International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# Impact of Permission in Android on Data Security

**Irum Ashraf**

Lecturer, Grand Asian University, Sialkot

**Abstract**

Android is an application platform for mobile devices with an advance and moderate frequent feature of operating systems. It includes all software systems, software package frameworks, and core programs. This platformpermits to collect and verify important personal information concerning the user/client through untrusted apps. However, to install an application, the device feature uses permissions that are allowed by the user. The user has an attribute to research permissions and aborts the setup if the permissions are unfriendly or unrestricted.Android permission analysis schemes play a major and important role in securing important information and privacy of usersfrom unbearablebehaviors of untrusted android permission apps within the aspect of securing purposes. This survey tries to handle and deal with the android permission application permissions related to security and privacy challenges. It includes various research articles published in computers and security, digital investigation, decision support systems, systems and software security, and information forensics journals in the last 10 years. The survey includes the subsequent prior considerations, analysis problems motivated by the theme, the methodology used, the ability of result analysis conducted, and mechanical man options thought-about for performance analysis.

**Keywords**—Android; Permissions; Signature-dynamic and static, Data Security

## 1- INTRODUCTION

The increasing use of smartphones during the 21st century has inspired the mobile industry for regeneration[1]. Powar and Meshram[2]and Felt et al,[3] has updated in their previous work report that robot mobile framework has gone through a change by the manufacturing trade, customers, and therefore the software system extension community. Andriod market (Google Play) reported in 2015 that over 1.5.5 million users downloaded the frequent different apps,[4]and became over a billion Andriod customers. Android is the most used mobile device operating system (OS) of today's world due to its recurring use.[5]. One of the primary reasons for its rise is its availability as a freewareOS.The earlier Android gadgets didn't accept security patches, which resultsin less range of users, however, consecutive Improvements [6, 7] have solved the matter. Android is the preferred OS at the moment. Andriod has modified our daily life as it's interconnected with other applications in a wide selection of gadgets, like smartphones televisions and other CCTV cameras. There are over 2.5 billion active Android users in the play store [8]. Google has givenan open-source license to an android so the user can download any application, and therefore the available source code is the reason behind the popularity of android. Android is well known for mobile organizations that require an immediate and modified OS for their devices.

Mobile device security has an important role in today's life, but it also should be user friendly. However, it ensuressecuringimportant informationaboutthe client/user by protectingits privacy. The mobile device companies are concerned regarding a user's security and data privacy on which they are working

promptly as much as they can. The internet has created multiple issues for clients for securing user's data privacy that includes threats and vulnerabilities.[9, 10] from affected phones to premium-rate phone numbers without even knowledge of the user/client. Because of this, the user's data from mobile is being sent to unauthorized third parties without any permission fromthe user.[11, 12]. This researchadds the previous reviews and work by extending the scope of malware/vulnerabilities development and Android security issues. The permissions given by the defined smartphone are checked and compared, and these permissions are granted accordingly. This work consists of a detailed literature survey. Sections 2 and 3 discuss the background literature of permission analysis and its implementation in android. In Section 4, attacks are presented through permission techniques, and consequently, Section 5 deals with the methodology of permission analysis. Section 6 analyzes the approach for the literature review. In Sections 7 and 8, exclusion criteria and data extraction are studied, respectively. Section 9 discusses future research directions, and the last section finally concludes this paper.

**Literature Survey :**

Today Mobile Technology is being largely used in the world for different purposes which have advantages and disadvantages both[13]. Since 2008 the usage of mobile technology has beendrastically increasedday by day includingimportant and personal data like a gallery, videos, debit card details, SMS messages, WhatsApp chat history, etc can be easily stored in the mobile memory[14].In the market,there are a variety of smartphones with different and moderate features of advanced operating systems. Google has given freeware license to the Android mobile operating system available in many smartphones through which they can easily install multiple apps. According to Google's information 1.3 million, Android devices are activated daily for different various purposes, but securing its privacy is an important and needed issue of today's[15]. Gartner has reported in his prior that Google's Android has successfully grabbed 82% of the market during2016[16]. In 2016 total of 432 million mobile deviceshas been sold out from which 352 million mobiles were using advanced and moderate Android operating system features[17].With the great success and triumph of the Android operating system, it is also dragged and diverted to vulnerabilities and malicious/malware attacks of the third party by getting the access of privacy permission without an acknowledgment of user which becomes a more concerned area of different big organizations and companies. According to the Google Android Security Report, a total of 655 abrasive vulnerabilities/malicious were found in 2016 which has affected the daily life of users [18]. A total of 316 vulnerabilities were found in the Android operating system in 2017 which is highercompared to any other operating system in smartphones [19]. According to Cisco's report 98% of the malware target the Android platform[13]. An increasing malware attacks day by day there should be a permission access application to secure client/user privacy data

The venerable/malware attacks in any application either installed by the user or installed on another mobile without the user's permission are performing various functions without the user's knowledge[18]. The main purpose of the third party/malware attacks is to access thepersonal data from the mobile devices, unlocking the mobile devices, sending and receiving SMS/MMS forknowledge ofconfidential information, making calls on the client's numbers and his friends for threat purposes, share the confidential important information through GPS[13, 14]. According to the previous studies, there are different researches carried out that characterize various existing.

According to previous research,there are also undertaken to characterize existing Android malware. A project named Android Malware Genome was undertaken to characterize existing Android malware[20].

According to previous research in 2022 the security impact of vendor customisations on Android devices by designing SEFA analysis framework.. This tool performs several

Android Malware can be categorized into different categories based on performing functions [21, 22]like Spyware, Trojans, Virus, Phishing Apps, Bot Process, RootKits. As per prior research, the table represents a list of top 10 Android Malware Families with their descriptions and capabilities [23-25]. Here top 10 Android Malware Families are taken to study the most recent famous malware.

| No | Malware Family | Description | Capability |
|---|---|---|---|
| 1 | FakeInst | Send premium SMS Messages | Send SMS |
| 2 | fake | Send premium SMS Messages | Send SMS |
| 3 | SNDApps | Steals various information like device Id, Email Id, Address and Phone no from the device and uploads the information to a remote server | Information Stealing |
| 4 | Boxer | Send premium SMS Messages | Send SMS |
| 5 | GinMaster | Steals confidential information from the device and sends it to a remote server | Information Stealing |
| 6 | VDLoader | Steals personal information | Root access and Information Stealing |
| 7 | FakeDolphin | Gives you dolphin browser and signs up a user for the services without their knowledge | Information Stealing |
| 8 | KungFu | It steals the information like IMEI, device, OS version and dumps into a local file that is sent to the remote server | Root Access, Botnet and Information Stealing |
| 9 | Bracebridge | It sends the confidential details like IMEI, SMS, IMSI to a remote server | Botnet and Information Stealing |
| 10 | JIFake | Send premium SMS Messages | Send SMS |

**Malware Families in Previous years**

## 2- Background Informationin Permission Analysis :

Static and dynamic analysis are two various types to detect malware. Somesignificant features are being removed from an app during static detectionand analyzed before the app is executed,[26, 27] while in case of dynamic detection which is quite opposite to the static wherethe app is executed throughthe simulator and decided accordingly to the log files.[28, 29] Both types have some advantages and disadvantages as briefedin Table 1.

## 2.1 Static analysis

We candetect malicious/venerablebehaviorsthrough code segments by static analysis.Although it needsa minimum duration period and resources as it does not get indulge in the execution of the applicationcompared to other analysis techniques for Android malware/malicious detection.[30, 31].By using this technique, we can withdraw some important features without installing the application on a device or emulator. However, this technique has an indicative downside of code obfuscation and dynamic code loading. Code obfuscation makes the pattern matching unable of detecting unpredictably. On the hand side, the advantages of static analysis are that it can detect possible security violations, runtime errors, and logical inconsistencies [32, 33] The permissions of API calls are usually used as static features as shown in Figure 1. The two main approaches for static analysis are signature-based and permission-based.

### 2.1.1Signature-based approach

This technique is also known as the misuse detection technique. The commercial anti-malware products are mostly used for the signature-based malware detection approaches. This technique gives an antique signature and removes all semantic patterns and formats. Signature-based recognition is very beneficial for the already known malware, but the biggest problem and disadvantage of this approach are that it cannot identify the unknown malware types and its source. Although the maximum malware sourcesremain undetected through this

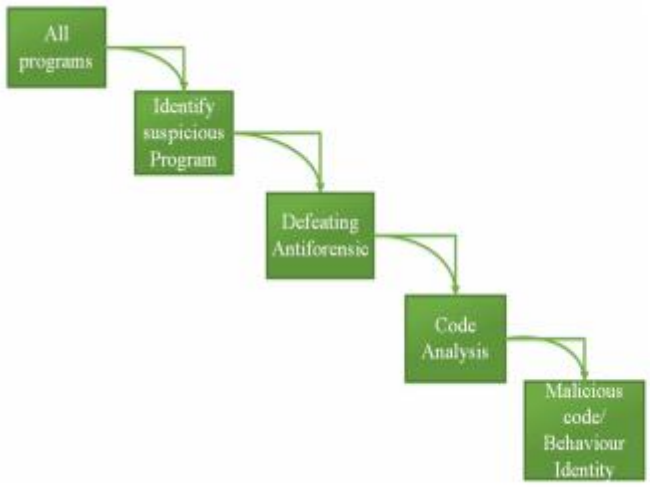



**FIGURE 1** Static analysis process

due to the incomplete signature database, while we have detected the malware, its variants are required to be instantly updated.[34, 35]

| Components | Static analysis | Dynamic analysis |
|---|---|---|
| **Target code execution** | Not possible | Possible |
| **Time required** | minimum | maximum |
| **Benefits** | Minimum cost and Minimum time required. | Gives a profoundexamination and higher discovery rate with obscure malware location |
| **Disadvantages** | Constrained signature database and can identify within the scope of only known malware types. | Power consumption and maximum time |
| **Code obfuscation** | yes | no |
| **Input** | Binary files, scripting, language files, etc | Memory snapshots, runtime, etc API data |

**Table 1. Comparison of Dynamic and static analysis.**

### 2.1.2 Permission-based analysis

The permission-based analysis is required where permission requested by an application/user showsa major role in accessing rights.Permission is requested by the applications in their manifest file. Android permission overwhelms the required access to the application data. The data stored in the mobile cannot be accessed without the permission of the user/client. An important file is present in the root directory of AndroidManifest.XML, which stores all information related to the application of the Android

system.[36, 37]. The permission process should be working effectively and smoothly before allowing the application to get the required asset.Specifically, agreement for any defined particular application in any case agreements for every single proclaimed permission is not required. It just verifies the show document and noof different records.[38, 39].

### 2.1.3Resources Based Static Analysis

Resources used in the application arestored in aresource file named AndroidManifest.xml file which contains all requiredinformation [14]. User interfaces modules hold resources of an application like menus, layout widgets,etc [14]. The AndroidManifest.xml file is present in the APK file. User interaction from these user interfaces is needed in malware running in the background[14]. So the user interfaces ll'be completely and properly analyzed.

### 2.1.4 Components Based Static Analysis

There are several componentsof AndroidApplications like Content Providers, Services, Intents, Activities, and Broadcast Receivers [14]. Information related to these components is stored in the AndroidManifest.xmlfile. At the background services, there are some malware running, gaining information about activities, Receivers,and intents.[14]. To analyze these components is also very much important for detecting malicious behavior and its sources

## 3- Dynamic analysis

By using dynamic analysis, we can test and evaluate data in real-time. The main purpose of this analysis is to repeatedly examine offline code and to detect the run-time error.By executing the application, we can evaluate the malware during dynamic analysis.[40, 41].As we compare dynamic analysiswith static analysis, this approach is more baffled and intricate as resource action is performed in the real environment. This analysis could be beneficial inloadingthe required information to determine the application behavior during runtime. By using a dynamic behavioral detection method many machines like Sandbox, virtual machine, and other forms of operation environment are constructed. By stimulating the execution of the application, we can obtain an application behavior model [42, 43]. There are two main approaches for dynamic analysis: anomaly-based and taint based.

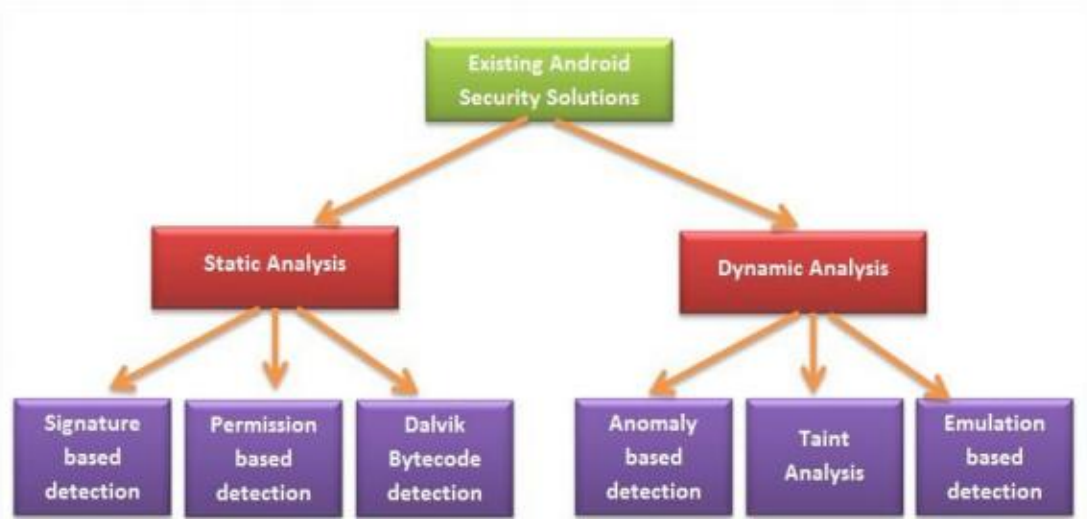
### 3.1 Anomaly-based detection

To detect malicious behavior, the function of this approach is reliable on the machine learning algorithm. By using features from the existing malware to train the model that predicts unknown malware. It requires a lot of effort and resource Applications (apps) installedfor identifying the malicious behavior in the system to perform an in-depth analysis of malware detection. The biggest disadvantage of this process is that, if it requires more system calls, then it classifies for the legitimate application.[44]

### 3.2 Taint analysis

Taint analysis checks the user's input and modifies the different variables accordingly.The only reason behind being one of the most used methods is that it focuses on those appsonly that are shared to get sensible important information. Dynamic taint analysis is a scientific technique that is used for this approach followed by Taint Droid. This technique is called taint, which marks the data of interest with an identifier. When information is being used, the taint stays with this information [45]. The Train Droid

provides a system-wide data stream for tracking Android.The multifeatured sensitive data like headphones, high pixelcamera, and GPS can also be tracked by this method. Taint Droid isused for tagging important and sensitive data and information. Through any channel, Taint Droidcan record the tagged data from the system.[46, 47] The major withdraw of this system is that it cannot down track data that leaves and returns the channel.




### 2.3 Call graph construction

Established algorithmsignore open-package assumption utilization situations, which is the reason that the expansion code is not considered while separating remain private application. Therefore, the expansion code can deal and reply direct call conditions between library strategies that are not obvious from the class chain [48] The abbreviated call graphs for permission analysis is depicted in Figure 2. The UI string transitively calls toString() on every instance of painting JList. AbstractMap.toString() is called by giving the custom HashMap usage as substance, which does no supplant toString(). This method repeats over the passage set and calls getValue() in every section. The essential thing of the section set is the aggressor's link case. Like this, it effectively calls Expression.GetValue() that brilliantly summons System. Set SecurityManager (null). To methodically discover exploitable callback executions, a static exam should watch that there is no assailant callable strategy present, which transitively calls sensitive support without appropriate cleaning or permission checks. In any case, the static examination desires to do furthermore not forget that calls to callbacks are made plans to all workable put stock in executions. Cutting aspect name-chart calculations exclude a call part from name locales of Entry. GetValue() to the method Expression.GetValue(). However, this part is needed to find out about the attack, which should be exhibited in the right ways. If this edge is incorporated, information flowtesting and searching for unguarded ways to delicate activities are empowered to distinguish the vulnerability.[49]

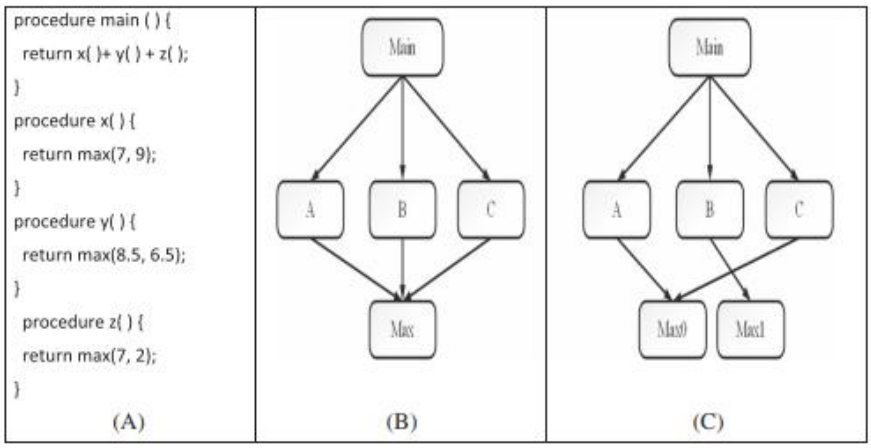



**FIGURE 2** Call graphs for permission analysis ((A) example program, (B) context-insensitive, and (C) context-sensitive)[21]

## 3- PERMISSION IMPLEMENTATION IN ANDROID APPLICATIONS

In the Android software program stack in which the Android framework degree, within the Android framework stage, a large portion of the contents are authorized. By using a supplementary series ID, some kind of permissions isdecided at the kernel level. It is allowed to use a supplementary organization ID if the application is part of a group to obtain all of the privileges of that particular institution[50, 51]. By using the permissions declared in the programs that take place to file, the organizations are arranged. Figure 3 illustrates that the Bluetooth, digitalcamera, and the Internet which need permissions canbe decided atthe kernel level. Communication list enforced inside the Android framework stage by group ID and other permissions such as SMS and MMS that are used in retrieving [52, 53]

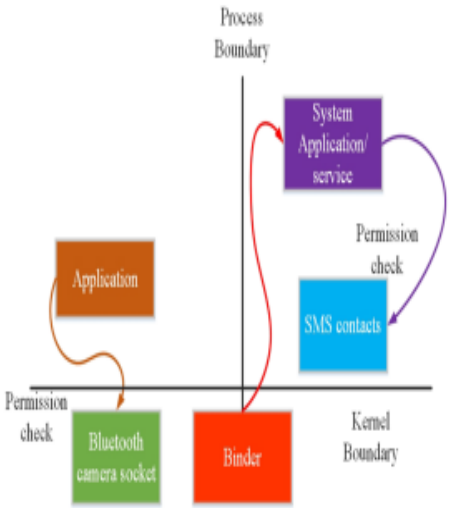



**FIGURE 3** Permission enforcement in Android application

## 4 -ATTACK OF PERMISSION ESCALATION

The permission attacks are categorized and elaborated in Figure 4. There are two types of permission escalation attack Confused deputy attack and collision attack. Privileged benign applications are in unprotected interfacesthatexploit and declare the vulnerabilities/malware by confused deputy attack. [54]. Tobuild a combined set of permissions, the collusion attack can be admitted by multiple applications. This allows us to represent malicious/malware actions by an unauthorized performance

[10]. There are two categories of collusion attacks (1)direct collision attack (2) indirect collusion attack. These two categories are used for interconnecting with each other. By using direct collusionway communication applications can be performed indirectly.Another category indirect collision attack is used where the application communicates with the third application.[55]. Certain information that is like files, buffers, and input-output devices is holdingby overt channels. Shared preferences, UNIX socket communication, and system logs are few other examples of overt channels.[56, 57]. The covert channels are used objectsforcommunication which aremostly unplanned. These objects are applied for covert channels with the Android's middleware layer bypass. At last, it is observed that the low throughput of a covert channel is suitable to exchange private data and communication[38, 58]

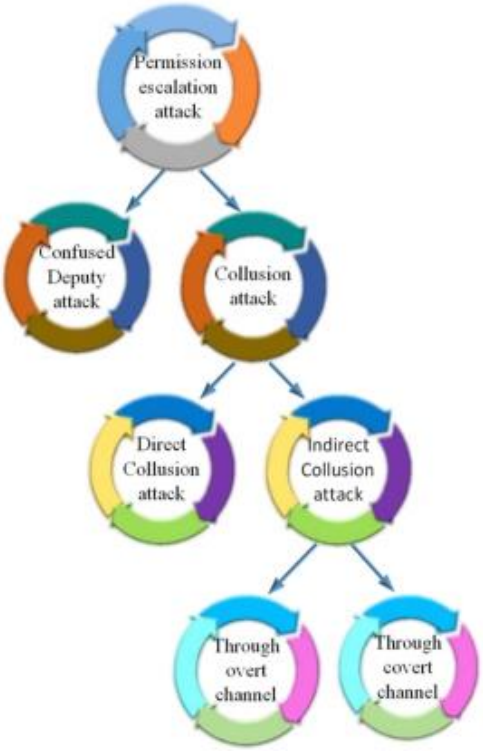



**FIGURE 4** Permission escalation attack

## 5- MALICIOUS APPLICATION DETECTION USING ANDROID PERMISSIONS

### 5.1 Static analysis-based approaches

Some major static analysis–based approaches from 2007 to 2019 are mentioned in Table 2. In 2014, Fang et al64 has investigated the various android security issues developed by permission-based mechanisms. They have also reviewed the workshop measurementon such issues based on their technical features.Android framework which is flexible fine-grained permission models can be improved by developing data-driven methods for strengthening the android securityrather than irrespective of the current model that is based on coarse-grained and inflexible permission models, Talha et al[59] has proposed a permission-based Android malware detection system called APK Auditor to maintain the consistency between application intentions and system implementations. The system uses a static analysis technique for classifying android applications into benign or malicious. This system consists of three different parts, 1-a signature database, 2-an android client, 3-a central server. The signature database stores to execute data on applications as well as analytical results, whereas the android client is used by the end-users to give application analysis requests. The central server manages the whole

analysis process by communicating with the database and the android user. Song et al[60]has integrated the static detection method with the analysis framework where only the static detection method can result in a high false rate and also the scope is limited. The proposed approach consists of four filtering layers, namely, the message digest values, combining malicious permissions, the dangerous permissions, and dangerous intentions. Rashidi et al[61] has proposed RecDroid that provides a user-help-user environment for the android permission control. It is a crowdsourcing recommendation framework that combines the expert users' responses and then recommends it to the inexperienced and untrained users. Seshagiri et a[62]has proposed a static approach, namely, Amrita Malware Analyzer. This framework detects the malicious code by performing plaintext attacks using strings contained in the malicious web pages. Sokolova et al[63] has proposed a five-step methodology for finding the patterns of each category. The category patterns and key permissions are found using graph analysis metrics by modeling the required permissions as a graph.

| No | Research paper and year | Application |
|---|---|---|
| **1** | Balzarotti et al[64]and 2007 | Analysis of web-based applications. |
| **2** | Basin et al[65]and 2009 | Automated analysis of security and design model |
| **3** | Stolpe[66] and 2010 derogation | Permission-based on the notion. |
| **4** | Jeon et al[67]and 2012 | Fine-grained permissions in Android applications |
| **5** | Zhou Jiang[68] and 2012 | To identify certainly malware application |
| **6** | Zhang et al[69] and 2013 | Screening undesirable behaviors in android apps |
| **7** | Fang et al[70] and 2014 | To refer the concerns in android security |
| **8** | Talha et aI[59]and 2015 | Static analysis to portray method characterize Android applications |
| **9** | Song et Rashidi et aI[61] and 2016 | Real-time expert recommendations Incorporated static location an examination system for Android |
| **10** | Rashidi et al[61]Real-time and 2016 | Rashidi et al67 Real-time expert recommendations and 2016 |
| **11** | Rashidi et al[61] and 2016 | Real-time expert and 2016 |
| **12** | Sokolova et aI[63] and 2017 | Sokolova et al69 Android application classification and 2017 anomaly detection |
| **13** | Li et al[71] and 2018 | Significant Permission IDentification (SigPID), a malware detection system based on permission usage analysis to cope |
| **14** | Liu et al[72]and 2019 | Alde that influences the Xposed framework to accomplish analytics libraries in other apps |

**TABLE 2 Analysis method based on static analysis**

### 5.2 Dynamic analysis–based approaches

Some major dynamic analysis–based approaches from 2007 to 2019 are mentioned in Table 3. In 2013, Zhang et al[69] has proposed a dynamic analysis platform, namely, VetDroid. The VetDroid is used to have permission to use behaviors by identifying the valid authentication permission and Invalid authentication permission through verification of correct permission information. It is also used in detecting privacy leak, analyzing fine-grained causes of data leaks, and detecting susceptibility in regular applications. Min and Cao[73] have proposed a runtime-based behavior analysis system for detecting the android malware and venerability by privacy leak. According to research done, the traditional method, ie, the signature-based method is not good enough for malware detection. During the

research 350 applications have been analyzed from a third party. Petsas et al[74] has investigated the anti-analysis techniques that can be used by malware rather than using dynamic analysis approaches. These techniques are based on three different categories, viz, static properties, dynamic heuristics, and VM-related complexities of the Android emulator. Further, they proposed some countermeasures like modifying emulator, accurate binary translation, and hardware-assisted virtualization for improving dynamic analysis resistance against VM detection evasion. Abah et al[75] has proposed a device monitoring system for an unrooted device. The system is used for collecting application data that is then used to feature vectors. These feature vectors briefed the behavior of application for detecting the malware/malicious. Ab Razak et al[76] has developed a behavior-based anomaly detection system to detect the deviation in the application's network behavior. The system is used to control network traffic by monitoring suspicious network activities. The semi-supervised machine learning techniques are used for learning the normal behavior of the application. The system is used to detect mobile malware that cannot be identified by a signature approach or by static or dynamic analysis method. Thanigaivelan et al[77] have proposed a context-based dynamically reconfigurable access control system (CoDRA). The code uses feature-based policies that control resource access and policy granularity. The policy enforcement wasimplemented by combining application behavior and resource features. The code uses static as well as dynamic constraints, unlike the tradition that uses only static constraints on application activities.

| | | |
|---|---|---|
| **1** | Centonze et al[78] and 2007 | Automatic identification of precise access control policies |
| | Zand and Ahmadian[79] and 2009 | Application of homotopy analysis |
| **2** | Blaschke[80] and Image analysis for remote sensing | Blaschke74 and Image analysis for remote sensing |
| **3** | Isohara et al[81] and 2011 | Android malware detection based on kernel |
| **4** | Min and Cao[73] and 2012 | Malware detection in Android by runtime based behavior analysis of dynamic |
| **5** | Amos et al[82]and 2013 | Android malware identification by applying method machine learning classifiers |
| **6** | Petsas et al[83] and 2014 | Hinder dynamic analysis of Android malware |
| **7** | Abah et al[75] and 2015 | Anomaly-based malware detection |
| **8** | Ab Razak et al[76] And 2016 | Application network behavior by the web application |
| **9** | Idrees et al[84] and 2017 | Permission- and intent-based framework for identifying Android malware apps |
| **10** | hanigaivelan et al[77] and 2018 | Context-based dynamically reconfigurable access control system for Android |
| **11** | hmad et al[14] StaDART and 2019 | analyzes the arguments of reflection APIs without framework modification |

**TABLE 3 Analysis method based on dynamic analysis**

**Methodology**

In figure 5 shows the process of the Literature survey Called SLS (Systematic Literature Survey) this survey is completed with following

- In the beginning, we identify the research questions and explore the topic which is covered in SLS and try to diagnose the relevant data

- In the 2nd step, we try to identify the keywords and search them which help us to find the most Important journals inside the magnitude of this SLS
- The searching mechanism operates by two elements the first element is based on prominent generation storage facilities. 2nd one highlights the methods of separation from leading perceptions which has two parts diaries and meeting paper.
- Expulsion stander is applied to the listed items to confine our analyses to exceptionally relevant papers, in this way sifting through papers of likely restricted intrigue
- consequently, results from both the tests merge in the list of publications to survey. Conclusively, future research is discussed.

**Research questions**

RQ1.Identify the significant risk levels for permissions while considering the repetition of an event in spiteful and typical applications?

Ans - Few Android sanctions have been officially given by Google. To characterize, a threat level is an essential part of the Android permission. The danger level depends on the event of permission in the malware and programming sets. To assimilate spiteful activities, the risk level should be considered to be.

RQ2. In which situation a profile Android application be applying dangers provided acceptance on resources?
Android integrates the chance to gather permission as indicated by the benefit to get to and to a specific station to capture messages being received by the client or to send messages for the benefit of the client the permission that enables the messages to the client. This gathering also contains permissions, which can be utilized to profit without their immediate association. In any case, amid the establishment, it is just the gathering, which is shown to the client, which implies that the client awards consents, for example, READ_SMS and others that are obscure.

RQ3. How machine learning and Android profiles help in malware detection?

Machine learning-based frameworks using Android profiles help in designing Android tests that categorize the rules as typical or malicious.

RQ4. What will be the security concerns?
During the Analysis, two security features have a concern. Some broad measurements and possible smaller parts are collected for investigation furthermore security issues presented with best possible solutions

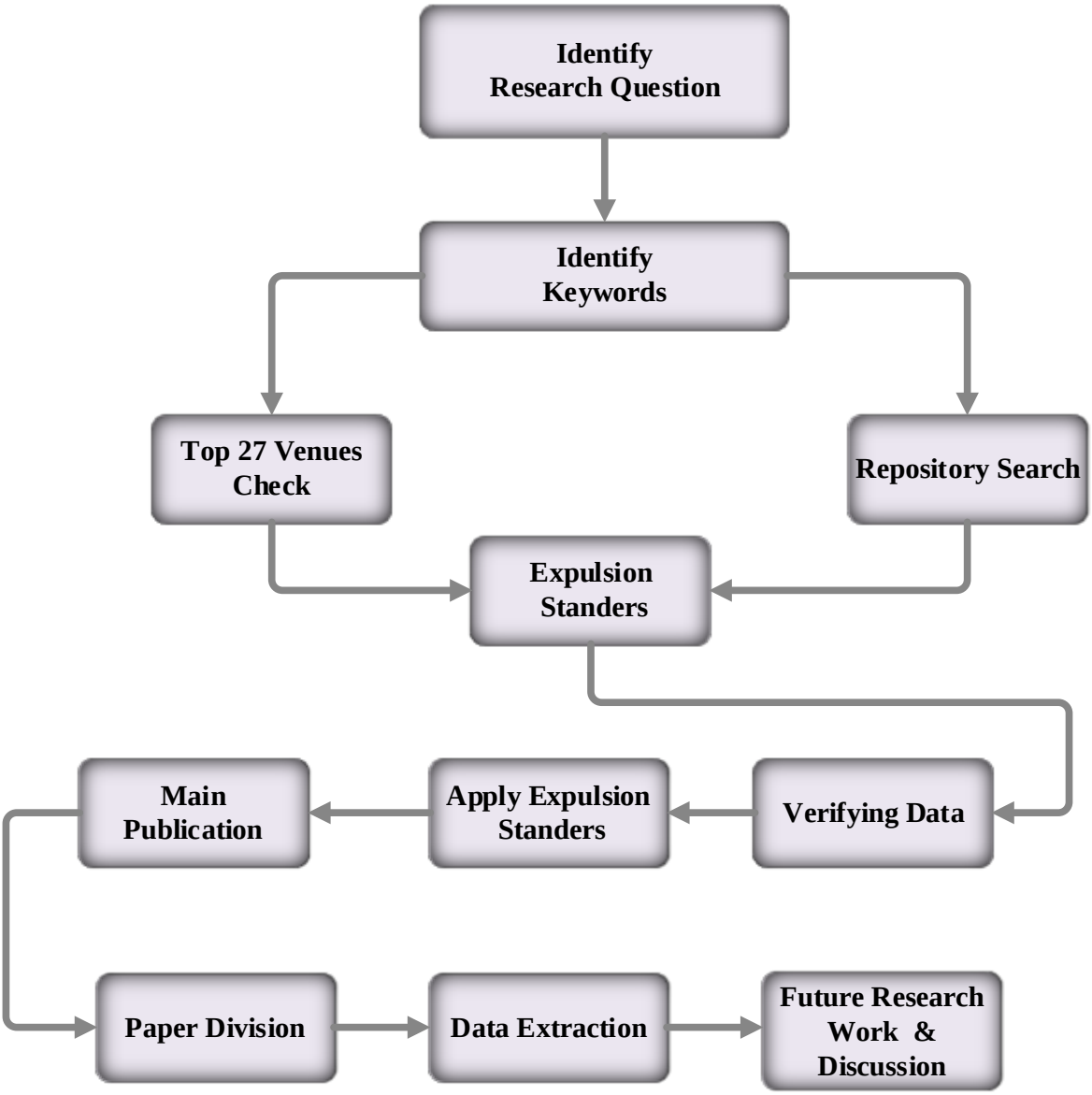



**FIGURE 5 Methodology**

## Search strategy

In the research strategy, keywords and data sets are used for finding the most suitable publications.

**1) Search keywords**

Analysis projects are formed by using search keywords, key aspects of permission analysis, and key aspects of static and dynamic analysis, which are used in our analysis. These search terms are shown in Table 4

***2)* Search data sets**

Data search is based on the storeroom and is additive by check beside top venues in Security. For collecting the substantial productions, a repository search is proposed and the best scene check applies just as the Verifying data process.

**3) Repository search**

Science Direct, Springer Link, IEEE Xplore Digital Library Web of Knowledge, Wiley Online Library, ACM Digital Library, Taylor and Francis, and Inderscience are the well-known Digital repositories to find the data sets of publications in the first attempt. Since now and again repository, the search engine highlighted a point of the enclosure to count hunt result meta-information download by a person. The

pursuit string is taken into highlighting and emphasizing the point when we gather all relative meta-information of distributions. Science authorizes data gathering on the first thousand things from its inquiry items. Terribly, in search string which we established in advance; we get more than 10 000 results on this repository.

| Line | Keywords |
|---|---|
| 1 | Static*; dynamic* |
| 2 | Permission; authorization; |
| 3 | Android permission analysis, permission in Android application |
| 4 | Security; Privacy |
| 5 | Android; Mobile; Smartphone*; |

**Table 4**

**Top venue**

We collect all applications in one account to ensure the repository search items are calculated in this paper. We have taken the main 27 venues for the SLR where 19 venues are from the software engineering and computer applications while the other eight venues are from the security and privacy field. Table 5 indicates these venues. The papers are taken from the IEEE Xplore, ACM, and Elsevier. In reality, the venues are not intensive on permission analysis of the Android[85]

| Acronym | Full name | H- index |
|---|---|---|
| Software engineering and computer applications (SE/CA) | | |
| JNCA | "Journal of Network and Computer Applications" | 59 |
| IJCIP | "International Journal of Critical Infrastructure Protection" | 21 |
| JERP | "Journal of Engineering Research and Applications" | 21 |
| JES | "Journal of Environmental Sciences" | 65 |
| IJCIS | "International Journal of Critical Infrastructure Protection" | 21 |
| ADVC | "Advances in Computers" | 56 |
| CEE | "Computers and Electrical Engineering" | 40 |
| FGCS | "Future Generation Computer Systems" | 80 |
| CN | "Computer Networks" | 54 |
| DI | "Digital Investigation" | 37 |
| ICC | "International Conference on Communications" | 7 |
| MobiSys | "International Conference on Mobile systems" | 14 |
| IWCMC | "International Wireless Communications and Mobile Computing Conference" | 7 |
| TWC | "International Conference on Trust and trustworthy computing" | 36 |
| MONET | "Mobile Networks and Applications" | 73 |
| Pervasive Mobcomput | "Pervasive and Mobile Computing" | 46 |
| IWCMC | "International Wireless Communications and Mobile Computing Conference" | 14 |
| GJCST | "Global Journal of Computer Science and Technology: Network" | 26 |
| JCST | "Asian Journal of Computer Science and Technology" | 5 |
| Security and privacy (S&P) | | |

| | | |
|---|---|---|
| COMPSEC | "Computers and Security" | 40 |
| CCS | "Computer and communications security" | 71 |
| CLSR | "Computer Law and Security Review" | 19 |
| ISI | "International Conference on Intelligence and Security Informatics" | 6 |
| NDSS | "Network and Distributed System Security Symposium" | 56 |
| IFS | "Information Forensics and Security" | 77 |
| SSP | "Symposium on Security and Privacy" | 59 |
| CIS | "Computational Intelligence and Security" | 37 |

**Table 5**

## Proposed solution and conclusion

Permission-based analysis and fundamental characteristics of Android malware analysis are presented in this paper. Permission analysis is a progressing approach through which many security issues have been highlighted and rapidly developing application code statically. For this review, 100 research articles are gathered, which are already published in security and privacy journal and conferences for different programming languages with software engineering methodology. The Android permission protocol is having severe security implications because the real-world Android application study confirms the findings but the Android permission protocol has several flaws. The various cases permit the attacker to evade the permission checks entirely. Although, we have covered application permission-based malware a new type of malware family is developing as the android market is growing very fast. A large-scale study on the above topic might give more clarity to the subject. It is also required to study a category that includes users too so that permissions can categorize more precisely

### References

1. Kaur, S., et al., *Review paper on implementing security on Android application.* 2013. **2**(3).
2. Powar, S., B.J.I.J.o.E.R. Meshram, and Applications, *Survey on Android security framework.* 2013. **3**(2): p. 907-911.
3. Felt, A.P., et al. *Android permissions demystified*. in *Proceedings of the 18th ACM conference on Computer and communications security*. 2011.
4. Moore, S.R., et al., *Cybersecurity for android applications: Permissions in android 5 and 6.* 2019. **35**(7): p. 630-640.
5. Garg, S., N.J.C. Baliyan, and E. Engineering, *A novel parallel classifier scheme for vulnerability detection in android.* 2019. **77**: p. 12-26.
6. Mollah, M.B., et al., *Security and privacy challenges in mobile cloud computing: Survey and way ahead.* 2017. **84**: p. 38-54.
7. Pennekamp, J., et al., *A survey on the evolution of privacy enforcement on smartphones and the road ahead.* 2017. **42**: p. 58-76.
8. R, B. *Brandom R*. Available from: https://www.theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote. Accessed August 2019. .
9. Fernando, N., S.W. Loke, and W.J.F.g.c.s. Rahayu, *Mobile cloud computing: A survey.* 2013. **29**(1): p. 84-106.
10. Clarke, R.J.C.L. and S. Review, *The prospects of easier security for small organisations and consumers.* 2015. **31**(4): p. 538-552.
11. Armando, A., A. Merlo, and L.J.I.j.o.c.i.p. Verderame, *Security considerations related to the use of mobile devices in the operation of critical infrastructures.* 2014. **7**(4): p. 247-256.
12. Virvilis, N., et al., *Security Busters: Web browser security vs. rogue sites.* 2015. **52**: p. 90-105.

13. S.Birundha, D.V.V., *Survey on Mobile Malware Detection Techniques in Android Operating System.* International Journal on Applications in Information and Communication Engineering. **2**.
14. Rai, N., and Tripti Arjariya, *A Survey on Detection Techniques of Android Malware.* International Journal of Computer Security and Source Code Analysis (IJCSSCA), 2015.
15. Arshad, S., et al., *Android Malware Detection & Protection: A Survey.* International Journal of Advanced Computer Science and Applications, 2016. **7**.
16. Room, G.N. *2017-02-15-gartner-says-worldwide-sales-of-smartphones-grew-7-percent-in-the-fourth-quarter-of-2016*. 8/23/2020]; Available from: https://www.gartner.com/en/newsroom/press-releases/2017-02-15-gartner-says-worldwide-sales-of-smartphones-grew-7-percent-in-the-fourth-quarter-of-2016.
17. iOS, p.o.n.s.r.A.o. *99.6 percent of new smartphones run Android or iOS - The Verge*. 8/23/2020]; Available from: https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016.
18. Malware, M. *Mobile Malware*. 8/23/2020]; Available from: http://www.webopedia.com/TERM/mobile_malware.html.
19. Report: Top Android Security Problems in 2017" 2017. *Report: Top Android Security Problems in 2017" 2017,* . 8/23/2020]; Available from: https://dzone.com/articles/report-top-android-security-problemsin-2017
20. Zhou, Y. *Malgenome Project" 2011*. 8/23/2020]; Available from: http://malgenomeproject.org.
21. Malik, V., N. Malik, and S.K. Goyal. *Analysis of Android malwares and their detection techniques*. in *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. 2016.
22. 2016, C.A.M. *Current Android Malwares" 2016*. 8/23/2020]; Available from: https://forensics.spreitzenbarth.de/android-malware/.
23. Types, C.M.M. *"Common Mobile Malware Types*. 8/23/2020]; Available from: https://www.veracode.com/blog/2013/10/common-mobile-malwaretypes-cybersecurity-101.
24. 2017", C.A.V.L. *Current Android Viruses List 2017"* Available from: https://drfone.wondershare.com/android-tips/top-android-virueslist.html.
25. Trojan:Android/GinMaster.A. *Trojan:Android/GinMaster.A*. 8/23/2020]; Available from: https://www.fsecure.com/v-descs/trojan_android_ginmaster.shtml.
26. Shrivastava, G., P.J.M.T. Kumar, and Applications, *SensDroid: analysis for malicious activity risk of Android application.* 2019. **78**(24): p. 35713-35731.
27. He, Y., et al., *Dynamic privacy leakage analysis of Android third-party libraries.* 2019. **46**: p. 259-270.
28. Zhang, J., et al., *Dalvik opcode graph based android malware variants detection using global topology features.* 2018. **6**: p. 51964-51974.
29. Faruki, P., et al. *AndroSimilar: robust statistical feature signature for Android malware detection*. in *Proceedings of the 6th International Conference on Security of Information and Networks*. 2013.
30. Omar, M., et al., *Android application security*, in *Applying Methods of Scientific Inquiry Into Intelligence, Security, and Counterterrorism*. 2019, IGI Global. p. 46-67.
31. Sharma, K. and B. Gupta, *Attack in smartphone wi-fi access channel: State of the art, current issues, and challenges*, in *Next-Generation Networks*. 2018, Springer. p. 555-561.
32. Cheng, X., et al., *Exploiting mobile big data: Sources, features, and applications.* 2017. **31**(1): p. 72-79.
33. Batyuk, L., et al. *Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications*. in *2011 6th International Conference on Malicious and Unwanted Software*. 2011. IEEE.
34. Xu, J., et al., *MobSafe: cloud computing based forensic analysis for massive mobile applications using data mining.* 2013. **18**(4): p. 418-427.

35. Xue, Y., et al., *RootAgency: A digital signature-based root privilege management agency for cloud terminal devices.* 2018. **444**: p. 36-50.
36. Sharma, K., B.B.J.I.J.o.E.-S. Gupta, and M. Applications, *Towards privacy risk analysis in Android applications using machine learning approaches.* 2019. **11**(2): p. 1-21.
37. Mustafa, T. and K.J.I.J.o.I.S. Sohr, *Understanding the implemented access control policy of Android system services with slicing and extended static checking.* 2015. **14**(4): p. 347-366.
38. Raveendranath, R., et al. *Android malware attacks and countermeasures: Current and future directions.* in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 2014. IEEE.
39. Wang, W., et al., *Exploring permission-induced risk in android applications for malicious application detection.* 2014. **9**(11): p. 1869-1882.
40. Yang, Z. and M. Yang. *Leakminer: Detect information leakage on android with static taint analysis*. in *2012 Third World Congress on Software Engineering*. 2012. IEEE.
41. Homayoun, S., et al., *BoTShark: A deep learning approach for botnet traffic detection,* in *Cyber Threat Intelligence*. 2018, Springer. p. 137-153.
42. Dietz, M., et al. *Quire: Lightweight provenance for smart phone operating systems*. in *USENIX security symposium*. 2011. San Francisco, CA;.
43. Milosevic, N., et al., *Machine learning aided Android malware classification.* 2017. **61**: p. 266-274.
44. Schlegel, R., et al. *Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones*. in *NDSS*. 2011.
45. Nauman, M., S. Khan, and X. Zhang. *Apex: extending android permission model and enforcement with user-defined runtime constraints*. in *Proceedings of the 5th ACM symposium on information, computer and communications security*. 2010.
46. Zhang, J., C. Tian, and Z. Duan. *FastDroid: efficient taint analysis for Android applications*. in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 2019. IEEE.
47. Bugiel, S., et al. *Poster: the quest for security against privilege escalation attacks on android*. in *Proceedings of the 18th ACM conference on Computer and communications security*. 2011.
48. Bugiel, S., et al., *Xmandroid: A new android evolution to mitigate privilege escalation attacks.* 2011.
49. Hinarejos, M.F., et al., *Risklaine: A probabilistic approach for assessing risk in certificate-based security.* 2018. **13**(8): p. 1975-1988.
50. Fan, M., et al., *Dapasa: detecting android piggybacked apps through sensitive subgraph analysis.* 2017. **12**(8): p. 1772-1785.
51. Schmidt, A.-D., et al. *Static analysis of executables for collaborative malware detection on android*. in *2009 IEEE International Conference on Communications*. 2009. IEEE.
52. Au, K.W.Y., et al. *Pscout: analyzing the android permission specification*. in *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012.
53. Meier, R., *Professional Android 4 application development*. 2012: John Wiley & Sons.
54. Girolami, M., S. Chessa, and A.J.C.N. Caruso, *On service discovery in mobile social networks: Survey and perspectives.* 2015. **88**: p. 51-71.
55. Buchinger, S., et al., *A survey on user studies and technical aspects of mobile multimedia applications.* Entertainment Computing, 2011. **2**(3): p. 175-190.
56. Mislan, R.P., E. Casey, and G.C. Kessler, *The growing need for on-scene triage of mobile devices.* Digital Investigation, 2010. **6**(3-4): p. 112-124.
57. Xue, W.-F., S.W. Homans, and S.E. Radford, *Systematic analysis of nucleation-dependent polymerization reveals new insights into the mechanism of amyloid self-assembly.* 2008. **105**(26): p. 8926-8931.
58. Shrivastava, G. and P. Kumar, *Android application behavioural analysis for data leakage.***n/a**(n/a): p. e12468.

59. Talha, K.A., D.I. Alper, and C. Aydin, *APK Auditor: Permission-based Android malware detection system.* Digital Investigation, 2015. **13**: p. 1-14.
60. Song, J., et al., *An integrated static detection and analysis framework for android.* Pervasive and Mobile Computing, 2016. **32**: p. 15-25.
61. Rashidi, B., C. Fung, and T. Vu, *Android fine-grained permission control system with real-time expert recommendations.* Pervasive and Mobile Computing, 2016. **32**: p. 62-77.
62. Seshagiri, P., A. Vazhayil, and P. Sriram, *AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts.* Procedia Computer Science, 2016. **93**: p. 768-773.
63. Sokolova, K., C. Perez, and M. Lemercier, *Android application classification and anomaly detection with graph-based permission patterns.* Decision Support Systems, 2017. **93**: p. 62-76.
64. Balzarotti, D., et al., *Multi-module vulnerability analysis of web-based applications*, in *Proceedings of the 14th ACM conference on Computer and communications security*. 2007, Association for Computing Machinery: Alexandria, Virginia, USA. p. 25–35.
65. Basin, D., et al., *Automated analysis of security-design models.* 2009. **51**(5): p. 815-831.
66. Stolpe, A., *A theory of permission based on the notion of derogation.* Journal of Applied Logic, 2010. **8**(1): p. 97-113.
67. Jeon, J., et al., *Dr. Android and Mr. Hide: fine-grained permissions in android applications*, in *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. 2012, Association for Computing Machinery: Raleigh, North Carolina, USA. p. 3–14.
68. Zhou, Y. and X. Jiang, *Dissecting Android Malware: Characterization and Evolution*, in *2012 IEEE Symposium on Security and Privacy*. 2012. p. 95-109.
69. Zhang, Y., et al., *Vetting undesirable behaviors in android apps with permission use analysis*, in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*. 2013. p. 611-622.
70. Fang, Z., W. Han, and Y. Li, *Permission based Android security: Issues and countermeasures.* Computers & Security, 2014. **43**: p. 205-218.
71. Li, J., et al., *Significant Permission Identification for Machine-Learning-Based Android Malware Detection.* IEEE Transactions on Industrial Informatics, 2018. **14**(7): p. 3216-3225.
72. Liu, X., et al., *Privacy Risk Analysis and Mitigation of Analytics Libraries in the Android Ecosystem.* IEEE Transactions on Mobile Computing, 2020. **19**(5): p. 1184-1199.
73. Min, L.X. and Q.H. Cao, *Runtime-Based Behavior Dynamic Analysis System for Android Malware Detection.* Advanced Materials Research, 2013. **756-759**: p. 2220-2225.
74. Petsas, T., et al., *Rage against the virtual machine*, in *Proceedings of the Seventh European Workshop on System Security - EuroSec '14*. 2014. p. 1-6.
75. Abah, W.O.V., Abdullahi M.B., Ume U.A., Adewale O.S., Joshua, *Extracting Android Applications Data for Anomaly-based Malware Detection.* 2015.
76. Razak, M.F.A., et al., *The rise of "malware": Bibliometric analysis of malware study.* Journal of Network and Computer Applications, 2016. **75**: p. 58-76.
77. Kumar Thanigaivelan, N., et al., *CoDRA: Context-based dynamically reconfigurable access control system for android.* Journal of Network and Computer Applications, 2018. **101**: p. 1-17.
78. Centonze, P., R.J. Flynn, and M. Pistoia. *Combining Static and Dynamic Analysis for Automatic Identification of Precise Access-Control Policies*. in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. 2007.
79. Moghimi Zand, M. and M.T. Ahmadian, *Application of homotopy analysis method in studying dynamic pull-in instability of microsystems.* Mechanics Research Communications, 2009. **36**(7): p. 851-858.
80. Blaschke, T., *Object based image analysis for remote sensing.* ISPRS Journal of Photogrammetry and Remote Sensing, 2010. **65**(1): p. 2-16.

81. Isohara, T., K. Takemori, and A. Kubota. *Kernel-based Behavior Analysis for Android Malware Detection*. in *2011 Seventh International Conference on Computational Intelligence and Security*. 2011.
82. Amos, B., H. Turner, and J. White. *Applying machine learning classifiers to dynamic Android malware detection at scale*. in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2013.
83. Petsas, T., et al., *Rage against the virtual machine: hindering dynamic analysis of Android malware*, in *Proceedings of the Seventh European Workshop on System Security*. 2014, Association for Computing Machinery: Amsterdam, The Netherlands. p. Article 5.
84. Idrees, F., et al., *PIndroid: A novel Android malware detection system using ensemble learning methods.* Computers & Security, 2017. **68**: p. 36-46.
85. Shrivastava, G., et al., *Privacy issues of android application permissions: A literature review.***n/a**(n/a): p. e3773.