Smartphones have revolutionized the lives of people around the world. With the smartphone technology advancing swiftly, there has been an exponential growth in smartphone sales and number of people using them. Over the past three-year smartphone shipments have increased three times from about 40 million to 120 million [4]. Unfortunately, the increased use of smartphones in day-to-day life has also led to a rapid increase in mobile malwares. As Google's Android surpassed its competition (e.g. Apple iOS) in becoming the most popular mobile platform, it unfortunately also became the most malware affected mobile platform. It has been highlighted that "among all mobile malware, the share of Android based malware is higher than 46% and still growing rapidly" [4,12].

Given its popularity, there are several hundreds of thousands of applications available to the Android users to add-to the functionality of their devices. Unfortunately, this has also left a lot of android users susceptible to a lot of malware attacks [13]. As compared to other operating systems, Android operating system allows their users to install applications from unverified sources (such as third-party markets), which makes the attackers easy to distribute malicious applications. A recent study shows that in 2012 alone, the number of malicious applications has reached over 55,000 and there were 119 new malware families discovered during the same year [14]. Now, this leads us to believe that there is an urgent need to stop the production of malware on Android markets and smartphones [14–17].

2.2.2. History

In contrast to a popular myth that no viable security threats in the earlier mobile phones, the development of devices which could communicate with each other, e.g. through Bluetooth and infrared, saw a surge in the malicious code being written for the operating systems of the time [18]. Android, in its initial stages, was no different. In Fact, when the first version of the Android OS was released to the public, its security flaws were exposed by a wave of malicious attacks [19]. A Chinese security research group named Blitz Force Massada announced that the operating system had suffered multi-module attacks affecting the very basic functionality of a mobile phone. Some of the attacks resulted in Radio turn offs, data gathering and leakage to the attacker, automatically triggering the phone to accept all phone calls, end ongoing phone calls etc. The most interesting thing about this attack was that it happened before first device running on Android was released to the market [20,21].

There were further attacks using malicious code and whose main objective was to collect private and sensitive information from the users, running in the background in stealth mode, without the user's knowledge. Spyware, like Mobile Spy, were found in Android devices and it secretly monitored calls, texts, GPS coordinates browser history etc. Financial institutions and their customers were targeted by fraudulent apps that resembled the official apps. The main objective of these apps was to leak user provided account details and passwords to the attacker [21].

From the initial stages of the operating system, Android has been known to be vulnerable to phishing attacks as well. It has been demonstrated by researchers from Trustwave that it was possible to develop kernel-level rootkits for Android that could be delivered over the air or through a rogue app. These rootkits could be triggered by something as simple as a call from a particular number. The developers at Kaspersky also found SMS Trojans, like FakePlayer, that sent malicious texts to premium numbers resulting in financial losses to the users who were charged for these premium services without their knowledge or consent. Some

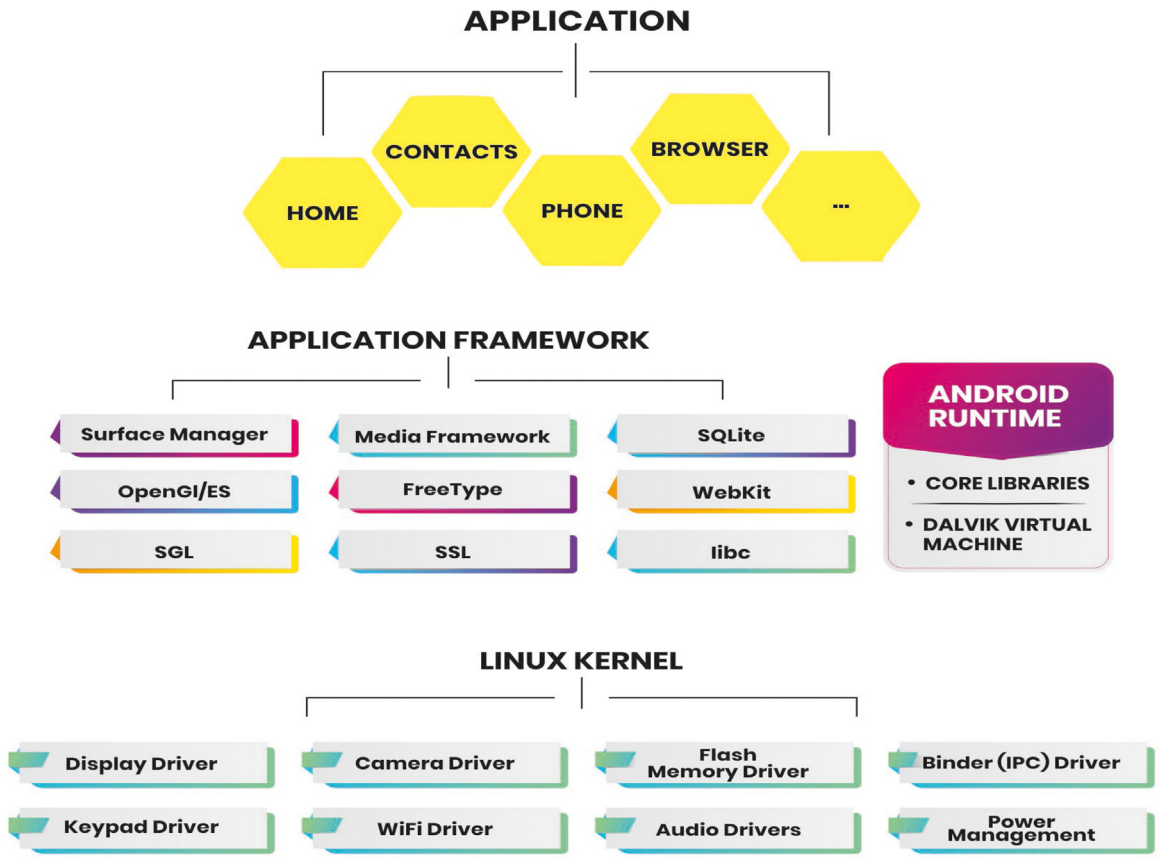

Fig. 2. Android OS functioning [2].

fraudulent apps mimicked some famous game apps to get the user to install them. These apps would leave a backdoor which the attacker can exploit at a later stage. Geinimi, on such spyware, was distributed by packaging it with genuine applications, which made it harder for the Google Playstore to screen it as malicious software. These applications were said to be "Trojanized". Another feature that made this type of malware dangerous was the "botnet" like capability. It allowed these malware infected apps to receive commands remotely, leaving very few commands in the source code itself which could be detected during the security analysis [21].

Although with time, Google Playstore made considerable efforts to monitor the store better but users were still at risk. DroidDream, a malware, infected 50 apps in the Playstore. Like Geinimi, it used the repackaging of official apps for distribution but what made it unique was its ability to use publicly known exploits in the Android community to gain root access and privileges in an Android device [21].

2.2.3. Types of malware

There are numerous ways in which android devices could be contaminated with malware. Some of the most prevalent ways are listed as follows: [21].

- *Repackaging a legitimate application:* One of the most commonly used methods used by the malware authors is to disassemble a popular but genuine application package and inject malicious code block or payload into it. The new package is then reassembled and made available for download on the official Android market store (Google PlayStore) or, preferably, on a third-party market. The official store does have checks and balances to deter such activity that third party stores and websites seldom do. An android user is at risk of a malware attack if he/she downloads and installs application packages from unverified sources [21–24].
- *Exploiting bugs in an Android application:* A bug in an Android application could also be used as vulnerability by an attacker. Applications that are not well supported or are not updated in a timely manner by the user are more susceptible to such attacks. The attacker may find a bug and use that vulnerability to compromise the user's data or information.
- *Fake applications:* Attacker use fake applications to lure users into downloading them and installing them on their devices. Such an application promises to provide a desirable functionality that is either not commonly available yet or has to be paid for in some other legitimate application. The malicious code, instead, provides a backdoor to the attacker, who may access the device and cause harm in various ways. For instance, during the early days of Android, some utility applications claimed to provide easy access to using the camera flash as an emergency torch were known to fish user data and relay it to the attackers who developed the app.
- *Remote install:* This type of attack usually consists of an attacker remotely installing malicious code onto a user's device, without his knowledge. The user's certificates are compromised by the attacker and passed on in the marketplace as his own, allowing for the malware to be installed into the device without the user's permissions. This malware too will provide a backdoor that allows the attackers to access user's data or personal information such as photographs, bank details, contact list etc.

2.2.4. Android security issue

As mentioned earlier, the Android OS is based on the Linux kernel and hence it follows a similar security model to the Linux

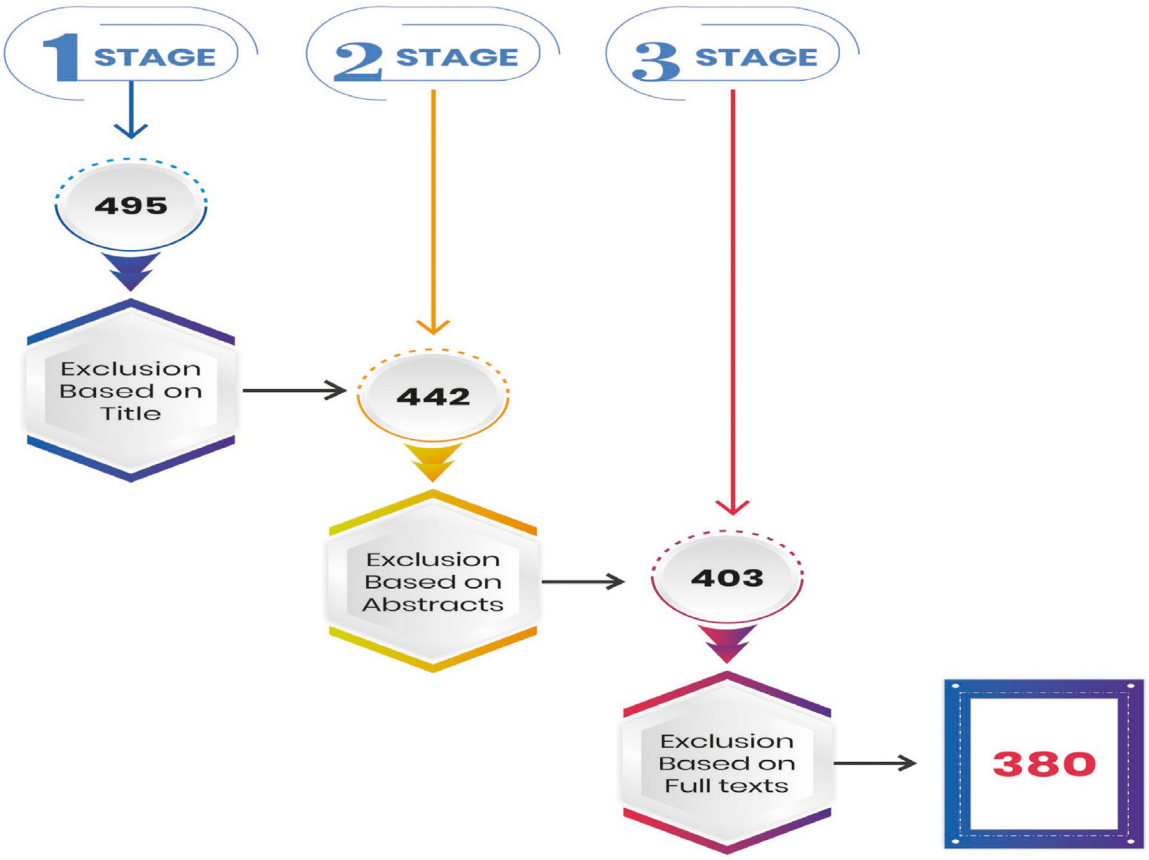

Fig. 3. Study selection procedure.

distributions. Android employs mechanisms like sandboxing, assigning signatures and the provision to users of assigning permissions to applications for the purpose of security [4]. The permission mechanism is used for limiting the scope of applications in allowing or denying them access to private or sensitive data, system or software resources and hardware resources. Like Linux, every app is part of a group and it derives permissions allocated to that group or classification. If an app is allowed or denied a system resource is decided by what class the application belongs to. Manifest file of android application contains the information about the permissions which are granted to the app [24]. In sandbox, isolated environment is provided to the system to perform analysis. Some systems are using application signature to detect malware applications, they use the hashing of application or finds any string that can identify that application. Although this mechanism is simple and it also has some defects that cannot protect the user's private information adequately [25–27].

3. Review method

Systematic literature review reported in the paper will include the following factors [28–31]:

- Development of Review Protocol
- Conducting Review
- Analyzing the Results
- Reporting the Results
- Discussion of Findings

3.1. Planning the review

The review protocol includes [32,33]:

- framework of important research questions,
- searching of research papers from various journals and conference proceedings,
- identification of related data from searched results,
- related data extraction and analysis,
- specification of data in proper formats in the form of tables and graphs,
- Reporting of findings.

3.2. Research questions

The main goal of the research questions is to explore the existing malicious application techniques, machine learning techniques used for classification, dataset used in various approaches and questions related to various uncovered points that are not covered in existing literature reviews. The Table 1 describes primary research questions and their motivation:

3.3. Sources of information

For a comprehensive literature survey wide range of searching is required. Before the start of review, an appropriate source must be chosen. So, all significant research articles must be included for the successful review paper. There are some kinds of electronic sources which are used for searching of highly relevant research articles:

- ACM Digital Library (www.acm.org/dl).
- IEEE Explore (ieeexplore.ieee.org).
- Science Direct (http://www.sciencedirect.com).
- Springer (http://www.springerlink.com).
- Wiley Interscience (www3.interscience.wiley.com).

Table 1
Research question, sub question and motivation.

Research Questions	Motivation
1. What is the current status of mobile operating systems used in market?	It helps in understanding the present status of mobile operating systems used by people in this global market.
2. What is the current status of mobile malwares in the market?	It helps in knowing the most affected operating systems and the growth of mobile malwares.
3. What type of research is covered in recent survey papers?	It will show the comparison of various recent surveys and types of parameters that they have covered.
4. What are various techniques used for detection of malicious applications in android mobiles/devices? a. What are various static techniques used? b. What are various dynamic techniques? c. Hybrid techniques?	The research question will explore various techniques used to detection malwares or malicious applications in android devices. We will also specify the methods used in these techniques.
5. What is repackaging and various methods used to detect these types of applications?	Here, we will mention how the repackaging is used by attackers/intruders? and how it can detect.
6. What are the gaps in dynamic analysis methods and how these gaps can be overcome?	As per the recent studies there are a number of drawbacks in present dynamic analysis techniques. So, this research question will explore various gaps in current techniques.
7. How machine learning is beneficial for malicious application detection?	As machine learning techniques are in trend now, so we will try to mention various machine learning techniques/ methods used for malicious application detection.
8. What are various datasets are used for the detection process?	Dataset is the primary component of the research work, so it is highly required to know about various types of available datasets.

3.4. Search criteria

Research papers are searched from all five sources with a full intention to cover all the papers of this area. A range of keywords are used to approach a relative search. The main keywords are: Android, malware, malicious, application and detection. Different combinations of these keywords are used for this process. Table 2 shows the search strings used and other factors related to it [28,30,34] (see Fig. 3).

In total 495 papers searched from different sources and out of which after studying (title, abstract and full paper) only 380 extracted for the review process.

Fig. 4 shows the information about the publication according to the year of publications, as per the bar chart it shows that last 8 years publications are included and every year there is growth in the research and in year 2018 there are maximum numbers of publications which are included in this research work.

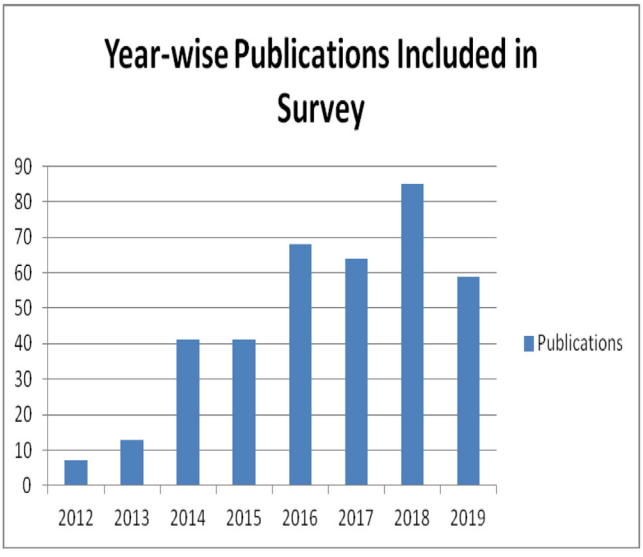

Fig. 4. Year-wise publications included in survey.

3.5. Inclusion and exclusion criteria

This is the primary process of the literature review, because it deals with the database and that database will act as the foundation for the paper. It is implemented very carefully. The inclusion and exclusion process is divided into 3 stages [28,30]:

- *Stage 1:* Papers are searched with specific keywords in all the electronic sources. In most of the cases, search is applied two times with combination of different keywords. The motive is to include all highly relevant research papers. String search is applied in the "abstract" part but in two sources string search is performed on "title", because "abstract" search was giving very irrelevant results that was very difficult to analyze. After the searching process, paper "titles" get analyzed and extremely irrelevant papers search out and excluded from the database.
- *Stage 2:* In this stage, "abstract" of the remaining articles is studied and relevant data is included and others neglected from the record. Here 403 articles left after analyzing 442.
- *Stage 3:* Final step if the Inclusion and Exclusion process, in this case full test paper is studied for the data extraction and 22 get rejected after this stage and finally left with 380 papers for this systematic literature review and all the information from these papers is included in research article

4. Results

4.1. Current status of mobile operating systems used in market

In the era of digitalization, mobile phone is the primary need of majority of people. In these mobile phones, the category of smart phone is more famous than others because it can be used for multiple purposes like calling, SMS, storage, picture capturing and working on internet. There are numerous types of operating systems are available in the market that are used in these smart phones. But as per the report from [1], 75% of the world is using Google's android operating system. The Fig. 5 shows the shares of different operating systems in global market. As the trend shows that share of android increased rapidly from 37 percent to 75 percent within 7 years. The usage of iOS is varying from 20%–25%. On the other hand, the usage of Symbian, Samsung and

Table 2
Count of papers with search strings in all electronic sources.

Sr. No.	E-Resources	Search within	Searching string	Dates	Product/content type	Count	Count (After title filtration)	Count (After abstract study)	Count (After full test study)
1	ieeexplore.ieee.org	Abstract	Android, Malware, Application, Detection	All dates	Conferences, Journals and Standards	275	245	219	206
2	www.acm.org	Title	Android, Malware, Detection	All dates	Journal, Proceedings, Transaction and Magazine	56	47	43	39
			Android, Malicious, Application			5	5	5	5
3	www.sciencedirect.com	Abstract	Android, Malicious/Malware, Application, Detection	All dates	All sources	65	55	47	45
4	www.springerlink.com	Title	Android, Malware, Detection	All dates	All sources	64	62	61	60
			Android, Malicious, Application			2	2	2	2
5	www3.interscience.wiley.com	Abstract	Android, Malicious, Application, Detection	All dates	Journals, Reference works, Databases	8	7	7	6
			Android, Malware, Detection			20	19	19	17

Table 3
Comparison of our survey with other related surveys.

	1	2	3	4	5	6	7	8	9	10	11
Years of publication → Topics covered ↓	2013	2013	2015	2015	2017	2017	2017	2018	2018	2019	Our survey
Static techniques used	√	√	√	√	√	√	√	√	√	√	√
Dynamic techniques used	√	√	√	√	√	√	√	√	√	√	√
Hybrid techniques used	√						√	√			√
Machine learning techniques used for classification	√	√			√			√			√
Current status of mobile operating systems used in market	√	√					√				√
Current status of mobile malwares in the market		√			√		√				√
Mapping of static and dynamic techniques											√
Repackaging and various methods used to detect these types of applications											√
Various datasets are used for the detection process											√
Deep learning algorithms for malicious application detection								√			√

1 — Mariantonietta La Polla et al. [35], 2 — Guillermo Suarez-Tangil et al. [36], 3 — Parvez Faruk et al. [37], 4 — Tan et al. [38], 5 — Ping Yan et al. [39], 6 — Raima Zachariah et al. [40], 7 — Kimberly Tam et al. [4], 8 — Alireza Souri et al. [41], 9 — Huasong Meng et al. [42], 10 — S. Sibi Chakkaravarthy et al. [43], 11 — Our survey.

Blackberry is decreasing every year. It shows that majority of the mobile users are preferring Google's android operating system.

4.2. Status of android mobile malwares in the market

Since the usage of mobile phones has increased rapidly from the last few years, the growth of android malware has also been at a fast pace. More and more the use of mobile phones, more the growth of Android malware. The number of users of the Android operating system is very large in number due to which the growth of malware is very easy and spreadable. As per the reports [10,44,45], the growth of Android malware since 2012 till 2018 can be summarized as follows: in 2012 malware was recorded as 0.35 million which increased by 188% in 2013, the year 2014 had an increase of 321%, the year 2015 had 66% increase, the year 2016 had 85% increase, the year 2017 had 48% increase and the year 2018 had an increase of 36%. This shows that it is increasing every year tremendously and reached at the value of 26.61 million in 2018 (see Fig. 6).

4.3. Related surveys regarding malware detection in android systems

A number of researchers have described different methods for android malicious application detection in their surveys. As per the Table 3, ten surveys are analyzed and compared with each other in tabular form. Its shows that static and dynamic techniques used for analysis are covered by all the researchers. But hybrid is described in only 3 papers. On the other hand, machine learning techniques that are used for classification of malicious and benign applications are also covered by very few authors.

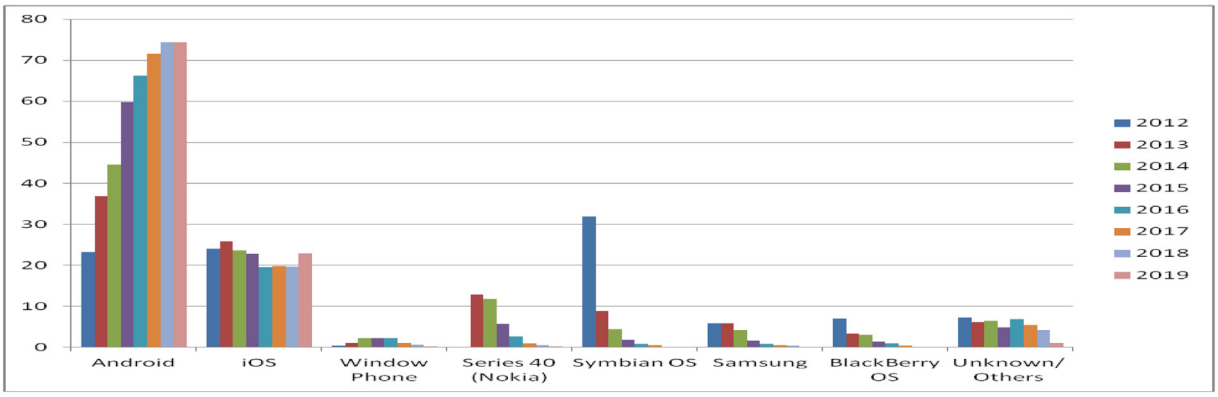

Fig. 5. Shares of different operating systems in global market.

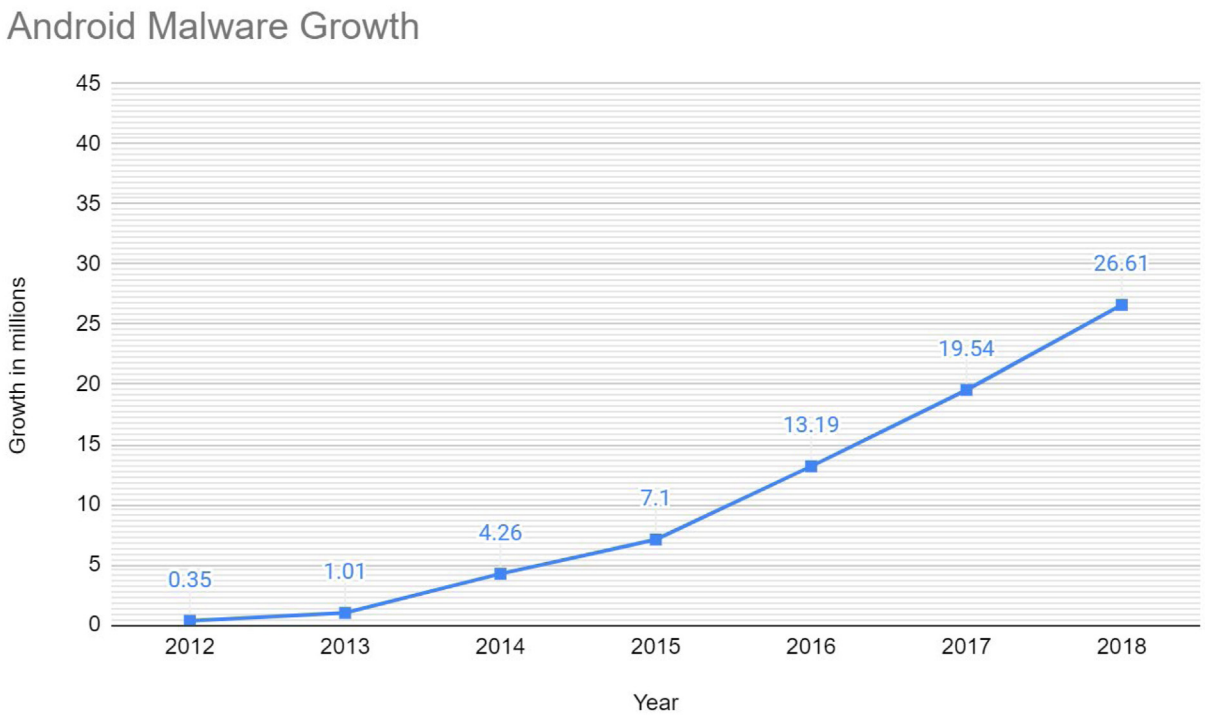

Fig. 6. Android malware growth in recent years.

In additions of these, there are some other factors which may affect the research in this field are still uncovered such as datasets used for analysis purpose, various deep learning techniques used, repackaging of applications and their detection, mapping of static and dynamic techniques used for hybrid techniques etc. So, the research in this field is infancy and its necessity to cover these undiscovered topics in traditional surveys.

4.4. Techniques used for detection of malicious applications in android mobiles/devices

As per the recent research a number of researchers have developed their techniques for the detection of malware/malicious application in android. Malware detection techniques are broadly divided into three following categories [46,47]:

- Static
- Dynamic
- hybrid

4.4.1. Static techniques

When analytic methods are used to study any program where no code execution is involved, such methods and procedures fall under the category of Static Analysis. There are several techniques to take a program through this static analysis, which involve following approaches.

Architecture

Static analysis is the process which is performed on the application package. Fig. 7 shows the architecture of static analysis. It includes the analysis of .apk file of the concerned application. First of all, application is downloaded from the play store or the application store. Application folder (.apk) include various types of files, but analysis process focuses on only two files one is manifest file (AndroidManifest.xml) and other is source code file (class.dex) [48,49]. Android manifest file contains information about the permissions, hardware components, software components and other details about the package. On the other hand, source code file contains all programming code which includes API information, intents, system events and other more factors. One more interesting factor is also included in this system call

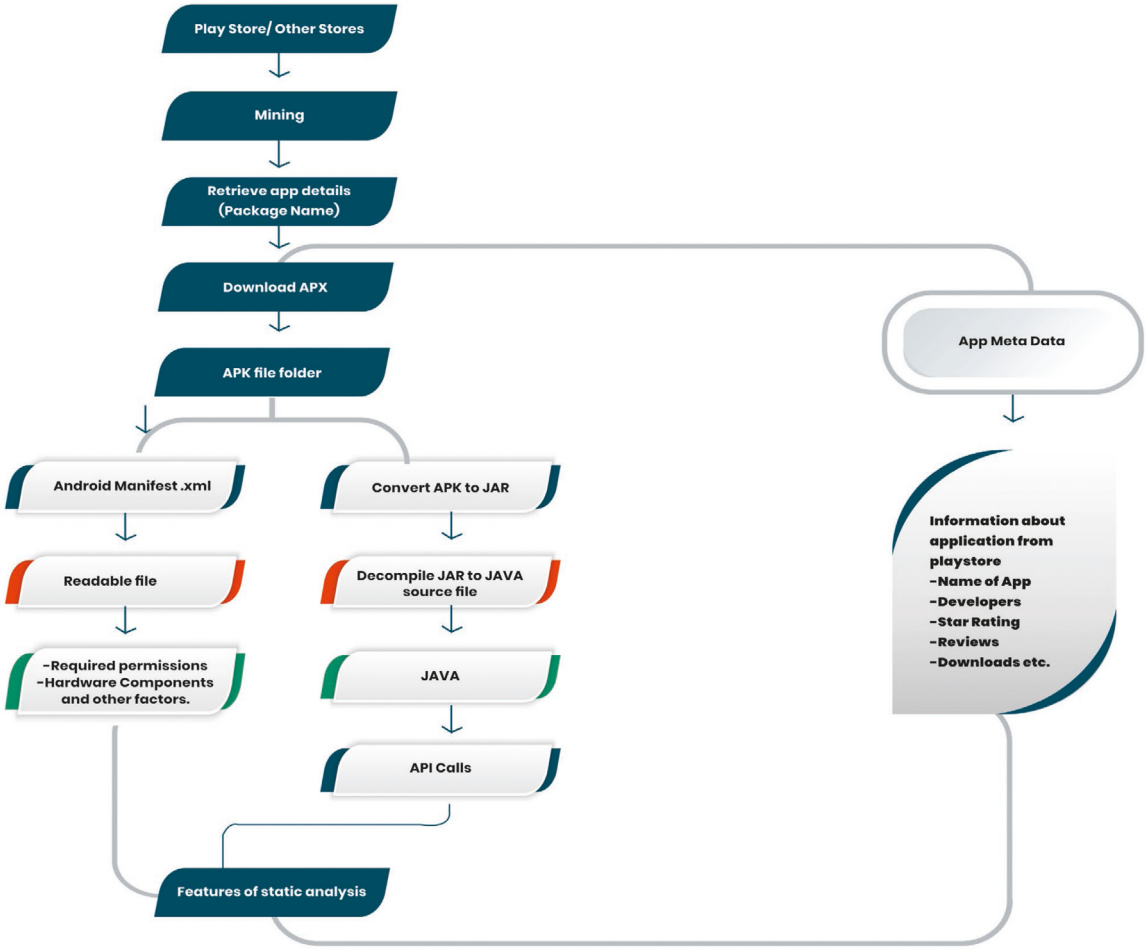

Fig. 7. Process of static analysis.

meta-data information [50–52]. This is the information that can be extracted from the Google play store. It includes the name of application, developer's information number of downloads, star rating, review, screenshots etc.

4.4.1.1. Various features used for static analysis.

1. *Permissions:* In permissions, android applications have a permission terminology which requires any android app to ask for respective specific permissions for the execution of code section and particularly run its functionality. For example: READ and WRITE permissions to access EXTERNAL STORAGE, permission to INTERNET ACCESS etc. These permissions have restricted any app from performing any background activity without the user consent and approval, and to run without permission [4,17]. For example, SEND_SMS, if an application wants to send SMS then android system checks whether that application have access to this permission then only, they can send SMS access to this permission otherwise that application is unable to send SMS. Same thing is applicable for other tasks. AndroidManifest.xml is a file in .APK folder of application which contains all the information about the permissions requested by the application.
2. *Intents:* It is a simple message object which is used to make communication between android components activities, content providers, broadcast receivers and services. To perform any operation which needs system intervention like DIALING a number, CAMERA capturing images or videos, etc. or to perform any other appropriate actions, Intents are used and therefore can be useful to analyze and understand malicious actions of any malware app [4,53].
3. *Hardware components:* Many hardware components are embedded into the device's main board to provide useful functionalities like GPS, microphone, gyroscope, etc. But again, they can be used by malware apps that can compromise user's personal information. So, studying and analyzing these activities will help to identify malware apps [4,10].
4. *Android application metadata information:* Android application's information on play store can be used to analysis the application. There are various factors which can be used for this purpose e.g. Application package name, developer information, star rating, number of people rated the stars, reviews, size of application, number of screenshots, promotional videos etc. These factors may help to categorize the application [4,10].
5. *API calls:* Android .apk file contains a variety of files, it contains one file named .dex file or classes.dex file, this file contains the information about source code of classes. But this .dex file is not human readable. So, it needs some conversion to become human readable. It can be converted to java file so that API calls data can be easily analyzed from the file. This API calls details can be used for static analysis of the android application [4,10,18].
6. *Graphs or structure based:* As the source code of file can be extracted from classes.dex file, so this source code can be used to create dependency graphs of statements that can help in static analysis of the android applications. One

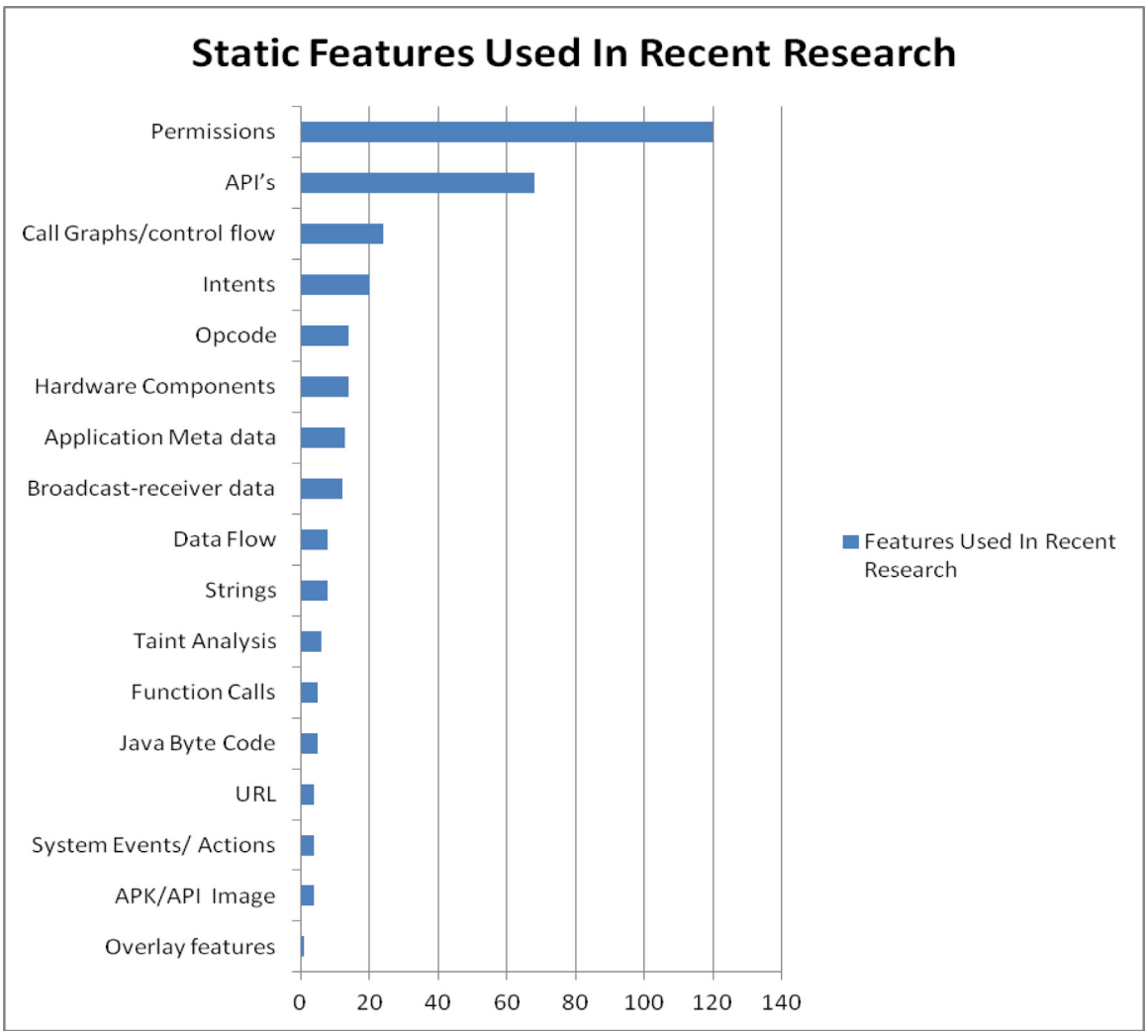

Fig. 8. Static features used in recent research.

malicious code pattern can be converted into graphs by assuming the statements as nodes and control flow as the path of nodes. So, these kinds of graph patterns can be used to detect same kind of malicious application content in other applications. In this case data flow graphs and graphs related to function call are created to design a pattern for malicious application detection [54–56].

7. *Taint analysis:* Taint analysis is the one of the special case of data flow analysis used for detection. It is the method to detect flow of sensitive data from the source and sink [57, 58]. Both the source and sick are analyzed for this purpose. Source means the data that is asked by the function from the system like function that acquire device ID and sink is the point which is used to send sensitive information from the system e.g. sending SMS to premium number.
8. *Strings/Opcode/URL:* Strings can be used as a parameter for the static analysis. In this case there are various methods by using which researchers can search for a particular string in the code. This method is applied on the java code. Any particular string can be searched from the code in benign and malicious application for the classification purpose. It may be any keyword, opcode or an URL used for specific purpose. Wang et al. [59] performed a test analysis on the network traffic for the analysis process to classify the applications. According to Parker et al. [60], disassemble the application using blacksamli dissembler and convert it to dalvik bytecode and then other components are removed except opcodes and opcode is used for analysis purpose.
9. *Java byte code/APK–API image:* Java byte code is used for investigation process in the form of images. As per study, in java source code objects and classes are analyzed in respect to the permissions mentioned in the manifest file [61,62]. Hang [63] and Kumar [64] used an image file as a static analysis feature. Application file (.APK) file converted to image using its byte code and then images are analyzed for classification using machine learning approaches [65].
10. *System events/Services-broadcast-receivers:* System services are the events that are used to make interaction of application with the system, that can be obtained from the API's [66] and these can be used as a feature for the analysis process. Services can be categorized as location, search, connectivity, telephony etc. [66]. Wei et al. [67] used the combination of system functions as a parameter in their study. Android application can interact with system and other applications with the help of broadcast receivers [68]. The system events are broadcasted in these messages and concerned application listens to specific message and applies their action such as (BOOT_COMPLETED, SMS_RECEIVED, CONNECTIVITY_CHANGE, etc...).

Table 4 consists of various features used for static analysis of android malicious application detection. it includes the set of

features along with the citations of research papers that have used those features. One paper may be mentioned in more than one feature, because multiple features or combination of one or more feature can also be used for the analysis (see Fig. 8).

4.4.1.2. Discussion. As per the recent research it has been observed that there are various features which are used for static analysis of the applications for the detection of malicious application. In Table 4, total 17 features are mentioned, these are the features which are used by the different researchers for the development of static analysis techniques. The frequency of permissions used by malicious and benign applications is used for the detection process [84]. It may be used solely or as a combination with other one or more features. Permissions are extracted from the manifest file of the .apk of any application. The main reason behind permissions is that it contains all the information about the permissions that are required to run the application on the system/mobile. So, researchers can analyze the behavior or intension of the application from the permissions requested. Wang et al. [103] have used unique requested permissions (URP) and unique used permissions (UUP) are compared win case of normal and malware application. Second most used feature is API calls. These API calls can be extracted from the class.dex file of the application. Process is described in Fig. 7, API calls used by the application can be analyzed for the analysis process. Here, suspicious and dangerous API call are detected and on the bases of these applications are classified. Droidlogger [297] used API call blocks, means set of APIs used for specific purpose and task, means a compete method used, it outperforms that single API call analysis. Combination of these two features (permissions and API) is also used many times [104,108,144]. This is only because both of these cover the main part of application such as manifest file and class.dex file. Other features meta-data information, string search, intents, call graphs, hardware components are also used but not as much permissions and API are used. Yan et al. [250] used overlays for the investigation process, various features such as screen width, orientation, type, flag and format are included and depicted that flag, type and format are strongly linked with the malice of an application because 84% of the malware application uses TYPE_SYSTEM_ERROR and more than two-third uses FLAG_FULLSCREEN or FLAG_LAYOUT_IN_SCREEN. As per recent research from 2016–2018 [64,192–194], one new method "APK image" is found for malicious application detection, in this process bytecode of APK is converted into image and then that image is used for the malicious pattern. Wu et al. [192] mentioned that this process reduces time and resource consumption because it avoids recompilation of .apk and no need of different file structures to process this.

4.4.2. Dynamic techniques

When analytic methods are used to study any program where code execution is involved with observations and parallel results, such methods and procedures fall under the category of dynamic Analysis. There are main two approaches are used for dynamic Analysis: in the Box Analysis and out the box analysis. These techniques need an isolated environment to run the app and observe the information in real-time [4,126,318]. Dynamic analysis is complex than Static analysis and need several resources and skill sets to observe and come to conclusion [319–321].

Architecture

In dynamic analysis, the behavior of application is analyzed for the malware detection. There are two ways where an application can be analyzed: In the box (on the device) and out the box (off the device) [4,298,322]. Then application is executed and various factors are analyzed for the investigation such as system calls, flow of information, usage of hardware components etc. are the main features that can be processed for the behavior analysis of the application at run time (see Fig. 9).

4.4.2.1. Various features used for dynamic analysis.

1. *System calls:* As per the architecture of the android when user tries to perform any activity then android runtime system executes the system call in kernel to perform that task. So, some of the events that remain undetected in static analysis that can be detected by using system call analysis by collecting system call logs. When events get executed in the emulator then log records are prepared of system calls and those system calls can be used to analyze the activities and find out the malicious activity.
2. *Information flow (Taint analysis and network traffic):* Information flow is the method which is used to track the flow of information from a device through its network. In case of taint analysis both the source and sink are observed for the purpose of analysis [101,323]. On the other side, in network traffic analysis data packets can be analyzed at various parameters such as packet size, type, frequency of in and out etc. Information propagates from source to destination and that flow of information is used for analysis of malicious activity. SCDFLOW [324] (Selecting Critical Data Flows) this system works to analyze critical data flow by checking the frequency of flow of data. As per study [325] most network traffic is shared on network is benign, only small part is malicious and that part can be detected with the help of analysis of flow of data from both malicious and benign application.
3. *Function call monitoring:* Function calls can be dynamically monitored using API framework, this method can be used to track the function calls and parameters used, this can be used to reconstruct the function call sequence that may be helpful for the analysis process [4].
4. *Dependency graphs:* Control flow graphs are created for specific sensitive/suspicious piece of code then that graph pattern is used for analyzing applications for classification. This method is mainly used where the malware suspicious code is known and we want to categorize the families of malwares [19].
5. *Hardware component analysis:* In this system running hardware components are used for analysis purpose. For example: battery data is used [3,4]: battery status, temperature, charger status, battery level. System parameters are also used for this purpose like utilization of CPU, CPU memory, App memory etc. Other hardware components GPS, Wi-Fi hardware etc.
6. *Used API and permissions:* There are some factors which can be analyzed in the both static and dynamic analysis processes like permissions and API's. During the execution of the application permissions used at the time of exaction also very important factor [302,314]. In contrast of it API's can also be analyzed for better results in dynamic analysis [278,326].
7. *Communication with the exterior:* Mobile phones are used for various communication purposes. There are some factors likes SMSs, phone calls, URLs, WiFi, GPS and Bluetooth which makes the connection of a system with other external entity [177,308]. So, these all communication factors can be observed as a single or as a combination of more than factor for the classification purpose.
8. *Code injection:* Special codes are injected into source code for the detection of malicious behavior. Fan et al. [310] developed an approach DroidInject, a code is injected in source code then that code executes at run time and track the progress of targeted API and creates log for the analysis process. Target application is repackaged with injected code and then application is monitored for the contacts used by the application for premium SMS [311].

Table 4
Static features/techniques used.

Feature used	Code	Count	Citations
Permissions	S1	120	[10,24,56,61,62,66–184].
Intents	S2	20	[10,24,66,74,80,81,83,97,105,112,116,129,131,137,184–189].
Hardware components	S3	14	[10,61,66,80,81,112,116,130,131,157,165,174,184,190,191].
APK/API image	S4	4	[64,192–194]
Call graphs/control flow	S5	24	[54,85,86,125,158,195–213].
API's	S6	68	[10,56,66–69,71,72,76,77,79–82,87,89,92,98,99,101,104,106,108, 109,114,116,117,121,122,124,126,127,129,136,144,147,149,152,153, 156,158,160,166–168,176,177,180,182,184,212,214–231].
Application meta data (names, certificate name, rating, category of app, reviews, screenshots)	S7	13	[80,81,122,139,157,161,171,177,181,232–235].
Taint analysis	S8	6	[58,172,185,236–238].
Opcode	S9	14	[60,74,80,81,132,168,184,235,239–244],
Strings	S10	8	[59,80,81,83,138,160,245,246].
Java byte code	S11	6	[61,62,95,168,247,248].
System events/Actions	S12	4	[66,67,82,249].
Overlay features	S13	1	[250]
Data flow	S14	8	[72,197,199,205,209,215,246,251].
Function calls	S15	5	[88,121,214,231,252].
Broadcast-receiver data	S16	11	[10,68,97,112,116,132,141,145,150,153,174].
URL	S17	4	[99,116,191,253].

Table 5
Dynamic features/techniques used.

Feature used	Code	Count	Citations
System calls	D1	43	[10,63,87,93,97,122,133,152,162,170,177,182,189,215,229,233,235,254–279].
Network traffic	D2	27	[3,59,71,120,137,155,175,249,276,279–296].
Application programming Interface	D3	8	[129,164,283,297–300]
Dynamic Code loading (DCL)	D4	7	[120,180,229,254,279,299,301].
Hardware/Resource consumption (CPU, memory, threads, battery, location)	D5	12	[3,114,137,190,274,283,292,302–306].
Communication with the exterior (SMSs, phone calls, URLs, WiFi, GPS and Bluetooth)	D6	12	[114,120,127,177,190,233,235,279,283,302,307,308],
Taint analysis	D7	6	[101,191,236,279,299,309]
Code injection	D8	3	[310–312].
Temporal sequences of API calls invoked by apps	D9	1	[313]
Permissions at runtime	D10	2	[302,314]
Overlay features	D11	1	[250].
Runtime UI	D12	3	[315–317].

9. *Runtime UI:* Graphical user interface can be used to detect the repackaged applications [315–317]. Yue et al. [315] developed approach RepDroid, used layout structure to monitor repackaged applications. Clone of application is detected using UI information without executing the entire application.
10. *Dynamic Code loading (DCL):* When application executes it can also load code at the time execution, that code is known as dynamic loading of code. Sometimes applications load malicious code using DCL. This feature can be used for the monitoring process [180,254,301]. Various approaches have been developed approaches for the detection of dynamic code loading at run time of application [180,327, 328].

4.4.2.2. Discussion. Table 5 gives information about a number of features used to analyze the applications at time of execution. There are 12 features identified from various research articles related to this field, out of which system call is used in 43 approaches. Amamra et al. [256] uses system call patterns for the investigation process, special approach that they have performed is system calls are filtered for abstraction and better results, it replaces system call with aliases. Vinod et al. [258] developed an approach to check the malicious application behavior from the frequency of occurrence of system calls. System calls are analyzed in two ways: namely absolute difference of weighted system calls (ADWSC) and ranked system calls using large population test (RSLPT). Network traffic is used as a parameter for detection of malicious behavior, in this case various factors are analyzed such as average packet size, number of packets sent/received, time interval between packets, ratio of incoming and outgoing packets etc. [282,286]. Wang et al. [59] considered HTPP flow as a document and investigated the HTTP flow request using natural language processing for string analysis. Other features hardware resources such as usage of CPU, memory, battery etc also used as feature for the behavior analysis but these are not very much effective as compared to system call and network flow. Basically, permissions and API's are used for static analysis but in some case, these are analyzed at the time of execution. Zhu [236] and Brown [309] used source and sink of information flow for the behavioral analysis, this process is also known as taint analysis. At last, it has been observed that system calls and network traffic are the features which are preferred by most of the researchers in their dynamic analysis (see Fig. 10).

4.4.3. Hybrid techniques

This type of analysis is the combination of the practices used in both static and dynamic analysis. Thus, it has the capability of performing more advanced analysis in real-time with a firm conclusion and result to debate on the characteristic of app to be categorized as a malware app or not. Hybrid technique involves the use of following approaches like analyzing Network Traffic, studying API's (Application Program Interfaces), System calls, dependency graphs, features, functioning call monitoring, Information flow, Inter-Process Communication analysis, Hardware analysis, reading Application meta data, etc [72]. Nowadays, any of the above techniques cannot be appropriately used alone for identifying any malware app. They need to be merged in a way to study every aspect of the app and declare it a malware app [77,

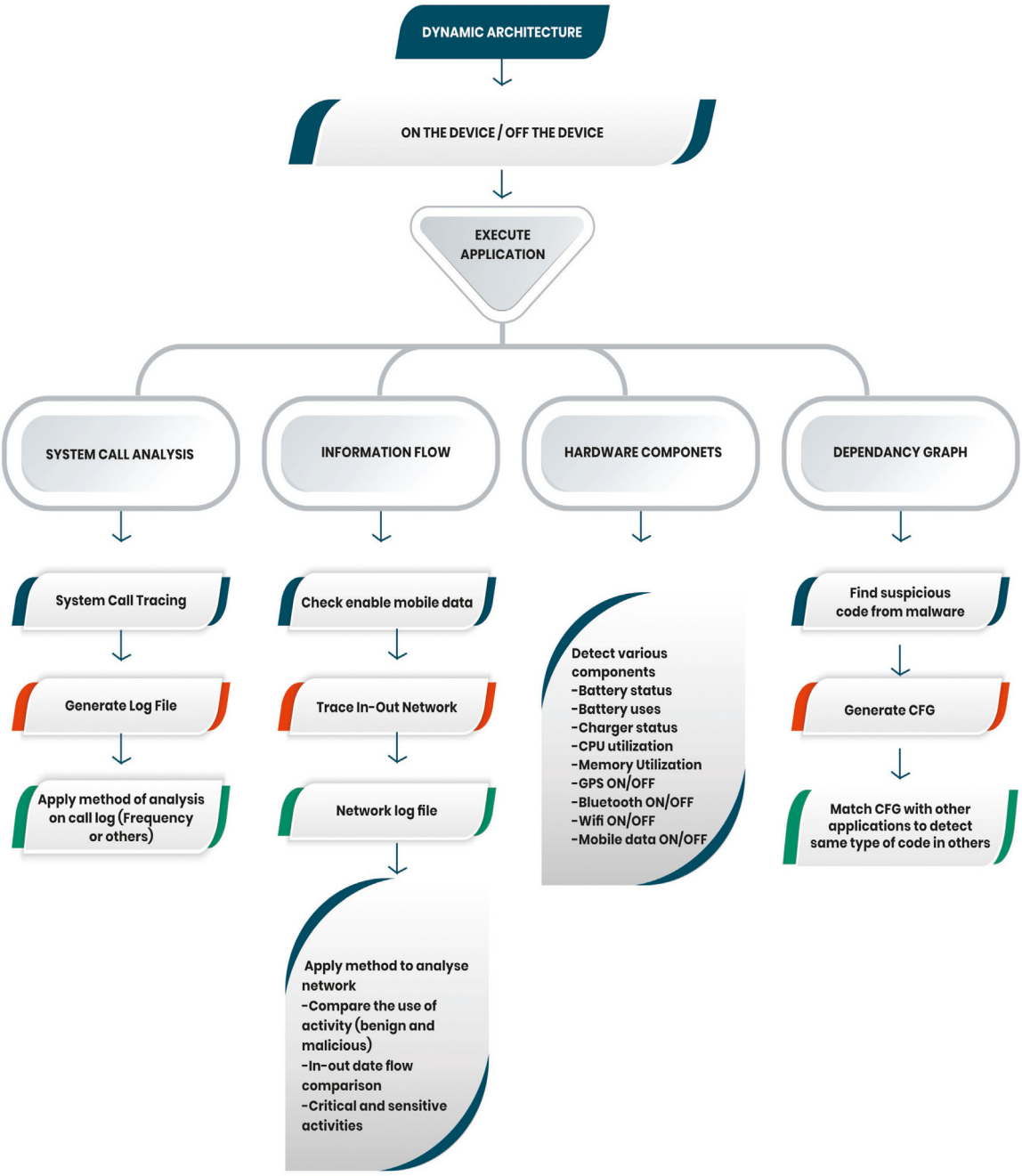

Fig. 9. Process of dynamic analysis.

329]. Sometimes, static systems are unable to detect obfuscation attacks, where attackers change their properties while they are working or executing. That kind of application can be detected in dynamic analysis. But static technique is easy to implement without executing and affecting the actual data of system. So, both the approaches have their own pros and cones [330,331]. This is more beneficial to utilize the properties of both the approaches.

4.4.3.1. Hybrid static–dynamic mapping. See Table 6 and Fig. 11.

4.4.3.2. Discussion. Table 6 describes the mapping of various static and dynamic techniques used to create hybrid technique. It shows different combinations of the static and dynamic techniques. As we can see in the table H2 is used by most of the researchers for developing hybrid technique. It involves permissions (S1) and API's (S6) in static part and system calls (D2) are used in dynamic. Fig. 11 shows the mapping of all static and dynamic features to consist the hybrid technique. In this it is found that S1 and S6 in static and D1 and D6 (Communication with exteriors) in dynamic are the most implemented methods for the combined or mixed technique.

4.5. Repackaging applications and various methods used to detect these types of applications

In several cases, popular applications available on official Google play store can be added with additional malicious code and published on third party application store with same metadata information [187,235,236,321]. These types of apps are known as repackaged application. Modifications can be applied by using various ways like: replacing any API with other, inserting malicious code in existing source code, redirecting the

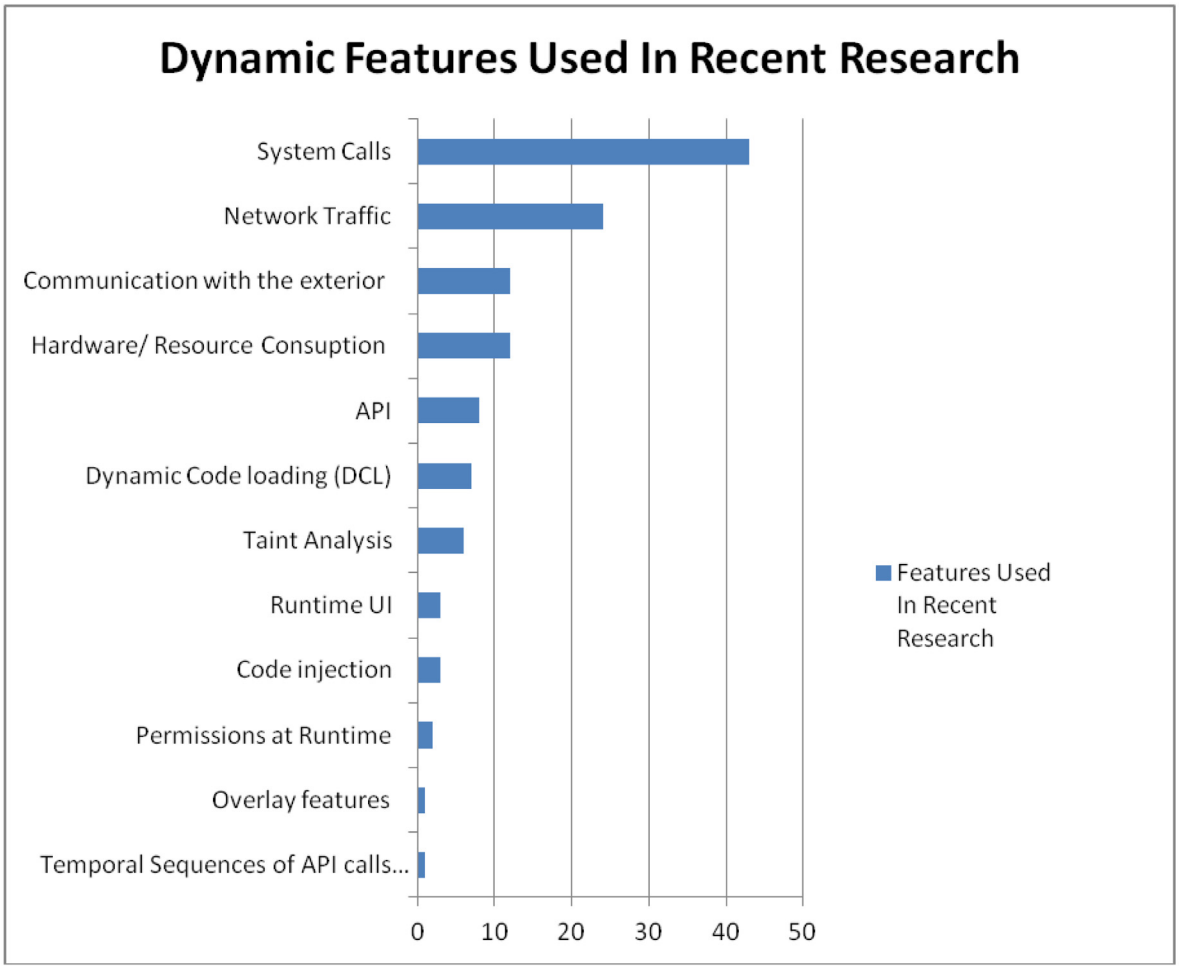

Fig. 10. Dynamic features used in recent research.

Table 6
Hybrid static–dynamic mapping.

Static features	Dynamic features	Code	Count	Citations
S1	D1	H1	3	[133,170,332]
S1+S6	D1	H2	5	[87,106,152,273,333].
S3	D5+D6	H3	1	[190]
S1+S6	D2	H4	1	[71]
S1+S6+S16	D1	H5	1	[97]
S1+S6	D7	H6	2	[101,122]
S7	D1+D6	H7	1	[233]
S1+S6	D6	H8	1	[127]
S1+S2+S6	D3	H9	1	[129]
S8	D7	H10	1	[236]
S7+S9	D1+D6	H11	1	[235]
S1+S6+S7	D1+D6	H12	1	[177]
S1+S3+S6	D1	H13	1	[10]
S1	D2	H14	2	[155,175,334]
S1	D2+D5	H15	1	[137]
S6	D1+D4	H16	1	[229]
S1	D3	H17	1	[164]
S1+S6	D5+D6	H18	1	[114]
S3	D6	H19	1	[191]

app towards some revenue advertisement etc. [335–338]. These applications can be detected by applying some methods. Some of the methods are described as following:

- *HTTP traffic monitoring*- As per Wu et al. [339], monitoring HTTP traffic of applications can be useful to detect this, here they divided traffic in two modules: Primary module traffic (like original application) and non-primary module traffic. After that they analyzed the implanted codes malicious code and advertisement code used to collect revenue [340].
- *AndrODet*- Mirzaei et al. [341] developed an approach for repackaging in android applications, with this approach obfuscation can be detected in three ways: applications name identifier renaming, string encryption used and control flow obfuscation in application.

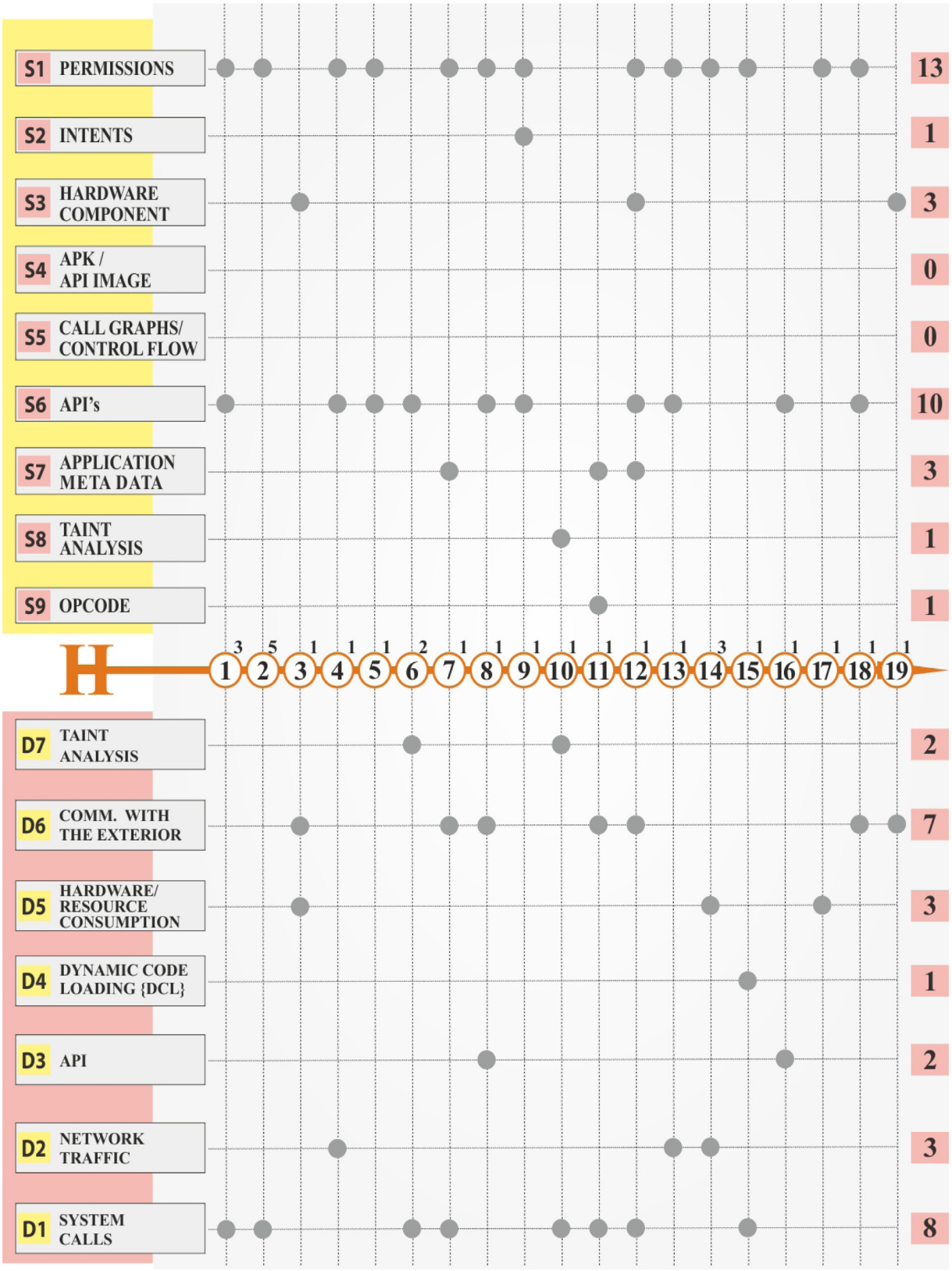

Fig. 11. Hybrid — Mapping of static–dynamic features.

- *CloneSpot*: Martin et al. [342] developed a novel approach based on meta-data publicly of applications available at Google Play to detect repackaged application.
- *App icon and name*: Lyu [343] and Gurulian [344] developed an approach based on hashing to detect icon/logo and name of repackaged application by comparing it with the original one.

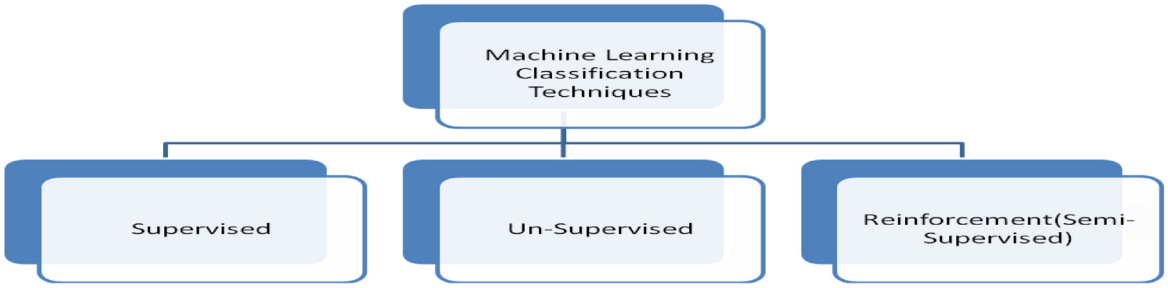

Fig. 12. Machine learning classification techniques.

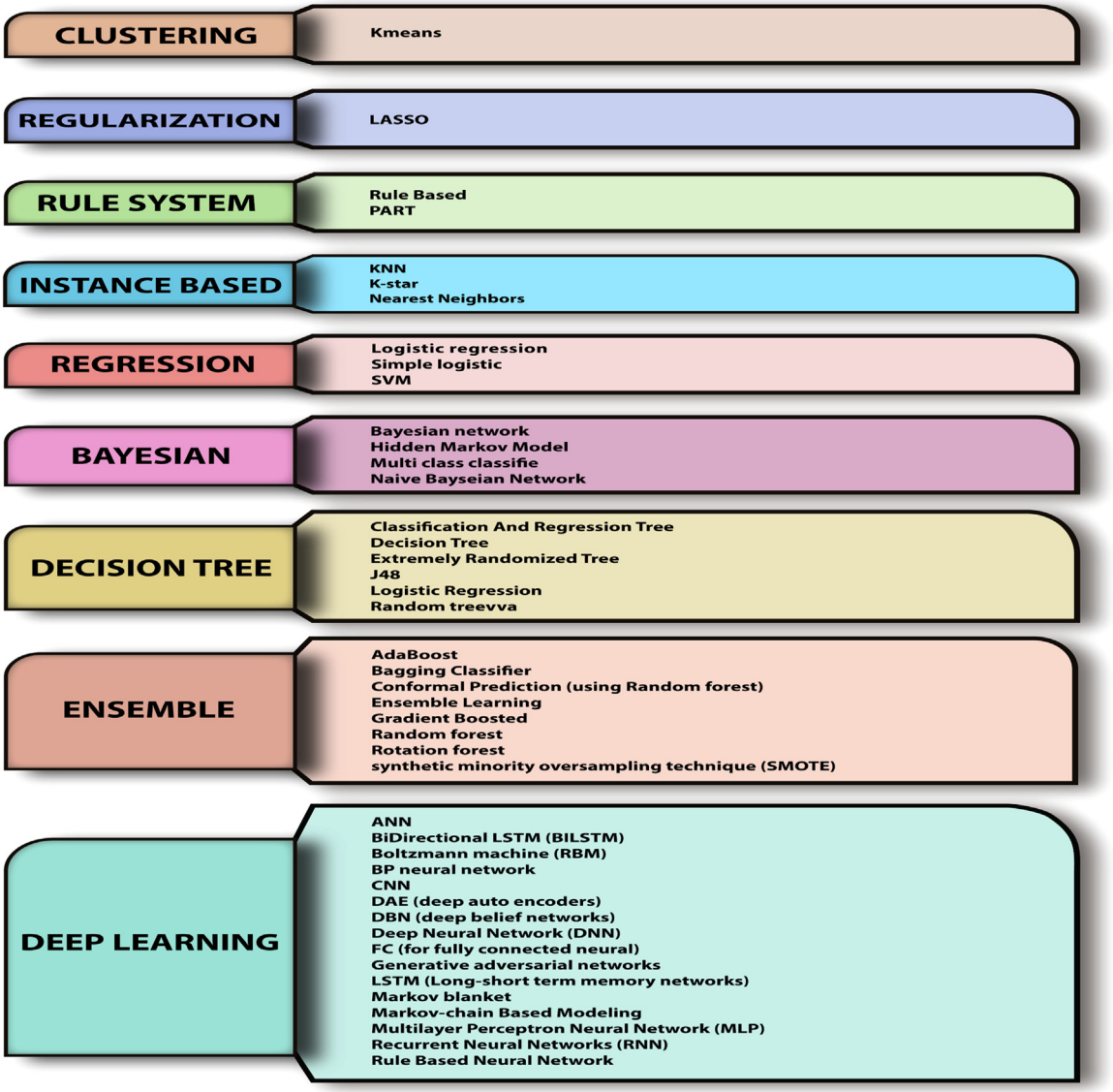

Fig. 13. Category wise grouping of classification algorithms used in recent research.

- *Opcode*: Shahriar et al. [336] detected repackaging of application using opcode used for malicious activities from smali code of an application. Kullback–Leibler Distance (KLD) approach is used for the classification purpose.
- *Application's .apk data*: Detection of alteration of an application by using various feature from the .apk of the file such as package name, version of application, application developer, class.dex, manifest file, resource file etc. These features are also investigated whenever system updates the application [336].

4.6. Gaps in dynamic analysis methods

- *Input generation method:* In dynamic analysis commonly system calls are used for investigation purpose that talks about the calls which are used by the application to the kernel.

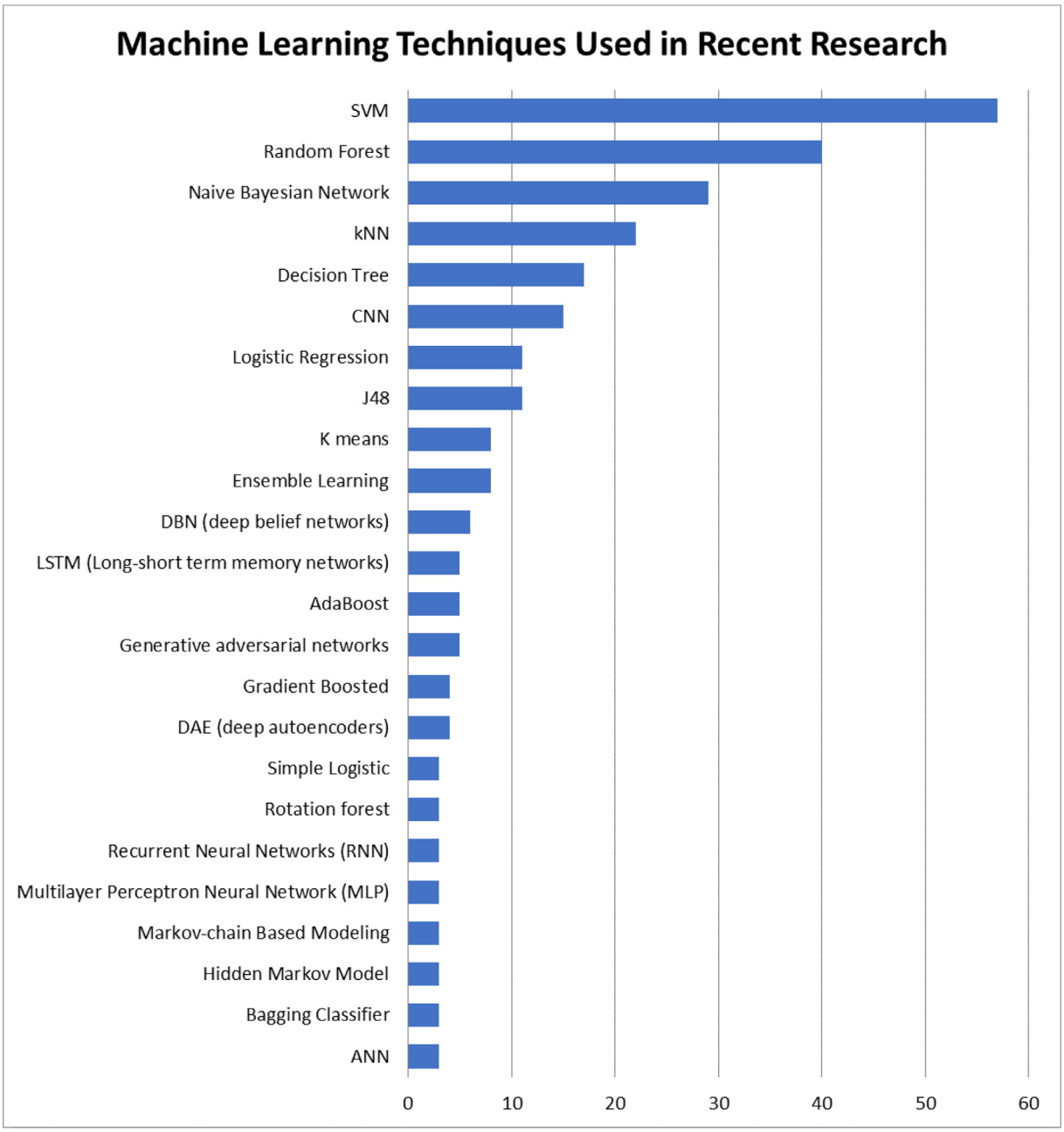

Fig. 14. Machine learning techniques used in recent research.

Most of the systems are using random based input generation method that means it does not cover all the activities of the application. There may be some possibility of malicious activity in the region which is missed by the random system. So, this method is unable to detect all the activities [319].

- *Use of emulator:* Investigation is harmful on real mobile device, some researchers started using emulators for the analysis purpose instead of real device. But these days, malware creators are very intelligent, some malicious applications hide their behavior (act as benign application) whenever they detect that application is running on emulator [170].

4.7. Machine learning techniques used for classification in malicious application detection process

Machine learning is the technique which is used in for the classification of application in the process of malicious application detection. Machine learning techniques learn the behavior of applications (both benign and malicious) from the defined feature set then classify the newly installed or added applications accordingly [184,345,346]. Machine learning process includes two main modules: learning and testing [347]. In learning module machine learning algorithms are trained according to the behavior of legitimate and malicious applications and in the second module algorithms are tested with the valid dataset for their working accuracy [190,348,349]. These techniques are broadly

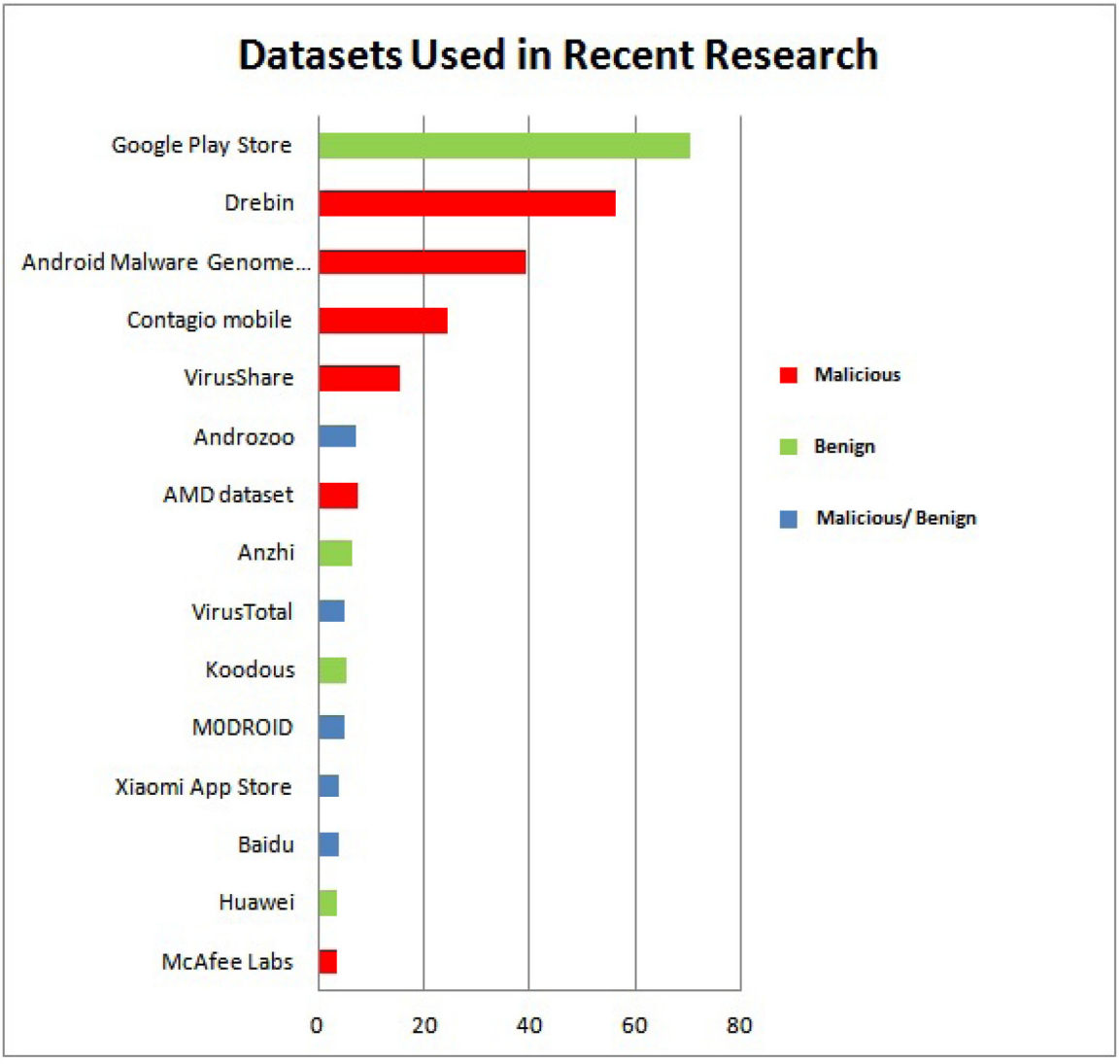

Fig. 15. Various dataset used in recent research related to malicious application detection in android.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

classified into three techniques: Supervised, semi-supervised and unsupervised [350,351].

There are a number of machine learning approaches which are used along with malware detection systems in mobiles. Fig. 12 shows a range of machine learning algorithm used in existing research papers. In this these algorithms are categorized according to their class. As per the Table 7, 44 different machine learning approaches are applied in the recent research in regard of malicious and benign application classification (see Fig. 13).

4.7.1. Discussion

Machine learning is the techniques which can be used to perform automatic analysis process by getting trained from the defined features and it is also beneficial to detect zero-day attacks [375]. Total 44 machine leaning algorithms has been found after analyzing recent research papers. Out of these five algorithms are more popular. According to chart in Fig. 14, SVM, Random forest, Naive Bayesian Network, kNN and Decision tree are used by the most of researchers. [83,120] mentioned in their research that SVM performs well as compared to Random tree and other machine learning techniques because it has high rate of accuracy. Rana et al. [347] compared the performance of various machine learning algorithms in case of malicious and legitimate application classification and resulted that SVM, Random forest and kNN performed well as compared to others with accuracy more than 90%. Kumar et al. [219] used machine learning and compared the results extracted from a variety of algorithms and found that SVM is performing better with accuracy more that 95%. A new parallel machine learning approach, they have used different classifiers at the same time such as rule based, function based, tree based and probabilistic are used in parallel and their results are combined and gained high accuracy [160]. But this approach is not much used after this research. Ma et al. [376] developed a new approach in which 2 layers of machine learning is used, in this they have used SVM in first level and then, NB-SVM, RBF-SVM and 2 layer linear-SVM in second level and found good results from RBF-SVM. As per the figure, classification algorithms are categorized according to their class of working. It classifies it into 9 different classes and various participating algorithms are organized accordingly. It shows that deep learning has large number of algorithms, 16 methods are mentioned as per recent research papers. CNN is the popular one as compared to other deep learning algorithms. But as comparison of others, it comes on 6th number because only 16 papers have used CNN

Table 7
Machine learning techniques used.

Sr. No.	Machine learning method used	Code	Count	Classification method	Category	Citations
1	AdaBoost	M1	5	Supervised	Ensemble	[83,181,257,258,268].
2	ANN	M2	3	Reinforcement (Semi-Supervised)	Deep learning	[147,269,313].
3	Bagging classifier	M3	3	Supervised	Ensemble	[145,181,332].
4	Bayesian network	M4	2	Supervised	Bayesian	[66,352].
5	BiDirectional LSTM (BiLSTM)	M5	1	Reinforcement (Semi-Supervised)	Deep learning	[241].
6	Boltzmann machines (RBM)	M6	1	Reinforcement (Semi-Supervised)	Deep learning	[101]
7	BP neural network	M7	1	Supervised	Deep learning	[87]
8	Classification and Regression Tree	M8	1	Supervised	Decision Tree	[184]
9	CNN	M9	16	Supervised	Deep learning	[94,99,100,105,135,194,217,225,239,241,242, 261,353–356],
10	Conformal prediction (using Random forest)	M10	1	Supervised	Ensemble	[357]
11	DAE (deep autoencoders)	M11	4	Un-Supervised	Deep learning	[98,216,241,354].
12	DBN (deep belief networks)	M12	6	Supervised	Deep learning	[101,136,166,222,238,241].
13	Decision tree	M13	17	Supervised	Decision Tree	[56,64,67,80,116,121,129,163,174,181,223,268, 286,302,314,347,358],
14	Deep Neural Network (DNN)	M14	4	Supervised	Deep learning	[51,165,323,324]
15	Ensemble learning	M15	8	Supervised	Ensemble	[24,79,108,131,157,184,279,359].
16	Extremely randomized tree	M16	2	Supervised	Decision Tree	[116,347].
17	FC (for fully connected neural)	M17	1	Supervised	Deep learning	[241].
18	Generative adversarial networks (GAN)	M18	5	Un-Supervised	Deep learning	[51,257,258,360,361].
19	Gradient boosted	M19	4	Supervised	Ensemble	[116,129,268,347].
20	Hidden Markov model	M20	3	Supervised	Bayesian	[240,263,300].
21	Hybrid neuro-fuzzy classifier	M21	1	Reinforcement (Semi-Supervised)	AI	[119].
22	J48	M22	11	Supervised	Decision Tree	[66,67,72,88,91,120,145,169,314,358,362].
23	K means	M23	8	Un-Supervised	Clustering	[70,71,115,188,221,253,293,347].
24	kNN	M24	23	Supervised	Instance Based	[3,12,56,64,66,87,88,97,121,124,140,145,184, 192,195,202,210,223,230,253,270,341,347].
25	k-star	M25	1	semi-supervised	Instance based	[314]
26	LASSO (least absolute shrinkage and selection operator)	M26	1	Supervised	Regularization	[265]
27	Logistic regression	M27	11	Supervised	Decision Tree	[3,56,80,91,120,174,195,210,307,347,358],
28	LSTM (Long-short term memory networks)	M28	5	Reinforcement (Semi-Supervised)	Deep learning	[241,267,270,363,364].
29	Markov blanket	M29	1	Supervised	Deep learning	[132]
30	Markov-chain based modeling	M30	3	Reinforcement (Semi-Supervised)	Deep learning	[201,269,277].
31	Multi class classifier	M31	1	Supervised	Bayesian	[173,341].
32	Multilayer Perception Neural Network (MLP)	M32	3	Supervised	Deep learning	[3,260,294].
33	Naive Bayesian network	M33	29	Supervised	Bayesian	[3,56,66,67,72,87,91,106,111,120,124,129,140, 145,148,161,167,174,181,184,203,223,314,326, 333,347,358,362],
34	Nearest neighbors	M34	1	Supervised	Instance Based	[181]
35	PART	M35	1	Supervised	Rule system	[88]
36	Random forest	M36	40	Supervised	Ensemble	[3,64,66,80,82,89,91,112,114,116,120–122,129,137,145,147,151,158,163,164,181,184, 198,207,210,223,230,257,258,265,268,314,332, 333,341,347,362,365],
37	Random tree	M37	2	Supervised	Decision Tree	[66,91].
38	Recurrent Neural Networks (RNN)	M38	3	Reinforcement (Semi-Supervised)	Deep learning	[105,241,366].
39	Rotation forest	M39	3	Supervised	Ensemble	[112,257,258].

(*continued on next page*)

Table 7 (*continued*).

Sr. No.	Machine learning method used	Code	Count	Classification method	Category	Citations
40	Rule based	M40	1	Supervised	Rule system	[85].
41	Rule based neural network	M41	1	Supervised	Deep learning	[367].
42	Simple logistic	M42	3	Supervised	Regression	[314,358,362].
43	SVM	M43	59	Supervised	Regression	[3,6,10,12,56,59,68,71,72,80,83,87,88,91,97, 112,114,120,121,139,145,147,149,158,163,164, 174,180,182,184,200,203,204,210,226,229,230, 233,244,246,264,265,268,277,278,295,307,326, 333,341,347,358,362,365,368–374].
44	Synthetic minority oversampling technique (SMOTE)	M44	1	Supervised	Ensemble	[295]

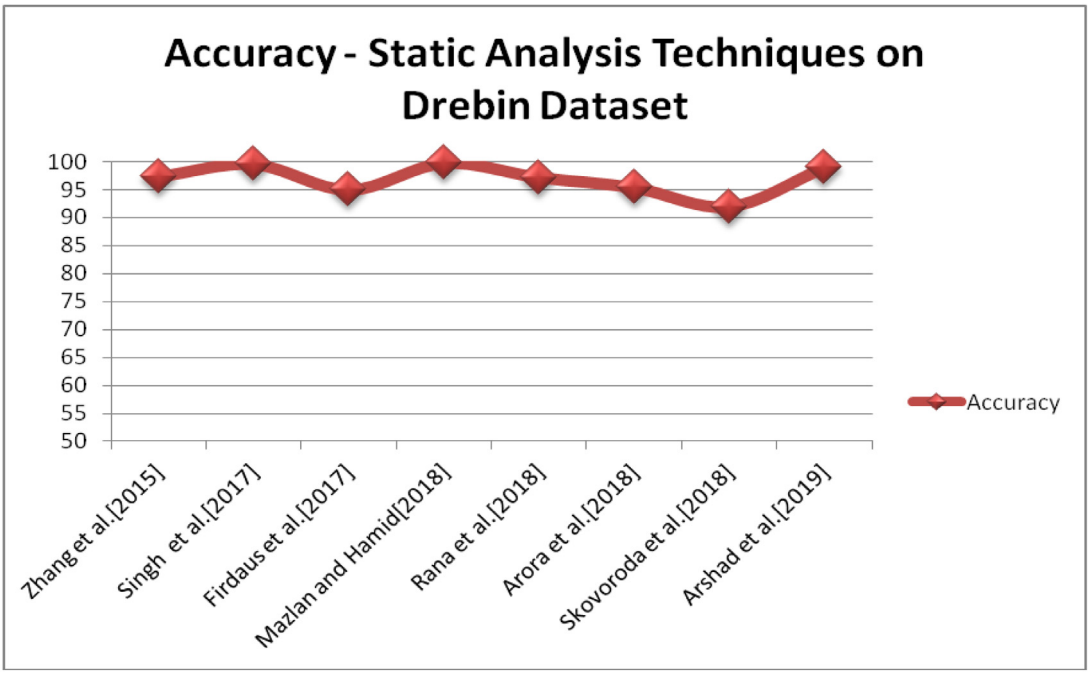

Fig. 16. Accuracy — Static analysis techniques on Drebin dataset.

for classification of applications. Generative adversarial networks (GAN) is also discussed in latest research studies [51,258,376] used to evaluate the performance of malware detector. Adversarial samples are generated to mislead the detection process that is used to find out the vulnerabilities in the machine learning classification system [258,360,361].

4.8. Various dataset used in recent research related to malicious application detection in android

According to previous research there are a number of datasets available for the experiments. These datasets contain information about specifically benign, malicious or both applications. As per the analysis of previous research papers about 66 different datasets are used. Table 8 gives information about the datasets, it includes the type of dataset, and number of papers used that dataset in their experiments and the citation of papers. It includes dataset from various projects, websites, antivirus companies, third party app stores etc.

4.8.1. Discussion

Fig. 15 shows a bar chart of various datasets used in recent studies and mentioned in descending order. The dataset used in less than 3 studies is excluded in this chart but mentioned in Table 8, Red bar in chart represents malicious, green represents benign and blue represents both malicious and benign datasets. In case of malicious datasets Drebin and Android Malware Genome project used by the most of studies and on the other hand for benign application Google playstore is used generally. Canadian Institute of Cybersecurity (CIC) is also providing latest malware datasets of android malwares [154,176]. CIC has a vast repository of malware datasets for the research community available at their website viz. ISCX Android Botnet dataset 2015, ISCX Android Validation dataset 2014, Android Adware and General Malware Dataset (CIC-AAGM2017), Android Malware Dataset (CIC-AndMal2017), Investigation of the Android Malware (CIC-InvesAndMal2019), CICMalDroid 2020, CCCS-CIC-AndMal2020. Kadir et al. [384] prepared dataset named ISCX Android Botnet Dataset which includes 1929 malware samples from 14 botnet families and these datasets are relatively used for machine learning based detection systems.

4.9. Comparison of various techniques used in recent research for malware detection analysis on "Drebin" dataset

As per the analysis performed from various traditional research publications, it has been observed that Drebin's dataset of malicious application is used by the most of the researcher to perform analysis using its techniques. In the view of this,

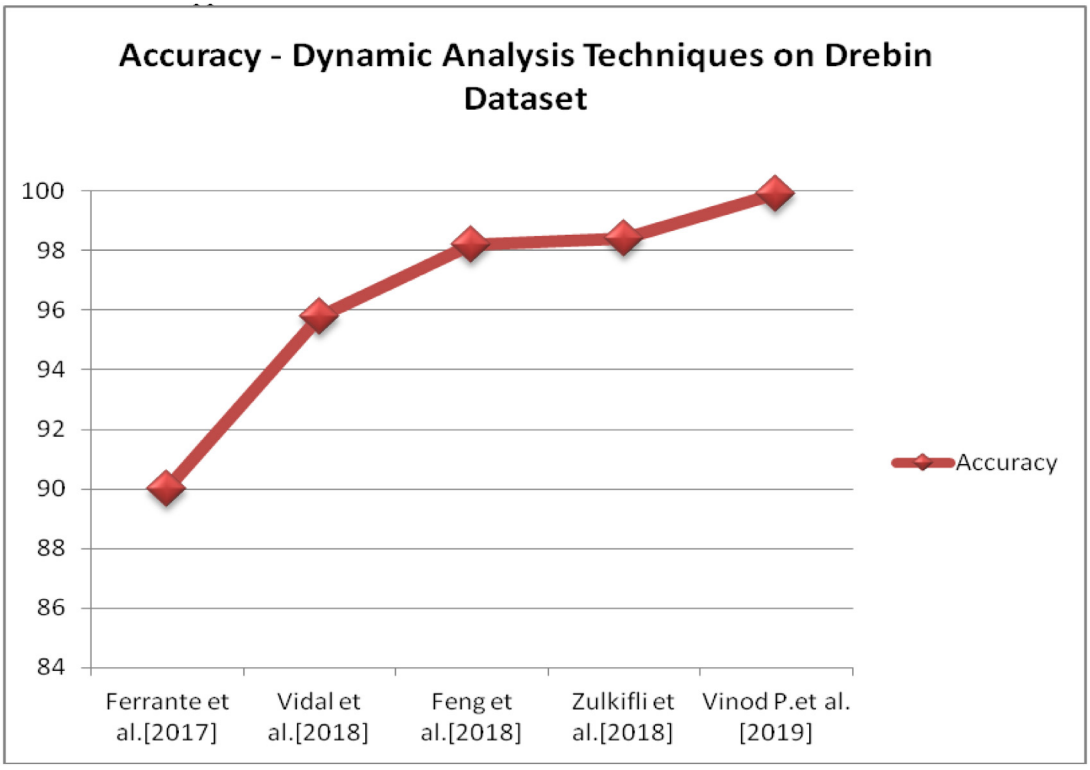

Fig. 17. Accuracy — Dynamic analysis techniques on Drebin dataset.

performance of various static and dynamic analysis techniques is compared in this module.

Evaluation parameter:
Accuracy: It is the measure of correctly predicting instances as either malware application or benign application from the specified dataset [286,387].

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Here, TP — true positives, TN — True negatives, FP — false positives and FN — false negatives.

4.9.1. Performance comparison of static analysis techniques on Drebin dataset

As per Fig. 16, it shows a line graph of accuracy comparison of various techniques used by the researchers, it covers 7 papers using from 2015 to 2019. In this all the analysis are performed on Drebin dataset with sample value 5560 applications and these papers are chosen according to the most usable techniques as per the analysis performed in the static analysis techniques. Permissions and API's are the two methods which are used by most of the researchers. So, all the mentioned analysis techniques in the chart are based on both the static techniques using common dataset (Drebin). Singh et al. [268] in 2017 and Arshad et al. [10] in 2019 has shown more accuracy as compared to others, it has been observed that they have taken API's as a parameter along with the permissions. Mazlan and Hamid [115] has performed analysis with more accuracy as compared to the but dataset sample was very less in that case, they have analyzed only 500 malicious applications.

4.9.2. Performance comparison of dynamic analysis techniques on Drebin dataset

There are number of technique and features which are used to perform dynamic analysis in malicious application detection android system. But as per the analysis performed in the previous modules shown in Fig. 17, it has been observed that system calls and network traffic is used by the most of the researcher to perform the dynamic analysis. It has also been observed that most of the analysis methods are performed on the Drebin dataset of malicious applications. So here we have compared the performance of various methods developed by the researchers based on these two features and common dataset. Some of the researcher has added some more features along with these two. Feng et al. [279] and Vinod et al. [258] has performed analysis with high accuracy and used system calls as their parameter for analysis, they have also used machine learning algorithms for classification. Zulkifli et al. [286] analyzed network traffic in their detection technique and achieved accuracy more that 98 percent.

5. Inferences

Fig. 18 shows the synthesis of literature review, it illustrates various detection techniques along with machine learning classification and datasets used. In this figure, we reported the studies which has used machine learning with any of detection technique. Two mostly used malware datasets are mentioned in the figure.

This section discusses some inferences from the literature review:

- Mostly static analysis techniques are used for malware detection as compared to other techniques. This is due to ease of usage and no need of execution of code. Permissions (S1) and API calls (S6) are the main features which are used by the most of the researchers. They are used individually or as a combination with other parameters. But APK/API image (S4) and overlays (S13) can also be explored to create efficient malicious application detection system because these are the least used features. Strings (S10), data flow (S14) and URL (S17) are also used only in one study with machine learning in 2018, so it needs to be explored,

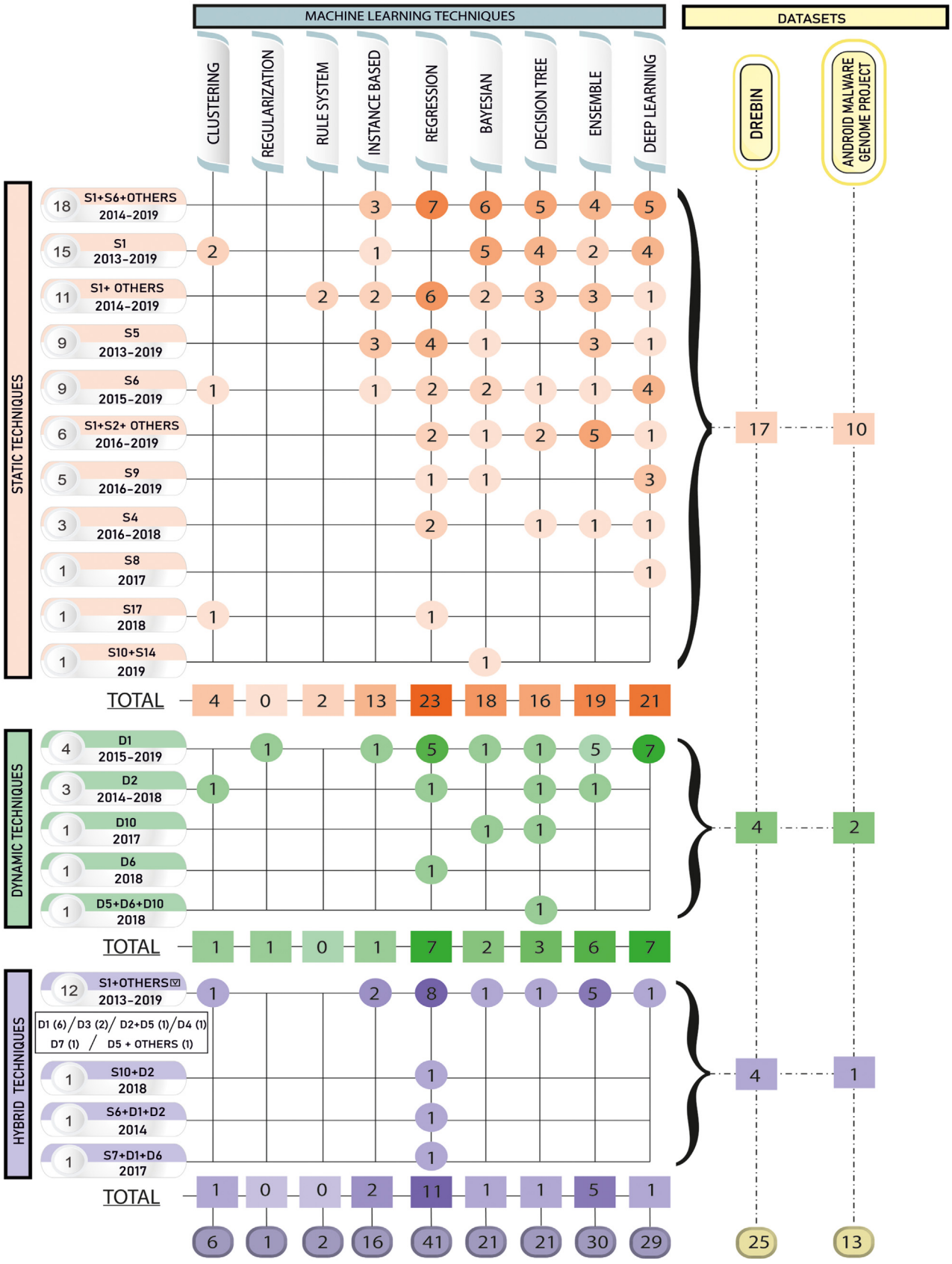

Fig. 18. Synthesis of literature review.

Table 8
Various dataset used in recent research related to malicious application detection in android.

Name of dataset	Code	Type (Malicious/Benign)	Count	Citations
Google play store	DS1	Benign	70	[5–7,10,12–14,22,24,48,54,60,63,72,75,78,79,91,99,102,106,108, 112,117,118,120,121,126,127,131,149,153–155,161,163,165–167, 173,178,187,188,192,196,198,200,201,223,233,235,238,242,244, 258,261,264,267,269,274,278,279,305,317,326,353,363,372,377].
Chinese market	DS2	Benign	2	[212,258]
Koodous	DS3	Benign/Malicious	5	[27,143,206,258,332]
Drebin	DS4	Malicious	56	[5,10,14,22,24,51,58,78,99,100,104,110,112,115,116,120,131,143, 149,153,157,168,176,187,192,201,205,219,223,230,230,233,240, 244,246,253,258,259,261,267,274,279,286,288,347,352,353,360–362,368,374,376,378–380].
AMD dataset	DS5	Malicious	7	[79,91,217,230,271,341,374].
Android Malware Genome project	DS6	Malicious	42	[5,6,24,63,72,76,78,100,119,121,124,125,127,132,143,155,163,165, 169,170,177,191,203,207,208,214,229,238,242,259,271,293,305, 313,314,326,352,362,378,380,381]
Androzoo	DS7	Benign/Malicious	8	[60,70,207,230,279,332,380,382]
Contagio mobile	DS8	Malicious	24	[5,12,22,24,78,113,118,126,143,158,162,165,167,177,205,223,225, 228,246,278,286,313,372]
Andro-Dumpsys	DS9	Benign/Malicious	2	[79,212].
AndroMalShare Project	DS10	Malicious	3	[24,79,314]
Anzhi	DS11	Benign	6	[117,184,198,295,316,354].
McAfee Labs	DS12	Benign	3	[170,260,362].
VirusShare	DS13	Malicious	15	[24,48,59,82,85,89,99,108,114,117,125,126,163,165,177,201,203, 205,207,238,354,363].
M0DROID	DS14	Benign/Malicious	4	[3,83,141,352].
Playdrone dataset	DS15	Benign	4	[48,201,217,313].
SlideME	DS16	Benign	1	[84,198,314].
MUDFLOW	DS17	Malicious	1	[58].
Xiaomi App store	DS18	Benign	4	[89,118,198,295]. ‘
AndroidDrawer	DS19	Benign	2	[198,217].
Apkmirror	DS20	Benign	1	[198].
AppsApk	DS21	Benign/ Malicious	1	[198,317].
ChinaMobile	DS22	Benign	1	[198].
Coolapk	DS23	Benign	1	[198].
Eoemarket	DS24	Benign	1	[198].
FDroid	DS25	Benign	1	[198].
Flyme	DS26	Benign	1	[198].
GetJar	DS27	Benign	1	[198].
GFan	DS28	Benign	2	[198,295,314].
Huawei	DS29	Benign	3	[118,198,295].
Wandoujia	DS30	Benign	2	[198,383].
Wangyi	DS31	Benign	1	[198].
MIGDroid	DS32	Malicious	1	[54]
Kaspersky	DS33	Malicious	1	[99].
Appchina	DS34	Benign	2	[169,314].
Hiapk	DS35	Benign	2	[314,316].
Mumayi	DS36	Benign	1	[314].
Pandaapp	DS37	Benign	2	[314,317].
DroidKin	DS38	Malicious	1	[314].
NQ Mobile	DS39	Malicious	1	[214].
T-Market	DS40	Benign/Malicious	1	[250].
Comodo cloud security center	DS41	Benign/Malicious	2	[220,222].
Baidu	DS42	Benign/Malicious	4	[59,168,295,316].
Lenovomm	DS43	Malicious	1	[316].
Nduo	DS44	Malicious	1	[316].
Anruan	DS45	Malicious	1	[317].
Canadian institute of cybersecurity's Android Botnet	DS46	Malicious	1	[154,384].
MobileSandbox project	DS47	Malicious	2	[192,244].
FalDroid	DS48	Malicious	1	[212].
ISCX malware (Canadian Institute of Cybersecurity)	DS49	Malicious	1	[176]
UpDroid	DS50	Malicious	1	[374].
VirusTotal	DS51	Benign/Malicious	5	[24,191,245,352,385].
MassVet	DS52	Malicious	1	[363].
YingYongBao stor	DS53	Benign	1	[228].
Kaggle.com	DS54	Benign/Malicious	1	[140].
Antimalwar	DS55	Benign/Malicious	1	[137].
Sogou	DS56	Benign	1	[295].
163 app store	DS57	Benign	1	[295].
10086 app store	DS58	Benign	1	[295].
91 app store	DS59	Benign	1	[295].
360 app store	DS60	Benign	1	[295].
Appgionee	DS61	Benign	1	[295].
Wang's repository	DS62	Benign/Malicious	1	[113]
Marvin	DS63	Benign/Malicious	1	[205].

(*continued on next page*)

Table 8 (*continued*).

Name of dataset	Code	Type (Malicious/Benign)	Count	Citations
Android data validation dataset (Canadian Institute of Cybersecurity)	DS64	Malicious	1	[154]
Android Adware and General Malware Dataset (CIC-AAGM2017)- (Canadian Institute of Cybersecurity)	DS65	Malicious	1	[386]
Android Malware Dataset (CIC-AndMal2017)- (Canadian Institute of Cybersecurity)	DS66	Malicious	1	[386]

- In Dynamic analysis techniques, system calls analysis (D1) is the most used feature in dynamic detection process. Network traffic (D2) is second most used feature in combination with machine learning techniques as shown in Fig. 18. It can be implemented to track the uncovered sections in static analysis but it needs some more caution because intelligent dynamic codes are injected by the attackers in the applications. That may keep then inactive whenever analysis is in process. So, there is a need to track these types of intelligent codes while performing dynamic analysis.
- It has been observed that in hybrid detection processes, combination of permissions (S1) and system calls (D1) are used in most of the research articles. Hybrid techniques can perform well but it makes the system complex. So, there is need to focus on lightweight hybrid analysis which cover all the analysis processes and produces high accuracy.
- Machine learning is used widely for the classification of applications to be benign or malicious based on features. We observed in our review that some intelligent malware families use code to mislead the training process of algorithms to increase the false positive rate at the time of evaluation. Fig. 18 shows that regression is the most used category in all the detection processes. Deep learning and ensemble are also used up to a great extent but rule-based system, clustering and regularization are least preferred classification techniques. In hybrid detection systems, only regression and ensemble methods are preferred. On the other hand, in dynamic systems regression and deep learning are preferred. In addition to these, ensemble is also used in many studies.
- 66 datasets are identified from various studies when we conducted the review. With reference to machine learning Drebin and Android malware genome projects are mentioned in most of the articles.
- Repackaging of application is also very popular to mislead the innocent application users. Repackaged apps are published on third party application stores. Therefore, the specific analysis of third-party apps stores is the need of the hour as it is used by the suspects to spread repackaged malicious application.

6. Conclusion and future research

To sum up, the review is conducted on 380 research papers which are identified from various international research articles from reputed electronic sources. The prime focus of study is to identify various android malicious application detection techniques. The results are presented in various forms such as tables, line charts, pie charts, flow diagrams and mapping table. It is found that detection system is divided into three categories: Static, dynamic and hybrid. It is identified that both static and dynamic techniques are used by veterans according to their requirements. In this research, 17 features are identified for static techniques. However, from all the static features permissions and API's are frequently used by researchers. APK image and overlays features are used by few researchers but implemented in latest research publications. In case of dynamic analysis approaches, 12 features are identified. Out of all the features, system calls and

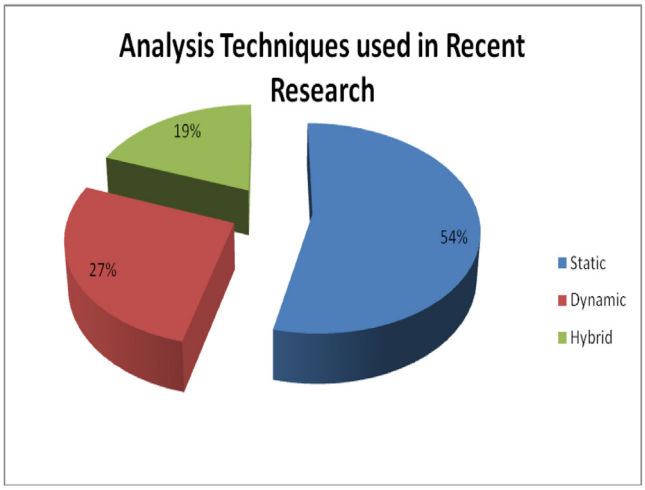

Fig. 19. Analysis techniques used in recent research.

network traffic analysis is used in most of papers as compared to others. The third category comprises of hybrid, 19 different hybrid approaches are identified which used the combination of static and dynamic features. In these permissions, API's, system calls, network traffic flow are generally used in different combinations. The combination of permission and API with system calls is mostly preferred by the researchers to develop the hybrid technique. Moreover, if we compare the utilization of these three techniques, it is found that static technique is used by in 54% of research papers whereas both dynamic and hybrid contributes the other half. These facts show that static technique is more popular as compared to dynamic and hybrid. It is quite evident that machine learning is used to classify malicious and legitimate applications (see Fig. 19).

44 different approaches of machine learning are identified and categorized in nine sub categories according to their working purpose (see Fig. 20).

Out of them SVM, random forest, naïve Bayesian network, KNN and decision tree are used mostly. Researchers conclude that the accuracy of SVM is more as compared to other techniques. After performing research, it is found that 66 types of datasets are used by veterans to perform experiments related to malicious application detection technique. These datasets are taken from different areas such antivirus companies, websites, projects working on malicious application detection, third party app stores and Google play store. It has been observed that Google play store is used in most research articles to work with legitimate application. However, Drebin and Android Malware Genome Project are used to collect malicious application samples. The above findings of research will help any researcher who wants to carry out work in android malicious application detection in various domains such as static, dynamic and hybrid using machine learning. This paper will guide them about the techniques, various features used and their utilized in these techniques, classification techniques and datasets. The main point is that count of researchers who

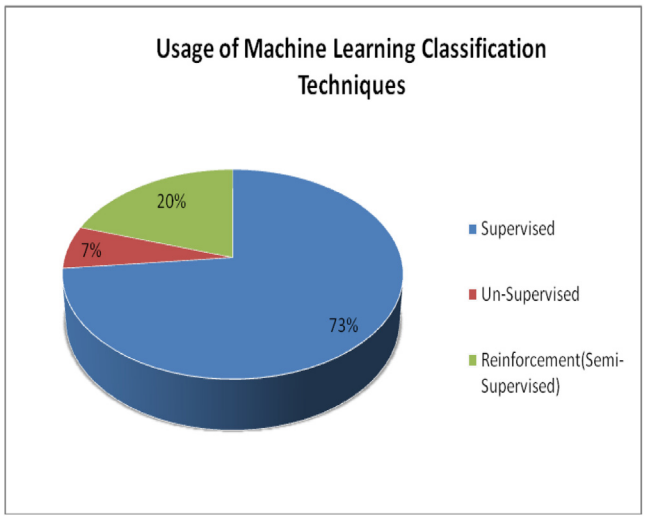

Fig. 20. Usage of machine learning classification techniques.

have used the technique is also mentioned in this research report which will be beneficial for future researches to easily pick the techniques according to their requirement and utilization in traditional research.

In the Future, the researchers can work on various factors related to android malware detection such as hybrid analysis can be explored with different strategies, because there are a number of combinations of static and dynamic parameters are still untouched, therefore those parameters can be used for the betterment of results. Other thing that has been observed during the survey that attacker try to mislead the machine learning algorithms during learning process, that may affect the classification results, so this is also a key area of research for future to track this kind of applications. Moreover, traditional developed systems for detection are very hard and complex but the time wants lightweight and quick analysis processes that work fast efficiently and accurately. In addition to this third-party app stores are also a big challenge for this field there is a need to track all these third-party app stores and categorize them accordingly to make the malware detection process easy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Statista reports Mobile operating systems' market share worldwide from 2012 to 2019, 2020, [Online]. Available: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. [Accessed: 14-04-2020].
[2] H. Rawal, C. Parekh, Android Internal Analysis of APK by Droid_Safe & APK Tool, Int. J. Adv. Res. Comput. Sci. 8 (5) (2017) 2397–2402.
[3] M. Al Ali, D. Svetinovic, Z. Aung, S. Lukman, Malware detection in android mobile platform using machine learning algorithms, in: 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS), IEEE, 2017, pp. 763–768.
[4] K. Tam, A. Feizollah, N.B. Anuar, R. Salleh, L. Cavallaro, The evolution of android malware and android analysis techniques, ACM Comput. Surv. 49 (4) (2017).
[5] A.T. Kabakus, I.A. Dogru, An in-depth analysis of Android malware using hybrid techniques, Digit. Investig. 24 (2018) 25–33.
[6] K. Bakour, H.M. Ünver, R. Ghanem, The Android malware detection systems between hope and reality, SN Appl. Sci. 1 (9) (2019) 1120–1132.
[7] A. Reina, A. Fattori, L. Cavallaro, A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors, in: sixth European Workshop on Systems Security, Prague, Czech Republic, 2013.
[8] Gartner Says Huawei Secured (2) Worldwide Smartphone Vendor Spot, Surpassing Apple in Second Quarter 2018, 2018, [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter. [Accessed: 14-04-2020].
[9] J. Winter, S. Battisti, T. Burström, S. Luukkainen, Exploring the success factors of mobile business ecosystems, Int. J. Innov. Technol. Manage. 15 (3) (2018) 1–23.
[10] S. Arshad, M.A. Shah, A. Wahid, A. Mehmood, H. Song, H. Yu, SAMADroid: a novel 3-level hybrid malware detection model for android operating system, IEEE Access 6 (2018) 4321–4339.
[11] Asaf Shabtai, Malware detection on mobile devices, in: 2010 Eleventh International Conference on Mobile Data Management, IEEE, 2010, pp. 289–290.
[12] Y. Chang, S. Wang, The concept of attack scenarios and its applications in Android malware detection, in: 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2016, pp. 1485–1492.
[13] K. Allix, Q. Jerome, T.F. Bissyandé, J. Klein, R. State, Y.L. Traon, A forensic analysis of Android Malware–how is malware written and how it could be Detected?, in: 2014 IEEE 38th Annual Computer Software and Applications Conference, IEEE, 2014, pp. 384–393.
[14] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, C.E.R.T. Siemens, DREBIN: Effective and explainable detection of android malware in your pocket, in: Network and Distributed System Security Symposium (NDSS, 2014, pp. 23–26.
[15] E.B. Karbab, M. Debbabi, A. Derhab, D. Mouheb, Cypider: building community-based cyber-defense infrastructure for android malware detection, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 348–362.
[16] P. Agrawal, B. Trivedi, A survey on android malware and their detection techniques, in: 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), IEEE, 2019, pp. 1–6.
[17] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, R. Maskeliunas, Android malware detection: A survey, in: International Conference on Applied Informatics, Springer, 2018, pp. 255–266.
[18] I. Martín, J.A. Hernández, S. Santos, Machine-learning based analysis and classification of android malware signatures, Future Gener. Comput. Syst. 97 (2019) 295–305.
[19] M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, T. Liu, Android malware familial classification and representative sample selection via Frequent Subgraph Analysis, IEEE Trans. Inf. Forensics Secur. 13 (8) (2018) 1890–1905.
[20] J. Lee, S. Lee, H. Lee, Screening smartphone applications using malware family signatures, Comput. Secur. 52 (2015) 234–249.
[21] C.A. Castillo, Android malware past, present, and future, White Paper of McAfee Mobile Security Working Group 1, California, USA, 2011, pp. 1–27.
[22] A. Aldini, F. Martinelli, A. Saracino, D. Sgandurra, Detection of repackaged mobile applications through a collaborative approach, Concurr. Comput.: Pract. Exper. 27 (11) (2015) 2818–2838.
[23] J. Choi, W. Sung, C. Choi, P. Kim, Personal information leakage detection method using the inference-based access control model on the Android platform, Pervasive Mob. Comput. 24 (2015) 138–149.
[24] F. Idrees, M. Rajarajan, M. Conti, T.M. Chen, Y. Rahulamathavan, PIndroid: A novel Android malware detection system using ensemble learning methods, Comput. Secur. 68 (2017) 36–46.
[25] J. Tao, Y. Zhang, P. Cao, Z. Wang, Q. Zhao, An android malware detection system based on behavior comparison analysis, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2017, pp. 387–396.
[26] H. Chi, X. Simms, A fast approach towards Android malware detection, in: International Conference on Computational Science and its Applications, Springer, 2015, pp. 77–89.
[27] A. Atzeni, F. Díaz, A. Marcelli, A. Sánchez, G. Squillero, A. Tonda, Countering android malware: A scalable semi-supervised approach for family-signature generation, IEEE Access 6 (2016) 59540–59556.
[28] D. Rattan, R. Bhatia, M. Singh, Software clone detection: A systematic review, Inf. Softw. Technol. 55 (7) (2013) 1165–1199.
[29] B. Kitchenham, P. Brereton, Z. Li, D. Budgen, A. Burn, Repeatability of systematic literature reviews, in: 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011), IET, 2011, pp. 46–55.
[30] B. Kitchenham, Procedures for performing systematic reviews, Keele UK Keele Univ. 33 (2004) 1–26.

[31] N. Alomar, M. Alsaleh, A. Alarifil, Social authentication applications, attacks, defense strategies and future research directions: a systematic review, IEEE Commun. Surv. Tutor. 19 (2) (2017) 1080–1111.
[32] B. Kitchenham, O. Pearl Brereton, D. Budgen, Protocol for Extending an Existing Tertiary Study of Systematic Literature Reviews in Software Engineering, 2017.
[33] B. Kitchenham, L. Madeyski, P. Brereton, Meta-analysis for families of experiments in software engineering: a systematic review and reproducibility and validity assessment, Empir. Softw. Eng. 25 (1) (2020) 353–401.
[34] M. Conti, Q.Q. Li, A. Maragno, R. Spolaor, The dark side (-channel) of mobile devices: A survey on network traffic analysis, IEEE Commun. Surv. Tutor. 20 (4) (2018) 2658–2713.
[35] M.L. Polla, F. Martinelli, D. Sgandurra, A survey on security for mobile devices, IEEE Commun. Surv. Tutor. 15 (1) (2012) 446–471.
[36] G. Suarez-Tangil, J.E. Tapiador, P. Peris-Lopez, A. Ribagorda, Evolution, detection and analysis of malware for smart devices, IEEE Commun. Surv. Tutor. 16 (2) (2016) 961–987.
[37] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M.S. Gaur, M. Conti, M. Rajarajan, Android security: a survey of issues, malware penetration, and defenses, IEEE Commun. Surv. Tutor. 17 (2) (2014) 998–1022.
[38] D.J. Tan, T. Chua, V.L. Thing, Securing android: a survey, taxonomy, and challenges, ACM Comput. Surv. 47 (4) (2015) 1–45.
[39] P. Yan, Z. Yan, A survey on dynamic mobile malware detection, Softw. Qual. J. 26 (3) (2018) 891–919.
[40] R. Zachariah, K. Akash, M.S. Yousef, A.M. Chacko, Android malware detection a survey, in: 2017 IEEE International Conference on Circuits and Systems (ICCS), IEEE, 2017, pp. 238–244.
[41] A. Souri, R. Hosseini, A state-of-the-art survey of malware detection approaches using data mining techniques, Hum.-Cent. Comput. Inf. Sci. 8 (1) (2018).
[42] H. Meng, V.L. Thing, Y. Cheng, Z. Dai, L. Zhang, A survey of Android exploits in the wild, Comput. Secur. 76 (2018) 71–91.
[43] S.S. Chakkaravarthy, D. Sangeetha, V. Vaidehi, A survey on malware analysis and mitigation techniques, Comp. Sci. Rev. 32 (2019) 1–23.
[44] Development of new Android malware worldwide from June 2016 to May 2019, 2020, [Online]. Available: https://www.statista.com/statistics/680705/global-android-malware-volume/. [Accessed: 14-04-2020].
[45] L. Chen, S. Hou, Y. Ye, Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks, in: Proceedings of the 33rd Annual Computer Security Applications Conference, 2017, pp. 362–372.
[46] N. Painter, B. Kadhiwala, Comparative analysis of android malware detection techniques, in: Proceedings of the International Conference on Data Engineering and Communication Technology, Springer, 2017, pp. 131–139.
[47] S. Khemani, D. Jain, G. Prasad, Android malware detection techniques, in: Emerging Research in Computing, Information, Communication and Applications, Springer, 2019, pp. 449–457.
[48] Tao Lei, Zhan Qin, Zhibo Wang, Qi Li, Dengpan Ye, Evedroid: Event-Aware Android malware detection against model degrading for IoT devices, IEEE Internet Things J. 6 (4) (2019) 6668–6680.
[49] T. Kim, B. Kang, M. Rho, S. Sezer, E.G. Im, A multimodal deep learning method for android malware detection using various features, IEEE Trans. Inf. Forensics Secur. 14 (3) (2018) 773–788.
[50] J. Gu, B. Sun, X. Du, J. Wang, Y. Zhuang, Z. Wang, Consortium blockchain-based malware detection in mobile devices, IEEE Access 6 (2018) 12118–12128.
[51] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, K. Ren, Android HIV: A study of repackaging malware for evading machine-learning detection, IEEE Trans. Inf. Forensics Secur. 15 (2019) 987–1001.
[52] K. Sugunan, T.G. Kumar, K.A. Dhanya, Static and dynamic analysis for android malware detection, in: Advances in Big Data and Cloud Computing., Springer, 2018, pp. 147–155.
[53] P.T. Duy, V. Pham, N.T. Cam, Eddleak: Enhancing precision of detecting inter-app data leakage in Android applications, in: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), IEEE, 2017, pp. 674–679.
[54] M. Fan, J. Liu, W. Wang, H. Li, Z. Tian, T. Liu, Dapasa: detecting android piggybacked apps through sensitive subgraph analysis, IEEE Trans. Inf. Forensics Secur. 12 (8) (2017) 1772–1785.
[55] D.V. Ng, J.G. Hwang, Android malware detection using the dendritic cell algorithm, in: 2014 International Conference on Machine Learning and Cybernetics, Vol. 1, IEEE, 2014, pp. 257–262.
[56] S.R. Tiwari, R.U. Shukla, An android malware detection technique using optimized permission and API with PCA, in: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE, 2018, pp. 2611–2616.
[57] W. Wang, Z. Gao, M. Zhao, Y. Li, J. Liu, X. Zhang, DroidEnsemble: Detecting Android malicious applications with Ensemble of string and Structural Static Features, IEEE Access (2018).
[58] P. Feng, C. Sun, J. Ma, Measuring the risk value of sensitive dataflow path in Android applications, Secur. Commun. Netw. 9 (18) (2016) 5918–5933.
[59] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, M. Conti, Detecting android malware leveraging text semantics of network flows, IEEE Trans. Inf. Forensics Secur. 13 (5) (2017) 1096–1109.
[60] C. Parker, J.T. McDonald, T. Johnsten, R.G. Benton, Android malware detection using step-size based multi-layered vector space models, in: 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, 2018, pp. 1–10.
[61] P. Kaur, S. Sharma, Spyware detection in android using hybridization of description analysis, permission mapping and interface analysis, Procedia Comput. Sci. 46 (2015) 794–803.
[62] X. Ju, Android malware detection through permission and package, in: Wavelet Analysis and Pattern Recognition (ICWAPR), IEEE, 2014.
[63] H. Han, Z. Chen, Q. Yan, L. Peng, L. Zhang, A real-time android malware detection system based on network traffic analysis, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2015, pp. 504–516.
[64] A. Kumar, K.P. Sagar, K.S. Kuppusamy, G. Aghila, Machine learning based malware classification for Android applications using multimodal image representations, in: 2016 10th International Conference on Intelligent Systems and Control (ISCO), IEEE, 2016, pp. 1–6.
[65] O.S. Adebayo, N. AbdulAziz, Android malware classification using static code analysis and Apriori algorithm improved with particle swarm optimization, in: 2014 4th World Congress on Information and Communication Technologies (WICT 2014), IEEE, 2014, pp. 123–128.
[66] J. Li, Z. Wang, T. Wang, J. Tang, Y. Yang, Y. Zhou, An android malware detection system based on feature fusion, Chin. J. Electron. 27 (6) (2016) 1206–1213.
[67] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, Z. Yan, Machine learning-based malicious application detection of android, IEEE Access 5 (2017) 25591–25601.
[68] H. Ali Alatwi, T. Oh, E. Fokoue, B. Stackpole, Android malware detection using category-based machine learning classifiers, in: Proceedings of the 17th Annual Conference on Information Technology Education, 2016, pp. 54–59.
[69] D. Wang, H. Jin, D. Zou, P. Xu, T. Zhu, G. Chen, Taming transitive permission attack via bytecode rewriting on Android application, Secur. Commun. Netw. 9 (13) (2016) 2100–2114.
[70] G. Shrivastava, P. Kumar, Android application behavioural analysis for data leakage, Expert Syst. (2019).
[71] C. Bae, S. Shin, A collaborative approach on host and network level android malware detection, Secur. Commun. Netw. 9 (18) (2016) 5639–5650.
[72] Y. Du, X. Wang, J. Wang, A static android malicious code detection method based on multi-source fusion, Secur. Commun. Netw. 8 (17) (2015) 3238–3246.
[73] S. Liang, M. Might, D.V. Horn, Anadroid: Malware analysis of android with user-supplied predicates, Electron. Notes Theor. Comput. Sci. 311 (2015) 3–14.
[74] A. Feizollah, N.B. Anuar, R. Salleh, G.S. Tangil, S. Furnell, Androdialysis: Analysis of android intent effectiveness in malware detection, Comput. Secur. 65 (2017) 121–134.
[75] K. Sokolova, C. Perez, M. Lemercier, Android application classification and anomaly detection with graph-based permission patterns, Decis. Support Syst. 93 (2017) 62–76.
[76] S. Sheen, R. Anitha, V. Natarajan, Android based malware detection using a multifeature collaborative decision fusion approach, Neurocomputing 151 (2015) 905–912.
[77] L. Nguyen-Vu, J. Ahn, S. Jung, Android fragmentation in malware detection, Comput. Secur. 87 (2019).
[78] K.A. Talha, D.I. Alper, C. Aydin, APK auditor: Permission-based Android malware detection system, Digit. Investig. 13 (2015) 1–14.
[79] L. Zhang, V. LL Thing, Y. Cheng, A scalable and extensible framework for android malware detection and family attribution, Comput. Secur. 80 (2019) 120–133.
[80] X. Wang, W. Wang, Y. He, J. Liu, Z. Han, X. Zhang, Characterizing Android apps' behavior for effective detection of malapps at large scale, Future Gener. Comput. Syst. 75 (2017) 30–45.
[81] D. Su, J. Liu, W. Wang, X. Wang, X. Du, M. Guizani, Discovering communities of malapps on Android-based mobile cyber-physical systems, Ad Hoc Netw. 80 (2018) 104–115.
[82] H.J. Zhu, Z.H. You, Z.X. Zhu, W.L. Shi, X. Chen, L. Cheng, DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model, Neurocomputing 272 (2018) 638–646.
[83] Z.U. Rehman, S.N. Khan, K. Muhammad, J.W. Lee, Z. Lv, S.W. Baik, P.A. Shah, K. Awan, I. Mehmood, Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Comput. Electr. Eng. 69 (2018) 828–841.

[84] V. Moonsamy, J. Rong, S. Liu, Mining permission patterns for contrasting clean and malicious android applications, Future Gener. Comput. Syst. 36 (2014) 122–132.
[85] K.O. Elish, X. Shu, D.D. Yao, B.G. Ryder, X. Jiang, Profiling user-trigger dependence for Android malware detection, Comput. Secur. 49 (2015) 255–273.
[86] Y. Zhang, W. Ren, T. Zhu, Y. Ren, SaaS: A situational awareness and analysis system for massive android malware detection, Future Gener. Comput. Syst. 95 (2019) 548–559.
[87] J. Xiao, K. Xu, J. Duan, Malicious android application detection based on composite features, in: Proceedings of the 3rd International Conference on Computer Science and Application Engineerin, 2019, pp. 1–6.
[88] W. Li, Z. Liu, Android malicious application detection method based on multi-class characteristics, in: Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence, 2019, pp. 157–161.
[89] J. Xiao, Z. Lu, Q. Xu, A new android malicious application detection method using feature importance score, in: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, 2018, pp. 145–150.
[90] S. Ngamwitroj, B. Limthanmaphon, Adaptive Android malware signature detection, in: Proceedings of the 2018 International Conference on Communication Engineering and Technology, 2018, pp. 22–25.
[91] S. Ilham, G. Abderrahim, B.A. Abdelhakim, Permission based malware detection in android devices, in: Proceedings of the 3rd International Conference on Smart City Applications, 2018, pp. 1–6.
[92] X. Liu, X. Dong, Q. Lei, Android malware detection based on multi-features, in: Proceedings of the 8th International Conference on Communication and Network Security, 2018, pp. 69–73.
[93] M. Leeds, M. Keffeler, T. Atkison, A comparison of features for android malware detection, in: Proceedings of the SouthEast Conference, 2017, pp. 63–68.
[94] Z. Wang, G. Li, Y. Chi, J. Zhang, T. Yang, Q. Liu, Android malware detection based on convolutional neural networks, in: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, 2019, pp. 1–6.
[95] P.M. Kate, S.V. Dhavale, Two phase static analysis technique for Android malware detection, in: Proceedings of the Third International Symposium on Women in Computing and Informatics, 2015, pp. 650–655.
[96] U. Pehlivan, N. Baltaci, C. Acartürk, N. Baykal, 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), IEEE, 2014, pp. 1–8.
[97] M. Kakavand, M. Dabbagh, A. Dehghantanha, Application of machine learning algorithms for Android malware detection, in: Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems, 2018, pp. 32–36.
[98] N. He, T. Wang, P. Chen, H. Yan, Z. Jin, An android malware detection method based on deep autoencoder, in: Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference, 2018, pp. 88–93.
[99] D. Zhu, T. Xi, P. Jing, D. Wu, Q. Xia, Y. Zhang, A Transparent and Multimodal Malware Detection Method for Android Apps, in: Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2019, pp. 51–60.
[100] Y. Zhang, Y. Yang, X. Wang, A novel android malware detection approach based on convolutional neural network, in: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, 2018, pp. 144–149.
[101] Z. Yuan, Y. Lu, Z. Wang, Y. Xue, Droid-sec: deep learning in android malware detection, in: Proceedings of the 2014 ACM conference on SIGCOMM, 2014, pp. 371–372.
[102] P. Faruki, V. Ganmoor, V. Laxmi, M.S. Gaur, A. Bharmal, AndroSimilar: robust statistical feature signature for Android malware detection, in: Proceedings of the 6th International Conference on Security of Information and Networks, 2013, pp. 152–159.
[103] C. Wang, Q. Xu, X. Lin, S. Liu, Research on data mining of permissions mode for Android malware detection, Cluster Comput. 22 (6) (2019) 13337–13350.
[104] A.K. Singh, C.D. Jaidhar, M.A.A. Kumara, Experimental analysis of Android malware detection based on combinations of permissions and API-calls, J. Comput. Virol. Hacking Tech. 15 (3) (2019) 209–218.
[105] W.Y. Lee, J. Saxe, R. Harang, SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks, in: Deep Learning Applications for Cyber Security, Springer, 2019, pp. 197–210.
[106] S. Roopak, T. Thomas, S. Emmanuel, Android malware detection mechanism based on bayesian model averaging, in: Recent Findings in Intelligent Computing Techniques, Springer, 2019, pp. 87–96.
[107] D. Zhu, T. Xi, Permission-based feature scaling method for lightweight Android malware detection, in: International Conference on Knowledge Science, Engineering and Management, Springer, 2019, pp. 714–725.
[108] C. Zhao, C. Wang, W. Zheng, Android malware detection based on sensitive permissions and APIs, in: International Conference on Security and Privacy in New Computing Environments, Springer, 2019, pp. 105–113.
[109] T. Li, B. Wu, W. Wen, Android malware detection method based on frequent pattern and weighted naive Bayes, in: China Cyber Security Annual Conference, Springer, 2018, pp. 36–51.
[110] A. Firdaus, N.B. Anuar, A. Karim, M.F. Ab Razak, Discovering optimal features using static analysis and a genetic search based method for Android malware detection, Front. Inf. Technol. Electron. Eng. 19 (6) (2018) 712–736.
[111] F. Shang, Y. Li, X. Deng, D. He, Android malware detection method based on naive Bayes and permission correlation algorithm, Cluster Comput. 21 (1) (2018) 955–966.
[112] M.M. John, P. Vinod, Statistical approach using meta features for Android malware detection system, in: Cyber Security, Springer, 2018, pp. 269–279.
[113] A. Bhattacharya, R.T. Goswami, A hybrid community based rough set feature selection technique in Android Malware detection, in: Smart Trends in Systems, Security and Sustainability, Springer, 2018, pp. 249–258.
[114] Y. Zhao, G. Xu, Y. Zhan, HFA-MD: An efficient hybrid features analysis based Android Malware Detection Method, in: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Springer, 2017, pp. 248–257.
[115] N.H. Mazlan, I.R.A. Hamid, Using weighted based feature selection technique for Android Malware Detection, in: International Conference on Mobile and Wireless Technology, Springer, 2017, pp. 54–64.
[116] M.S. Rana, S.S.M.M. Rahman, A.H. Sung, Evaluation of tree based machine learning classifiers for android malware detection, in: International Conference on Computational Collective Intelligence, Spinger, 2018, pp. 377–385.
[117] C. Liu, J. Li, M. Yu, B. Luo, S. Li, K. Chen, W. Huang, B. Lv, FGFDect: A fine-grained features classification model for Android Malware Detection, in: International Conference on Security and Privacy in Communication Systems, Springer, 2018, pp. 281–293.
[118] X. Li, G. Wang, S. Ali, Q. He, Android malware detection using category-based permission vectors, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2018, pp. 399–414.
[119] A. Altaher, An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features, Neural Comput. Appl. 28 (12) (2017) 4147–4157.
[120] F. Yang, Y. Zhuang, J. Wang, Android malware detection using hybrid analysis and machine learning technique, in: International Conference on Cloud Computing and Security, Springer, 2017, pp. 565–575.
[121] Y. Ding, S. Zhu, X. Xia, Android malware detection method based on function call graphs, in: International Conference on Neural Information Processing, Springer, 2016, pp. 70–77.
[122] J.T. Andoor, A filtering based Android Malware Detection system for google playstore, in: Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), Springer, 2015, pp. 559–566.
[123] P. de la, J. Gaviria, B. Sanz, I. S.Grueiro, P.G. Bringas, The evolution of permission as feature for Android malware detection, in: Computational Intelligence in Security for Information Systems Conference, Springer, 2015, pp. 389–400.
[124] A. Sharma, S.K. Dash, Mining api calls and permissions for android malware detection, in: International Conference on Cryptology and Network Security. Springer, pp . 191–205.
[125] B. Wolfe, K.O. Elish, D.D. Yao, Comprehensive behavior profiling for proactive android malware detection, in: International Conference on Information Security, Springer, 2014, pp. 328–344.
[126] G. Tao, Z. Zheng, Z. Guo, M.R. Lyu, MalPat: Mining patterns of malicious and benign Android apps via permission-related APIs, IEEE Trans. Reliab. 67 (1) (2017) 355–369.
[127] Z. Ni, M. Yang, Z. Ling, J. Wu, J. Luo, Real-time detection of malicious behavior in Android apps, in: 2016 International Conference on Advanced Cloud and Big Data (CBD), IEEE, 2016, pp. 221–227.
[128] P. Xiong, X. Wang, W. Niu, T. Zhu, G. Li, Android malware detection with contrasting permission patterns, China Commun. 11 (8) (2014) 1–14.
[129] S.J. Hussain, U. Ahmed, H. Liaquat, S. Mir, N.Z. Jhanjhi, M. Humayun, IMIAD: Intelligent malware identification for Android Platform, in: 2019 International Conference on Computer and Information Sciences (ICCIS), IEEE, 2019, pp. 1–6.
[130] S. Soviany, A. Scheianu, G. Suciu, A. Vulpe, O. Fratu, C. Istrate, Android malware detection and crypto-mining recognition methodology with machine learning, in: 2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC), IEEE, 2018, pp. 14–21.
[131] L.D. Coronado-De-Alba, A. Rodríguez-Mota, P.J. Escamilla-Ambrosio, Feature selection and ensemble of classifiers for Android malware detection, in: 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), IEEE, 2016, pp. 1–6.

[132] X. Zhang, D. Hu, Y. Fan, K. Yu, A novel android malware detection method based on markov blanket, in: 2016 IEEE First International Conference on Data Science in Cyberspace (DSC), IEEE, 2016, pp. 347–352.
[133] S. Sun, X. Fu, H. Ruan, X. Du, B. Luo, M. Guizani, Real-time behavior analysis and identification for Android application, IEEE Access 6 (2018) 38041–38051.
[134] S. Feldman, D. Stadther, B. Wang, Manilyzer: automated android malware detection through manifest analysis, in: 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, IEEE, 2014, pp. 767–772.
[135] M. Ganesh, P. Pednekar, P. Prabhuswamy, D.S. Nair, Y. Park, H. Jeon, Cnn-based android malware detection, in: 2017 International Conference on Software Security and Assurance (ICSSA), IEEE, 2017, pp. 60–65.
[136] S. Sabhadiya, J. Barad, J. Gheewala, Android malware detection using deep learning, in: 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, 2019, pp. 1254–1260.
[137] M.S. Alam, S.T. Vuong, Random forest classification for detecting android malware, in: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, IEEE, 2013, pp. 663–669.
[138] C. Yuan, S. Wei, Y. Wang, Y. You, S. ZiLiang, Android applications categorization using bayesian classification, in: 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, 2016, pp. 173–176.
[139] T. Ban, T. Takahashi, S. Guo, D. Inoue, K. Nakao, Integration of multi-modal features for android malware detection using linear svm, in: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), IEEE, 2016, pp. 141–146.
[140] D.O. Şah ın, O.E. Kural, S. Akleylek, E. Kiliç, New results on permission based static analysis for android malware, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2018, pp. 1–4.
[141] F. Mohsen, H. Bisgin, Z. Scott, K. Strait, Detecting android malwares by mining statically registered broadcast receivers, in: 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), IEEE, 2017, pp. 67–76.
[142] B. Shrestha, D. Ma, Y. Zhu, H. Li, N. Saxena, Tap-wave-rub: Lightweight human interaction approach to curb emerging smartphone malware, IEEE Trans. Inf. Forensics Secur. 10 (11) (2015) 2270–2283.
[143] A. Arora, S.K. Peddoju, M. Conti, PermPair: Android malware detection using Permission Pairs, IEEE Trans. Inf. Forensics Secur. (2019).
[144] P.P. Chan, W. Song, Static detection of Android malware by using permissions and API calls, in: 2014 International Conference on Machine Learning and Cybernetics, Vol. 1, IEEE, 2014, pp. 82–87.
[145] S. Anwar, J.M. Zain, Z. Inayat, R.U. Haq, A. Karim, A.N. Jabir, A static approach towards mobile botnet detection, in: 2016 3rd International Conference on Electronic Design (ICED), IEEE, 2016, pp. 563–567.
[146] D. Nguyen, V. Tong, D. Tran, H. Tran, A. Mellouk, Mining frequent patterns for scalable and accurate malware detection system in Android, in: 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), IEEE, 2018, pp. 370–375.
[147] M. Qiao, A.H. Sung, Q. Liu, Merging permission and API features for Android malware detection, in: 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), IEEE, 2016, pp. 566–571.
[148] X. Li, J. Liu, Y. Huo, R. Zhang, Y. Yao, An Android malware detection method based on AndroidManifest file, in: 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), IEEE, 2016, pp. 239–243.
[149] W. Li, J. Ge, G. Dai, Detecting malware for android platform: An svm-based approach, in: 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, IEEE, 2015, pp. 464–469.
[150] L. Cen, C.S. Gates, L. Si, N. Li, A probabilistic discriminative model for android malware detection with decompiled source code, IEEE Trans. Dependable Secure Comput. 12 (4) (2014) 400–412.
[151] W. Glodek, R. Harang, Rapid permissions-based detection and analysis of mobile malware using random decision forests, in: MILCOM 2013-2013 IEEE Military Communications Conference, IEEE, 2017, pp. 980–985.
[152] Q. Fang, X. Yang, C. Ji, A hybrid detection method for Android Malware, in: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2019, pp. 2127–2132.
[153] K. Zhao, D. Zhang, X. Su, W. Li, Fest: A feature extraction and selection tool for Android malware detection, in: 2015 IEEE Symposium on Computers and Communication (ISCC), IEEE, 2015, pp. 714–720.
[154] H.C. Takawale, A. Thakur, Talos app: On-device machine learning using tensorflow to detect Android Malware, in: 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, IEEE, 2018, pp. 250–255.
[155] S. Kandukuru, R.M. Sharma, Android malicious application detection using permission vector and network traffic analysis, in: 2017 2nd International Conference for Convergence in Technology (I2CT), IEEE, 2017, pp. 1126–1132.
[156] S. Aonzo, A. Merlo, M. Migliardi, L. Oneto, F. Palmieri, Low-resource Footprint, data-driven malware detection on Android, IEEE Trans. Sustain. Comput. (2017).
[157] S. Morales-Ortega, P.J. Escamilla-Ambrosio, A. Rodriguez-Mota, L.D. Coronado-De-Alba, Native malware detection in smartphones with android os using static analysis, feature selection and ensemble classifiers, in: 2016 11th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, 2016, pp. 1–8.
[158] R. Riasat, M. Sakeena, A.H. Sadiq, Y. Wang, Onamd: an online android malware detection approach, in: 2018 International Conference on Machine Learning and Cybernetics (ICMLC), Vol. 1, IEEE, 2018, pp. 190–196.
[159] S. Liang, X. Du, Permission-combination-based scheme for android mobile malware detection, in: 2014 IEEE International Conference on Communications (ICC), IEEE, 2014, pp. 2301–2306.
[160] S.Y. Yerima, S. Sezer, I. Muttik, Android malware detection using parallel machine learning classifiers, in: 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, IEEE, 2014, pp. 37–42.
[161] N.B. Akhuseyinoglu, K. Akhuseyinoglu, AntiWare: An automated android malware detection tool based on machine learning approach and official market metadata, in: 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, 2016, pp. 1–7.
[162] D. Chen, H. Zhang, X. Zhang, D. Wang, Android malicious application detection based on ontology technology integrated with permissions and system calls, in: 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), IEEE, 2016, pp. 481–484.
[163] Z. Wang, K. Li, Y. Hu, A. Fukuda, W. Kong, Multilevel permission extraction in Android Applications for Malware Detection, in: 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), IEEE, 2019, pp. 1–5.
[164] W. Kuo, T. Liu, C. Wang, Study on Android hybrid malware detection based on machine learning, in: 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), IEEE, 2019, pp. 31–35.
[165] J. McGiff, W.G. Hatcher, J. Nguyen, W. Yu, E. Blasch, C. Lu, Towards multimodal learning for android malware detection, in: 2019 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2019, pp. 432–436.
[166] W. Li, Z. Wang, J. Cai, S. Cheng, An Android malware detection approach using weight-adjusted deep learning, in: 2018 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2018, pp. 437–441.
[167] C. Liu, Z. Zhang, S. Wang, An android malware detection approach using Bayesian inference, in: 2016 IEEE International Conference on Computer and Information Technology (CIT), IEEE, 2016, pp. 476–483.
[168] M. Yang, Q. Wen, Detecting android malware with intensive feature engineering, in: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2016, pp. 157–161.
[169] X. Liu, J. Liu, A two-layered permission-based android malware detection scheme, in: 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, IEEE, 2014, pp. 142–148.
[170] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, DynaLog: An automated dynamic analysis framework for characterizing android applications, in: 2016 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2016, pp. 1–8.
[171] A.A.A. Samra, K. Yim, O.A. Ghanem, Analysis of clustering technique in android malware detection, in: 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE, 2013, pp. 729–733.
[172] Z. Qin, Y. Xu, Y. Di, Q. Zhang, J. Huang, Android malware detection based on permission and behavior analysis, in: International Conference on Cyberspace Technology (CCT 2014), IEEE, 2014, pp. 085–94.
[173] P.R.K. Varma, K.P. Raj, K.V.S. Raju, 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), IEEE, 2017, pp. 294–299.
[174] K. Wang, T. Song, A. Liang, Mmda: Metadata based malware detection on android, in: 2016 12th International Conference on Computational Intelligence and Security (CIS), IEEE, 2016, pp. 598–602.
[175] A. Arora, S.K. Peddoju, 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), IEEE, 2018, pp. 808–813.
[176] A. Skovoroda, D. Gamayunov, Automated static analysis and classification of Android malware using permission and API calls models, in: 2017 15th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2017, pp. 243–24309.

[177] A. Saracino, D. Sgandurra, G. Dini, F. Martinelli, Madam: Effective and efficient behavior-based android malware detection and prevention, IEEE Trans. Dependable Secure Comput. 15 (1) (2016) 83–97.
[178] M. Deypir, A new approach for effective malware detection in android-based devices, in: 2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), IEEE, 2016, pp. 112–116.
[179] K. Kavitha, P. Salini, V. Ilamathy, Exploring the malicious android applications and reducing risk using static analysis, in: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, 2016, pp. 1316–1319.
[180] J.D. Koli, Randroid: android malware detection using random machine learning classifiers, in: 2018 Technologies for Smart-City Energy Security and Power (ICSESP), IEEE, 2018, pp. 1–6.
[181] A. Martín, A. Calleja, H.D. Menéndez, J. Tapiador, D. Camacho, ADROIT: Android malware detection using meta-information, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2016, pp. 1–8.
[182] Y. Liu, Y. Zhang, H. Li, X. Chen, A hybrid malware detecting scheme for mobile Android applications, in: 2016 IEEE International Conference on Consumer Electronics (ICCE), IEEE, 2016, pp. 155–156.
[183] H. Shahriar, M. Islam, V. Clincy, Android malware detection using permission analysis, in: SoutheastCon 2017, IEEE, 2017, pp. 1–6.
[184] W. Wang, Y. Li, X. Wang, J. Liu, X. Zhang, Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers, Future Gener. Comput. Syst. 78 (2018) 987–994.
[185] Y. Feng, S. Anand, I. Dillig, A. Aiken, Apposcopy: Semantics-based detection of android malware through static analysis, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 576–587.
[186] K.O. Elish, H. Cai, D. Barton, D. Yao, B.G. Ryder, Identifying mobile Inter-App communication risks, IEEE Trans. Mob. Comput. 19 (1) (2018) 90–102.
[187] D. Zhang, Y. Guo, D. Guo, R. Wang, G. Yu, Contextual approach for identifying malicious Inter-Component privacy leaks in Android apps, in: 2017 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2017, pp. 228–235.
[188] K. Xu, Y. Li, R.H. Deng, Iccdetector: Icc-based malware detection on android, IEEE Trans. Inf. Forensics Secur. 11 (6) (2016) 1252–1264.
[189] Y. Jeong, H. Lee, S. Cho, S. Han, M. Park, A kernel-based monitoring approach for analyzing malicious behavior on Android, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, 2014, pp. 1737-1738.
[190] L. Gheorghe, B. Marin, G. Gibson, L. Mogosanu, R. Deaconescu, V.G. Voiculescu, M. Carabas, Smart malware detection on Android, Secur. Commun. Netw. 8 (18) (2015) 4254–4272.
[191] N.T. Cam, N.C.H. Phuoc, NeSeDroid—Android malware detection based on Network Traffic and sensitive resource accessing, in: Proceedings of the International Conference on Data Engineering and Communication Technology, Springer, 2017, pp. 19–30.
[192] Q. Wu, Z. Qin, J. Zhang, H. Yin, G. Yang, K. Hu, Android malware detection using local binary pattern and principal component analysis, in: International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2017, pp. 262–275.
[193] M. Yang, Q. Wen, Detecting android malware by applying classification techniques on images patterns, in: 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), IEEE, 2017, pp. 344–347.
[194] Y. Ding, W. Zhao, Z. Wang, L. Wang, Automaticlly learning featurs of Android apps using CNN, in: 2018 International Conference on Machine Learning and Cybernetics (ICMLC), Vol. 1, IEEE, 2018, pp. 331–336.
[195] S. Wu, P. Wang, X. Li, Y. Zhang, Effective detection of android malware based on the usage of data flow APIs and machine learning, Inf. Softw. Technol. 75 (2016) 17–25.
[196] S. Alam, S.A. Alharbi, S. Yildirim, Mining nested flow of dominant APIs for detecting android malware, Comput. Netw. 167 (2020).
[197] P. Zegzhda, D. Zegzhda, E. Pavlenko, A. Dremov, Detecting Android application malicious behaviors based on the analysis of control flows and data flows, in: Proceedings of the 10th International Conference on Security of Information and Networks, 2017, pp. 280–283.
[198] G. Meng, Y. Xue, Z. Xu, Y. Liu, J. Zhang, 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, 2016, pp. 306–317.
[199] Y. Feng, I. Dillig, S. Anand, A. Aiken, Apposcopy: automated detection of Android malware (invited talk), in: Proceedings of the 2nd International Workshop on Software Development Lifecycle for Mobile, 2014, pp. 13–14.
[200] H. Gascon, F. Yamaguchi, D. Arp, K. Rieck, Structural detection of android malware using embedded call graphs, in: Proceedings of the 2013 ACM workshop on Artificial intelligence and security, 2013, pp. 45–54.
[201] L. Onwuzurike, E. Mariconti, P. Andriotis, E. Cristofaro, G. Ross, G. Stringhini, MaMaDroid: Detecting android malware by building Markov chains of behavioral models (extended version, ACM Trans. Priv. Secur. 22 (2) (2019) 1–34.
[202] S. Badhani, S.K. Muttoo, Android malware detection using code graphs, in: System Performance and Management Analytics, Springer, 2019, pp. 203–215.
[203] Y. Liu, L. Zhang, X. Huang, Using g features to improve the efficiency of function call graph based android malware detection, Wirel. Pers. Commun. 103 (4) (2018) 2947–2955.
[204] A. Narayanan, M. Chandramohan, L. Chen, Y. Liul, A multi-view context-aware approach to android malware detection and malicious code localization, Empir. Softw. Eng. 23 (3) (2018) 1222–1274.
[205] Z. Xu, K. Ren, S. Qin, F. Craciun, Cdgdroid: Android malware detection based on deep learning using CFG and DFG, in: International Conference on Formal Engineering Methods, Springer, 2018, pp. 177–193.
[206] M. Leslous, V.V.T. Tong, J. Lalande, T. Genet, Gpfinder: Tracking the invisible in android malware, in: 2017 12th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, 2017, pp. 39–46.
[207] N. Xie, F. Zeng, X. Qin, Y. Zhang, M. Zhou, C. Lv, 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), IEEE, 2018, pp. 52–59.
[208] M.A. Atici, S. Sagiroglu, I.A. Dogru, Android malware analysis approach based on control flow graphs and machine learning algorithms, in: 2016 4th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2016, pp. 26–31.
[209] C. Liu, J. Li, M. Yu, G. Li, B. Luo, K. Chen, J. Jiang, W. Huang, Uref-flow: A unified android malware detection model based on reflective calls, in: 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), IEEE, 2018, pp. 1–7.
[210] X. Ge, Y. Pan, Y. Fan, C. Fang, Amdroid: Android malware detection using function call graphs, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 71–77.
[211] J. Kwon, J. Jeong, J. Lee, H. Lee, Droidgraph: discovering android malware by analyzing semantic behavior, in: 2014 IEEE Conference on Communications and Network Security, IEEE, 2014, pp. 498–499.
[212] W. Wang, J. Wei, S. Zhang, X. Luo, Lscdroid: Malware detection based on local sensitive API invocation sequences, IEEE Trans. Reliab. (2019).
[213] Z. Ma, H. Ge, Y. Liu, M. Zhao, J. Ma, A combination method for android malware detection based on control flow graphs and machine learning algorithms, IEEE Access 7 (2019) 21235–21245.
[214] S. Zou, J. Zhang, X. Lin, An effective behavior-based android malware detection system, Secur. Commun. Netw. 8 (12) (2015) 2079–2089.
[215] G. Ali, I. Alisha, B. Saltaformaggio, D. Xu, G.G.R. III, Toward a more dependable hybrid analysis of android malware using aspect-oriented programming, Comput. Secur. 73 (2018) 235–248.
[216] S. Hou, A. Saas, L. Chen, Y. Ye, T. Bourlai, Deep neural networks for automatic android malware detection, in: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2017, pp. 803-810.
[217] P. Zegzhda, D. Zegzhda, E. Pavlenko, G. Ignatev, Applying deep learning techniques for Android malware detection, in: Proceedings of the 11th International Conference on Security of Information and Networks, 2018, pp. 1-8.
[218] J. Allen, M. Landen, S. Chaba, Y. Ji, S.P.H. Chung, W. Lee, Improving accuracy of Android malware detection with lightweight contextual awareness, in: Proceedings of the 34th Annual Computer Security Applications Conference, 2018, pp. 210-221.
[219] R. Kumar, Z. Xiaosong, R.U. Khan, J. Kumar, I. Ahad, Effective and explainable detection of Android malware based on machine learning algorithms, in: Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, 2018, pp. 35-40.
[220] S. Hou, Y. Ye, Y. Song, M. Abdulhayoglu, Hindroid: An intelligent android malware detection system based on structured heterogeneous information network, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1507-1515.
[221] A. Martín, H.D. Menéndez, D. Camacho, Mocdroid: multi-objective evolutionary classifier for android malware detection, Soft Comput. 21 (24) (2017) 7405–7415.
[222] S. Hou, A. Saas, Y. Ye, L. Chen, Droiddelver: An android malware detection system using deep belief network based on api call blocks, in: International Conference on Web-Age Information Management, Springer, 2016, pp. 54–66.
[223] P. Zhang, S. Cheng, S. Lou, F. Jiang, A novel android malware detection approach using operand sequences, in: 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), IEEE, 2018, pp. 1–5.

[224] X. Liao, Z. Geng, Y. Meng, Y. Yu, Y. Li, D. Kang, A detection method for android repackaged applications with malicious features similarity of family homology, in: 2018 International Computers, Signals and Systems Conference (ICOMSSC), IEEE, 2018, pp. 853–856.

[225] R. Nix, J. Zhang, Classification of android apps and malware using deep neural networks, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1871–1878.

[226] J. Zhu, Z. Wu, Z. Guan, Z. Chen, API sequences based malware detection for android, in: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), IEEE, 2015, pp. 673–676.

[227] K. Xu, Y. Li, R. Deng, K. Chen, J. Xu, Droidevolver: Self-evolving android malware detection system, in: 2019 IEEE European Symposium on Security and Privacy (EuroS & P), IEEE, 2019, pp. 47–62.

[228] Y. Sun, Y. Xie, Z. Qiu, Y. Pan, J. Weng, S. Guo, Detecting android malware based on extreme learning machine, in: 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, 2017, pp. 47–53.

[229] D. Quan, L. Zhai, F. Yang, P. Wang, Detection of android malicious apps based on the sensitive behaviors, in: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2014, pp. 877–883.

[230] H. Zhang, S. Luo, Y. Zhang, L. Pan, An efficient android malware detection system based on method-level behavioral semantic analysis, IEEE Access 7 (2019) 69246–69256.

[231] Q. Li, X. Li, Android malware detection based on static analysis of characteristic tree, in: 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, IEEE, 2015, pp. 84–91.

[232] P. Teufl, M. Ferk, A. Fitzek, D. Hein, S. Kraxberger, C. Orthacker, Malware detection by applying knowledge discovery processes to application metadata on the Android Market (Google Play), Secur. Commun. Netw. 9 (5) (2016) 389–419.

[233] F. Martinelli, F. Mercaldo, A. Saracino, Bridemaid: An hybrid tool for accurate detection of android malware, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 899-901.

[234] A. Munoz, I. Martin, A. Guzman, J.A. Hernandez, Android malware detection from google play meta-data: Selection of important features, in: 2015 IEEE Conference on Communications and Network Security (CNS), IEEE, 2015, pp. 701–702.

[235] F. Martinelli, F. Mercaldo, A. Saracino, C.A. Visaggio, I find your behavior disturbing: Static and dynamic app behavioral analysis for detection of android malware, in: 2016 14th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2016, pp. 129–136.

[236] P. Wang, W.J. Chao, K. Chao, C. Lo, Using taint analysis for threat risk of cloud applications, in: 2014 IEEE 11th International Conference on E-Business Engineering, IEEE, 2014, pp. 185–190.

[237] C. Chen, J. Lin, G. Lai, Detecting mobile application malicious behaviors based on data flow of source code, in: 2014 International Conference on Trustworthy Systems and their Applications, IEEE, 2014, pp. 1–6.

[238] D. Zhu, H. Jin, Y. Yang, D. Wu, W. Chen, Deepflow: Deep learning-based malware detection by mining android application for abnormal usage of sensitive data, in: 2017 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2017, pp. 438–443.

[239] D. Li, L. Zhao, Q. Cheng, N. Lu, W. Shi, Opcode sequence analysis of android malware by a convolutional neural network, in: Concurrency and Computation: Practice and Experience, 2019.

[240] G. Canfora, F. Mercaldo, C.A. Visaggio, An hmm and structural entropy based detector for android malware: An empirical study, Comput. Secur. 61 (2016) 1–18.

[241] M. Amin, T.A. Tanveer, M. Tehseen, M. Khan, F.A. Khan, S. Anwar, Static malware detection and attribution in android byte-code through an end-to-end eep system, Future Gener. Comput. Syst. 102 (2020) 112–126.

[242] N. McLaughlin, J.M. Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, Deep android malware detection, in: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017, pp. 301–308.

[243] G. Canfora, F. Mercaldo, C.A. Visaggio, Mobile malware detection using op-code frequency histograms, in: 2015 12th International Joint Conference on E-Business and Telecommunications (ICETE), Vol. 4, IEEE, 2015, pp. 27–38.

[244] J. Zhang, Z. Qin, K. Zhang, H. Yin, J. Zou, Dalvik opcode graph based android malware variants detection using global topology features, IEEE Access 6 (2018) 51964–51974.

[245] B. Sanz, I. Santos, X. Ugarte-Pedrero, C. Laorden, J. Nieves, P.G. Bringas, Anomaly detection using string analysis for android malware detection, in: International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, Springer, 2014, pp. 469–478.

[246] S. Lou, S. Cheng, J. Huang, F. Jiang, Tfdroid: Android malware detection by topics and sensitive data flows using machine learning techniques, in: 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), IEEE, 2019, pp. 30–36.

[247] H. Gonzalez, N. Stakhanova, A.A. Ghorbani, Measuring code reuse in android apps, in: 2016 14th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2016, pp. 187–195.

[248] R. Dhaya, M. Poongodi, Detecting software vulnerabilities in android using static analysis, in: 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, IEEE, 2014, pp. 915–918.

[249] W. Wang, I. Murynets, J. Bickford, C.V. Wart, G. Xu, What you see predicts what you get—lightweight agent-based malware detection, Secur. Commun. Netw. 6 (1) (2013) 33–48.

[250] Y. Yan, Z. Li, Q.A. Chen, C. Wilson, T. Xu, E. Zhai, Y. Li, Y. Liu, Understanding and Detecting Overlay-based Android Malware at Market Scales, in: Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, 2019, pp. 168-179.

[251] O. Tripp, M. Pistoia, P. Ferrara, J. Rubin, Pinpointing mobile malware using code analysis, in: 2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), IEEE, 2016, pp. 275–276.

[252] A. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K.A. Yuksel, S.A. Camtepe, S. Albayrak, Static analysis of executables for collaborative malware detection on android, in: 2009 IEEE International Conference on Communications, IEEE, 2009, pp. 1–5.

[253] Z. Xiong, T. Guo, Q. Zhang, Y. Cheng, K. Xu, Android malware detection methods based on the combination of clustering and classification, in: International Conference on Network and System Security, Springer, 2018, pp. 411–422.

[254] M. Zheng, M. Sun, J.C. Lui, Droidtrace: A ptrace based android dynamic analysis system with forward execution capability, in: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, 2014, pp. 128–133.

[255] X. Xiao, X. Xiao, Y. Jiang, X. Liu, R. Ye, Identifying android malware with system call co-occurrence matrices, Trans. Emerg. Telecommun. Technol. 27 (5) (2016) 675–684.

[256] A. Amamra, J. Robert, C. Talhi, Enhancing malware detection for android systems using a system call filtering and abstraction process, Secur. Commun. Netw. 8 (7) (2015) 1179–1192.

[257] K. Deepa, G. Radhamani, P. Vinod, M. Shojafar, N. Kumar, M. Conti, Identification of android malware using refined system calls, Concurr. Comput.-Pract. Exp. 31 (20) (2019).

[258] P. Vinod, A. Zemmari, M. Conti, A machine learning based approach to detect malicious android apps using discriminant system calls, Future Gener. Comput. Syst. 94 (2019) 333–350.

[259] J.M. Vidal, M.A.S. Monge, L.J.G. Villalba, A novel pattern recognition system for detecting android malware by analyzing suspicious boot sequences, Knowl.-Based Syst. 150 (2016) 198–217.

[260] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, DL-droid: Deep learning based android malware detection using real devices, Comput. Secur. 89 (2020).

[261] F. Martinelli, F. Marulli, F. Mercaldo, Evaluating convolutional neural network for effective mobile malware detection, Procedia Comput. Sci. 112 (2017) 2372–2381.

[262] Y.D. Lin, Y.C. Lai, C.H. Chen, H.C. Tsai, Identifying android malicious repackaged applications by thread-grained system call sequences, Comput. Secur. 39 (2013) 340–350.

[263] S. Bhandari, R. Panihar, S. Naval, V. Laxmi, A. Zemmari, M.S. Gaur, Sword: semantic aware android malware detector, J. Inform. Secur. Appl. 42 (2018) 46–56.

[264] G. Canfora, E. Medvet, F. Mercaldo, C.A. Visaggio, Detecting android malware using sequences of system calls, in: Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, Vol. 201, pp. 13-20.

[265] M. Dimjašević, S. Atzeni, I. Ugrina, Z. Rakamaric, Evaluation of android malware detection based on system calls, in: Proceedings of the 2016 ACM on International Workshop on Security and Privacy Analytics, 2016, pp. 1-8.

[266] I. Burguera, U. Zurutuza, S.N. 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, 2011, pp. 15-26.

[267] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, A.K. Sangaiah, Android malware detection based on system call sequences and LSTM, Multimedia Tools Appl. (4) (2019) 3979–3999.

[268] L. Singh, M. Hofmann, Dynamic behavior analysis of android applications for malware detection, in: 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT), IEEE, 2017, pp. 1–7.
[269] X. Xiao, Z. Wang, Q. Li, S. Xia, Y. Jiang, Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting android malware with system call sequences, IET Inf. Secur. 11 (1) (2016) 8–15.
[270] Y. Malik, C.R.S. Campos, F. Jaafar, Detecting android security vulnerabilities using machine learning and system calls analysis, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 109–113.
[271] A.S.M. Ahsan-Ul-Haque, M.S. Hossain, M. Atiquzzaman, Sequencing system calls for effective malware detection in android, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–7.
[272] M. Jaiswal, Y. Malik, F. Jaafar, Android gaming malware detection using system call analysis, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2018, pp. 1–5.
[273] H. Liang, Y. Song, D. Xiao, An end-to-end model for android malware detection, in: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2017, pp. 140–142.
[274] A. Ferrante, E. Medvet, F. Mercaldo, J. Milosevic, C.A. Visaggio, Spotting the malicious moment: Characterizing malware behavior using dynamic features, in: 2016 11th International Conference on Availability, Reliability and Security (ARES), IEEE, 2016, pp. 372–381.
[275] S. Shifu, A. Saas, L. Chen, Y. Ye, Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs, in: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), IEEE, 2016, pp. 104–111.
[276] M.R. Amin, M. Zaman, M.S. Hossain, M. Atiquzzaman, Behavioral malware detection approaches for android, in: 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.
[277] S. Zhang, X. Xiao, Cscdroid: Accurately detect android malware via contribution-level-based system call categorization, in: 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, 2017, pp. 193–200.
[278] V. Wahanggara, Y. Prayudi, Malware detection through call system on android smartphone using vector machine method, in: 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), IEEE, 2015, pp. 62–67.
[279] P. Feng, J. Ma, C. Sun, X. Xu, Y. Ma, A novel dynamic android malware detection system with ensemble learning, IEEE Access 6 (2018) 30996–31011.
[280] X. Su, D. Zhang, W. Li, X. Wang, Androgenerator: An automated and configurable android app network traffic generation system, Secur. Commun. Netw. 8 (18) (2015) 4273–4288.
[281] A. Shabtai, L.T. Chekina, D. Mimran, L. Rokach, B. Shapira, Y. Elovici, Mobile malware detection through analysis of deviations in application network behavior, Comput. Secur. 43 (2014) 1–18.
[282] A. Arora, S.K. Peddoju, Minimizing network traffic features for android mobile malware detection, in: Proceedings of the 18th International Conference on Distributed Computing and Networking, 2017, pp. 1-10.
[283] J. Gajrani, J. Sarswat, M. Tripathi, V. Laxmi, M.S. Gaur, M. Conti, A robust dynamic analysis system preventing Sandbox detection by Android malware, in Proceedings of the 8th International Conference on Security of Information and Networks, 2015, 290-295.
[284] J. Malik, R. Kaushal, CREDROID: Android malware detection by network traffic analysis, in: Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing, 2016, pp. 28-36.
[285] P. de la, J. Gaviria, I. Pastor-López, B. Sanz, P.G. Bringas, Network traffic analysis for android malware detection, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2019, pp. 468–479.
[286] A. Zulkifli, I.R.A. Hamid, W.M. Shah, Z. Abdullah, Android malware detection based on network traffic using decision tree algorithm, in: International Conference on Soft Computing and Data Mining, Springer, 2018, pp. 485–494.
[287] J. Li, L. Zhai, X. Zhang, D. Quan, Research of android malware detection based on network traffic monitoring, in: 2014 9th IEEE Conference on Industrial Electronics and Applications, IEEE, 2014, pp. 1739–1744.
[288] Z. Chen, H. Han, Q. Yan, B. Yang, L. Peng, L. Zhang, J. Li, A first look at android malware traffic in first few minutes, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, IEEE, 2015, pp. 206–213.
[289] A. Arora, S. Garg, S.K. Peddoju, Malware detection using network traffic analysis in android based mobile devices, in: 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, IEEE, 2014, pp. 66–71.
[290] T. Wei, C. Mao, A.B. Jeng, H. Lee, H. Wang, D. Wu, Android malware detection via a latent network behavior analysis, in: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2012, pp. 1251–1258.
[291] L. Tenenboim-Chekina, O. Barad, A. Shabtai, D. Mimran, L. Rokach, B. Shapira, Y. Elovici, Detecting application update attack on mobile devices through network featur, in: 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2013, pp. 91–92.
[292] K. Ariyapala, H.G. Do, H.N. Anh, W.K. Ng, M. Conti, A host and network based intrusion detection for android smartphones, in: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), IEEE, 2016, pp. 849–854.
[293] A. Feizollah, N.B. Anuar, R. Salleh, F. Amalina, Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis, in: 2014 International Symposium on Biometrics and Security Technologies (ISBAST), IEEE, 2014, pp. 193–197.
[294] P.I. Radoglou-Grammatikis, P.G. Sarigiannidis, Flow anomaly based intrusion detection system for android mobile devices, in: 2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST), IEEE, 2017, pp. 1–4.
[295] Y. Pang, Z. Chen, X. Li, S. Wang, C. Zhao, L. Wang, K. Ji, Z. Li, Finding android malware trace from highly imbalanced network traffic, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Vol. 1, IEEE, 2017, pp. 588–595.
[296] L. Watkins, A.L. Kalathummarath, W.H. Robinson, Network-based detection of mobile malware exhibiting obfuscated or silent network behavior, in: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2018, pp. 1–4.
[297] S. Dai, T. Wei, W. Zou, Droidlogger: Reveal suspicious behavior of android applications via instrumentation, in: 2012 7th International Conference on Computing and Convergence Technology (ICCCT), IEEE, 2012, pp. 550–555.
[298] R. Zachariah, K. Akash, M.S. Yousef, A.M. Chacko, Adultswine: A case study, in: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2018, pp. 1345–1349.
[299] H.M. Jung, K. Kim, H. Cho, A study of android malware detection techniques in virtual environment, Cluster Comput. 19 (4) (2016) 2295–2304.
[300] S. Xu, X. Ma, Y. Liu, Q. Sheng, Malicious application dynamic detection in real-time API analysis, in: 2016 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2016, pp. 788–794.
[301] Z. Qu, S. Alam, Y. Chen, X. Zhou, W. Hong, R. Riley, Dydroid: Measuring dynamic code loading and its security implications in android applications, in: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2017, pp. 415–426.
[302] P. Faruki, A. Zemmari, M.S. Gaur, V. Laxmi, M. Conti, Mimeodroid: large scale dynamic app analysis on cloned devices via machine learning classifiers, in: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), IEEE, 2016, pp. 60–65.
[303] P. Borges, B. Sousa, L. Ferreira, F.B. Saghezchi, G. Mantas, J. Ribeiro, J. Rodriguez, L. Cordeiro, P. Simoes, Towards a hybrid intrusion detection system for android-based PPDR terminals, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, 2017, pp. 1034–1039.
[304] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, G. Wang, Malwarehouse: A data collection-as-a-service of mobile malware behavioral patterns, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), IEEE, 2018, pp. 1503–1508.
[305] J. Milosevic, A. Ferrante, M. Malek, Malaware: Effective and efficient run-time mobile malware detector, in: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, 2016, pp. 270–277.
[306] A. Merlo, M. Migliardi, P. Fontanelli, On energy-based profiling of malware in android, in: 2014 International Conference on High Performance Computing & Simulation (HPCS), IEEE, 2014, pp. 535–542.
[307] M.W. Afridi, T. Ali, T. Alghamdi, T. Ali, M. Yasar, Android application behavioral analysis through intent monitoring, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2018, pp. 1–8.
[308] S. Ma, Z. Tang, Q. Xiao, J. Liu, T.T. Duong, X. Lin, H. Zhu, Detecting GPS information leakage in android applications, in: 2013 IEEE Global Communications Conference (GLOBECOM), IEEE, 2013, pp. 826–831.
[309] J. Brown, M. Anwar, G. Dozier, Detection of mobile malware: an artificial immunity approach, in: 2016 IEEE Security and Privacy Workshops (SPW), IEEE, 2016, pp. 74–80.

[310] W. Fan, Y. Sang, D. Zhang, R. Sun, Y. Liu, DroidInjector: A process injection-based dynamic tracking system for runtime behaviors of android applications, Comput. Secur. 70 (2017) 224–237.
[311] P. Berthome, T. Fecherolle, N. Guilloteau, J. Lalande, Repackaging android applications for auditing access to private data, in: 2012 Seventh International Conference on Availability, Reliability and Security, IEEE, 2012, pp. 388–396.
[312] G. Peng, Y. Shao, T. Wang, X. Zhan, H. Zhang, Research on android malware detection and interception based on behavior monitorin, Wuhan Univ. J. Nat. Sci. 17 (5) (2012) 421–427.
[313] G. D'Angelo, M. Ficco, F. Palmieri, Malware detection in mobile environments based on autoencoders and API-images, J. Parallel Distrib. Comput. 137 (2020) 26–33.
[314] A. Mahindru, P. Singh, Dynamic permissions based android malware detection using machine learning techniques, in: Proceedings of the 10th Innovations in Software Engineering Conference, ACM, 2017, pp. 202–210.
[315] S. Yue, W. Feng, J. Ma, Y. Jiang, X. Tao, C. Xu, J. Lu, Repdroid: an automated tool for android application repackaging detection, in: 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC), IEEE, 2017, pp. 132–142.
[316] Y. Cuixia, Z. Chaoshun, G. Shanqing, H. Chengyu, C. Lizhen, Ui ripping in android: Reverse engineering of graphical user interfaces and its application, in: 2015 IEEE Conference on Collaboration and Internet Computing (CIC), IEEE, 2015, pp. 160–167.
[317] C. Soh, H.B.K. Tan, Y.L. Arnatovich, L. Wang, Detecting clones in android applications through analyzing user interfaces, in: 2015 IEEE 23rd International Conference on Program Comprehension, IEEE, 2015, pp. 163–173.
[318] W. Wu, S. Hung, DroidDolphin: a dynamic Android malware detection framework using big data and machine learning, in: Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, 2014, pp. 247-252.
[319] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, Improving dynamic analysis of android apps using hybrid test input generation, in: Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2017, pp. 1–8.
[320] Paul McNeil, Sachin Shetty, Divya Guntu, Gauree Barve, SCREDENT: Scalable real-time anomalies detection and notification of targeted malware in mobile devices, Procedia Comput. Sci. 83 (2016) 1219–1225.
[321] L. Chen, S. Hou, Y. Ye, S. Xu, Droideye: Fortifying security of learning-based lassifier against adversarial android malware attacks, in: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2018, pp. 782–789.
[322] S. Iqbal, M. Zulkernine, Spydroid: A framework for employing multiple real-time malware detectors on android, in: 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, 2018, pp. 1–8.
[323] I. Khokhlov, L. Reznik, Colluded applications vulnerabilities in android devices, in: 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, 2017, pp. 462–469.
[324] P. Feng, J. Ma, C. Sun, Selecting critical data flows in android applications for abnormal behavior detection, Mob. Inf. Syst. (2017).
[325] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, B. Yang, Machine learning based mobile malware detection using highly imbalanced network traffic, Inform. Sci. 433 (2018) 346–364.
[326] M. Yang, S. Wang, Z. Ling, Y. Liu, Z. Ni, Detection of malicious behavior in android apps through API calls and permission uses analysis, Concurr. Comput.: Pract. Exper. 29 (19) (2017).
[327] A.I. Aysan, F. Sakiz, S. Sen, Analysis of dynamic code updating in android with security perspective, IET Inf. Secur. 13 (3) (2018) 269–277.
[328] C. Sun, H. Zhang, S. Qin, N. He, J. Qin, H. Pan, Dexx: a double layer unpacking framework for android, IEEE Access 6 (2018) 61267–61276.
[329] M. Sharma, M. Chawla, J. Gajrani, A survey of android malware detection strategy and techniques, in: Proceedings of International Conference on ICT for Sustainable Development, Springer, 2016, pp. 39–51.
[330] B.A. Mantoo, S.S. Khurana, Static, Dynamic and intrinsic features based android malware detection using machine learning, in: Proceedings of ICRIC 2019, Springer, 2020, pp. 31–45.
[331] S. Zhao, X. Li, G. Xu, L. Zhang, Z. Feng, Attack tree based android malware detection with hybrid analysis, in: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2014, pp. 380–387.
[332] A. Martín, R.L. Cabrera, D. Camacho, Android malware detection through hybrid features fusion and ensemble classifiers: the andropytool framework and the omnidroid dataset, Inf. Fusion 52 (2019) 128–142.
[333] D. Saif, S.M. El-Gokhy, E. Sallam, Deep belief networks-based framework for malware detection in android systems, Alexandria Eng. J. 57 (4) (2018) 4049–4057.
[334] F. Xue, S. You, Z. Qi, H. Liu, A multidimensional feature extraction method based on android malware detection, in: International Conference on Signal and Information Processing, Networking and Computers, Springer, 2018, pp. 3–8.
[335] S. Rastogi, K. Bhushan, B.B. Gupta, Android applications repackaging detection techniques for smartphone devices, Procedia Comput. Sci. 78 (C) (2016) 26–32.
[336] H. Shahriar, V. Clincy, Detection of repackaged android malware, in: The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014), IEEE, 2014, pp. 349–354.
[337] Q. Chen, J. Wang, Y. Wang, An online approach for detecting repackaged android applications based on multi-user collaboration, in: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), IEEE, 2015, pp. 312–315.
[338] X. Sun, J. Han, H. Dai, Q. Li, An active android application repacking detection approach, in: 2018 10th International Conference on Communication Software and Networks (ICCSN), IEEE, 2018, pp. 493–496.
[339] X. Wu, D. Zhang, X. Su, W. Li, Detect repackaged android application based on http traffic similarity, Secur. Commun. Netw. 8 (13) (2015) 2257–2266.
[340] J. Kraunelis, X. Fu, W. Yu, W. Zhao, A framework for detecting and countering android UI attacks via inspection of IPC traffic, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.
[341] O. Mirzaei, J.M. Fuentes, J. Tapiador, L.G. Manzano, Androdet: An adaptive android obfuscation detector, Future Gener. Comput. Syst. 90 (2019) 240–261.
[342] I. Martín, J.A. Hernández, Clonespot: Fast detection of android repackages, Future Gener. Comput. Syst. 94 (2019) 740–748.
[343] F. Lyu, Y. Lin, J. Yang, J. Zhou, Suidroid: An efficient hardening-resilient approach to android app clone detection, in: 2016 IEEE Trustcom/BigDataSE/ISPA, IEEE, 2016, pp. 511–518.
[344] I. Gurulian, K. Markantonakis, L. Cavallaro, K. Mayes, Reprint of you can't touch this: Consumer-centric android application repackaging detection, Future Gener. Comput. Syst. 80 (2018) 537–545.
[345] J. Garcia, M. Hammad, S. Malek, Lightweight, obfuscation-resilient detection and family identification of android malware, in: 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), IEEE, 2018, p. 497.
[346] S. Badhani, S.K. Muttoo, Comparative analysis of pre-and post-classification ensemble methods for android malware detection, in: International Conference on Advances in Computing and Data Science, Springer, 2018, pp. 442–453.
[347] M.S. Rana, C. Gudla, A.H. Sung, Evaluating machine learning models for android malware detection: A comparison study, in: Proceedings of the 2018 VII International Conference on Network, Communication and Computing, 2018, pp. 17-21.
[348] N. Painter, B. Kadhiwala, Machine-learning-based android malware detection techniques—A comparative analysis, in: Information and Communication Technology for Sustainable Development, Springer, 2018, pp. 181–190.
[349] S. Chen, M. Xue, Z. Tang, L. Xu, 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, 2016, pp. 377-388.
[350] Anshul Arora, Sateesh K. Peddoju, Vikas Chouhan, Ajay Chaudhary, Hybrid Android malware detection by combining supervised and unsupervised learning, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, 2018, pp. 798-800.
[351] L. Apvrille, A. Apvrille, Identifying unknown android malware with feature extractions and classification techniques, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, IEEE, 2015, pp. 182–189.
[352] H. Fereidooni, M. Conti, D. Yao, A. Sperduti, ANASTASIA: Android malware detection using static analysis of applications, in: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE, 2016, pp. 1–5.
[353] J. Jung, J. Choi, S. Cho, S. Han, M. Park, Y. Hwang, Android malware detection using convolutional neural networks and data section images, in: Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, 2018, pp. 149-153.
[354] W. Wang, M. Zhao, J. Wang, Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network, J. Ambient Intell. Humaniz. Comput. 10 (8) (2019) 3035–3043.
[355] Y. Liu, G. Li, Z. Jin, Call graph based android malware detection with CNN, in: Software Engineering and Methodology for Emerging Domains, Springer, 2017, pp. 72–82.
[356] G. Bajwa, M. Fazeen, R. Dantu, S. Tanpure, Unintentional bugs to vulnerability mapping in android applications, in: 2015 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2015, pp. 176–178.

[357] H. Papadopoulos, N. Georgiou, C. Eliades, A. Konstantinidis, Android malware detection with unbiased confidence guarantees, Neurocomputing 280 (2018) 3–12.
[358] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, Emulator vs real phone: Android malware detection using machine learning, in: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics, 2017, pp. 65-72.
[359] C. Zhao, W. Zheng, L. Gong, M. Zhang, C. Wang, Quick and accurate android malware detection based on sensitive APIs, in: 2018 IEEE International Conference on Smart Internet of Things (SmartIoT), IEEE, 2018, pp. 143–148.
[360] M. Amin, B. Shah, A. Sharif, T. Ali, K. Kim, S. Anwar, Android malware detection through generative adversarial networks, Trans. Emerg. Telecommun. Technol. (2019).
[361] M. Shahpasand, L. Hamey, D. Vatsalan, M. Xue, Adversarial attacks on mobile malware detection, in: 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile), IEEE, 2019, pp. 17–20.
[362] S.Y. Yerima, S. Khan, Longitudinal performance analysis of machine learning based android malware detectors, in: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2019, pp. 1–8.
[363] K. Xu, Y. Li, R.H. Deng, K. Chen, Deeprefiner: Multi-layer android malware detection system applying deep neural networks, in: 2018 IEEE European Symposium on Security and Privacy (EuroS & P), IEEE, 2018, pp. 473–487.
[364] P. Kaushik, P.K. Yadav, A novel approach for detecting malware in android applications using deep learning, in: 2018 Eleventh International Conference on Contemporary Computing (IC3), IEEE, 2018, pp. 1–4.
[365] G. Canfora, A.D. Lorenzo, E. Medvet, F. Mercaldo, C.A. Visaggio, Effectiveness of opcode ngrams for detection of multi family android malware, in: 2015 10th International Conference on Availability, Reliability and Security, IEEE, 2015, pp. 333–340.
[366] P. Gronát, J.A. Aldana-Iuit, M. Bálek, Maxnet: Neural network architecture for continuous detection of malicious activity, in: 2019 IEEE Security and Privacy Workshops (SPW), IEEE, 2019, pp. 28–35.
[367] R. Graf, L.A. Kaplan, R. King, Neural network-based technique for android smartphone applications classification, in: 2019 11th International Conference on Cyber Conflict (CyCon), Vol. 900, IEEE, 2019, pp. 1–17.
[368] G. Dai, J. Ge, M. Cai, D. Xu, W. Li, Svm-based malware detection for android applications. in: Proceedings of the 8th ACM conference on security & privacy in wireless and mobile networks, 2015, pp. 1-2.
[369] H. Ham, H. Kim, M. Kim, M. Choi, Linear SVM-based android malware detection, in: Frontier and Innovation in Future Computing and Communications, Springer, pp. 575-585.
[370] B. Rashidi, C. Fung, E. Bertino, Android malicious application detection using support vector machine and active learning, in: 2017 13th International Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–9.
[371] J. Sahs, L. Khan, A machine learning approach to android malware detection, in: 2012 European Intelligence and Security Informatics Conference, IEEE, 2012, pp. 141–147.
[372] H. Chuang, S. Wang, Machine learning based hybrid behavior models for android malware analysis, in: 2015 IEEE International Conference on Software Quality, Reliability and Security, IEEE, 2015, pp. 201–206.
[373] J. Du, H. Chen, W. Zhon, Z. Liu, A. Xu, A dynamic and static combined android malicious code detection model based on SVM, in: 2018 5th International Conference on Systems and Informatics (ICSAI), IEEE, 2018, pp. 801–806.
[374] S. Türker, A.B. Can, Andmfc: Android malware family classification framework, in: 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), IEEE, 2019, pp. 1–6.
[375] M. Grace, Y. Zhou, Q. Zhang, S. Zou, X. Jiang, Riskranker: scalable and accurate zero-day android malware detection, in: Proceedings of the 10th international conference on Mobile systems, applications, and services, 2012, pp. 281-294.
[376] L. Ma, Y. Yang, X. Wang, J. He, Ultra-lightweight malware detection of android using 2-level machine learning, in: 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), IEEE, 2016, pp. 729–733.
[377] S. Alam, Z. Qu, R. Riley, Y. Chen, V. Rastogi, Droidnative: Automating and optimizing detection of android native code malware variants, Comput. Secur. 65 (2017) 230–246.
[378] A. Feizollah, N.B. Anuar, R. Salleh, A.W.A. Wahab, A review on feature selection in mobile malware detection, Digit. Investig. 13 (2015) 22–37.
[379] M.S. Rana, C. Gudla, A.H. Sung, Evaluating machine learning models on the ethereum blockchain for android malware detection, in: Intelligent Computing-Proceedings of the Computing Conference, Springer, 2019, pp. 446–461.
[380] Z. Shan, I. Neamtiu, R. Samuel, Self-hiding behavior in android apps: detection and characterization, in: Proceedings of the 40th International Conference on Software Engineering, IEEE, 2018, pp. 728–739.
[381] E.B. Karbab, M. Debbabi, D. Mouheb, Fingerprinting android packaging: Generating DNAs for malware detection, Digit. Investig. 18 (2016) 33–45.
[382] S. Badhani, S.K. Muttoo, Cendroid—A cluster-ensemble classifier for detecting malicious android applications, Comput. Secur. 85 (2019) 25–40.
[383] M. Rahman, Droidmln: A markov logic network approach to detect android malware, in: 2013 12th International Conference on Machine Learning and Applications, Vol. 2, IEEE, 2013, pp. 166–169.
[384] A.F. Abdul Kadir, N. Stakhanova, A.A. Ghorbani, Android botnets: What urls are telling us, in: International Conference on Network and System Security, Springer, Cham, 2015, pp. 78–91.
[385] H. Huang, C. Zheng, J. Zeng, W. Zhou, S. Zhu, P. Liu, S. Chari, C. Zhang, Android malware development on public malware scanning platforms: A large-scale data-driven study, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE, 2016, pp. 1090–1099.
[386] L. Taheri, A.F. Abdul Kadir, Arash.Habibi. Lashkari, Extensible android malware detection and family classification using network-flows and api-calls, in: 2019 International Carnahan Conference on Security Technology (ICCST), IEEE, 2019, pp. 1–8.
[387] Ying Pang, Lizhi Peng, Zhenxiang Chen, Bo Yang, Hongli Zhang, Imbalanced learning based on adaptive weighting and Gaussian function synthesizing with an application on android malware detection, Inform. Sci. 484 (2019) 95–112.