PERMISSIONS RANKING WITH STATISTICAL TECHNIQUES FOR ANDROID MALWARE DETECTION

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF SCIENCE

IN

MATHEMATICS

Submitted by:

Anjali Mishra

(2K20/MSCMAT/05)

Mahima

(2K20/MSCMAT/17)

Under the supervision of

Dr. ANSHUL ARORA

DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

MAY, 2022

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

We, (Anjali Mishra, Mahima), 2K20/MSCMAT/05,2K20/MSCMAT/17, students of MSc(Mathematics), hereby, declare that the project Dissertation titled, " Permissions Ranking with Statistical Techniques for Android Malware Detection" which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Anjali Mishra Mahima

Place: Delhi ANJALI MISHRA AND MAHIMA

Date: 5th May 2022

DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "Permissions Ranking with Statistical Techniques for Android Malware Detection" which is submitted by [Anjali Mishra, Mahima], 2K20/MSCMAT/05,2K20/MSCMAT/17 [Department of Applied Mathematics], Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is a record of the project work carried by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date: 5th May 2022

Dr. ANSHUL ARORA

SUPERVISOR

ASSISTANT PROFESSOR

ABSTRACT

In today's world, smartphones play a vital role in our lives as the control of, everything we do, has been taken over by digital platforms. At present, it is impossible to live without smartphones. There is more than one operating system for smartphones amongst which the Android operating system can be considered one of the most popular ones.

Android is an open-source operating system which means it is available freely. Anyone can develop their android application. Furthermore, being an open-source system cybercriminals also use it to develop their malicious applications. That is why Android has become a key for attackers to invade the privacy of users and sabotage using the same weapon. Further, many malicious applications perform venomous activities to breach the privacy of the users. So, it is mandatory to detect these infringe or malicious applications.

In this paper, we have proposed two statistical-based ranking techniques by taking some references from the *Spearman Ranking test* and *Mann-Whitney U test* to rank the permissions in order to check whether the permission is significant or not. Such a ranking helps us to eliminate irrelevant permissions. Further, we apply machine learning algorithms to the dataset by eliminating the lower-ranked permissions from the dataset. The experimental results demonstrate that we obtain the highest accuracy of 99.4% and 99.25% using both ranking techniques respectively.

ACKNOWLEDGEMENT

The most awaited moment of successful completion of an endeavor is always a result of persons involved implicitly or explicitly in it. The successful completion of the planning and research phases of our project is the result of dedicated efforts put in by many people and this report would be incomplete without giving due credits to them. This acknowledgment is but a token of gratitude in recognition of their help in our endeavor. We sincerely thank our project guide for providing us the solutions that always take us out of all the chaos. It had been an honor and pleasure to work under him. Not just the knowledge but a lot we have learned from them calm and composed attitude for which we will remain indebted to them throughout our life.

We would like to give our sincere thanks to ANSHUL ARORA (Professor) and S.Sivaprasad Kumar (Professor), HOD of the Department of Applied Mathematics, and all the faculties from whom we have learned a lot. We would like to acknowledge our work on “Permissions Ranking with Statistical Techniques for Android Malware Detection” Last but not least, we would like to thank our colleagues, friends, and our parents who were a constant and willing source of encouragement and inspiration for us throughout the project.

Thanking you
Anjali Mishra
Mahima

CONTENTS

LIST OF TABELS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 COMMENCEMENT

In today's world, smartphones have become an irreplaceable part of everyone's lives by replacing personal computers with Internet usage and allowing users to check messages, emails, social media accounts, and access online banking services on such devices.

In the late 1990s, growth in the use of PDAs (personal digital assistants) increased, and it did not take much time to transform them into mobile devices, popularly known as smartphones. The number of smartphone users is increasing over time. In 2022, worldwide 6,648 million people are using smartphones. That's approximately 83.72 percent of the world's population, up from 3.7 billion smartphone users just five years ago in 2016 [1].

In addition, by 2027, the number of smartphone users is predicted to expand at a rate of 4 percent per year, reaching 7.69 billion [2].

Among all existing smartphone operating systems, Android is the most popular operating system.

Android is a modified version of the Linux kernel, developed by Open Handset Alliance and commercially sponsored by Google. Other open-source software is designed primarily for touchscreen mobile devices such as smartphones and tablets. It was introduced in November 2007, with the first commercial Android device, the HTC Dream, being launched in September 2008.

Android has a 70.97 % market share worldwide [3]. The main reason behind this is its wide range of functionalities and its open-source nature. Being an open-source technology, android is not only limited to smartphones but has also extended its reach to TV, car, and automation systems. Android provides SDK to developers to plant their apps and allocate those apps through Android application stores.

Unfortunately, an increasing number of benign applications also raise the number of malware applications. Also, the popularity of Android attracts cyber criminals who design malicious / malware applications.

Malware applications can exploit one's device, steal personal data and compromise security. These malicious applications are plotted to carry out multiple attacks such as spyware, SMS Trojans, mobile banking Trojans, and viruses shown in Fig.1.

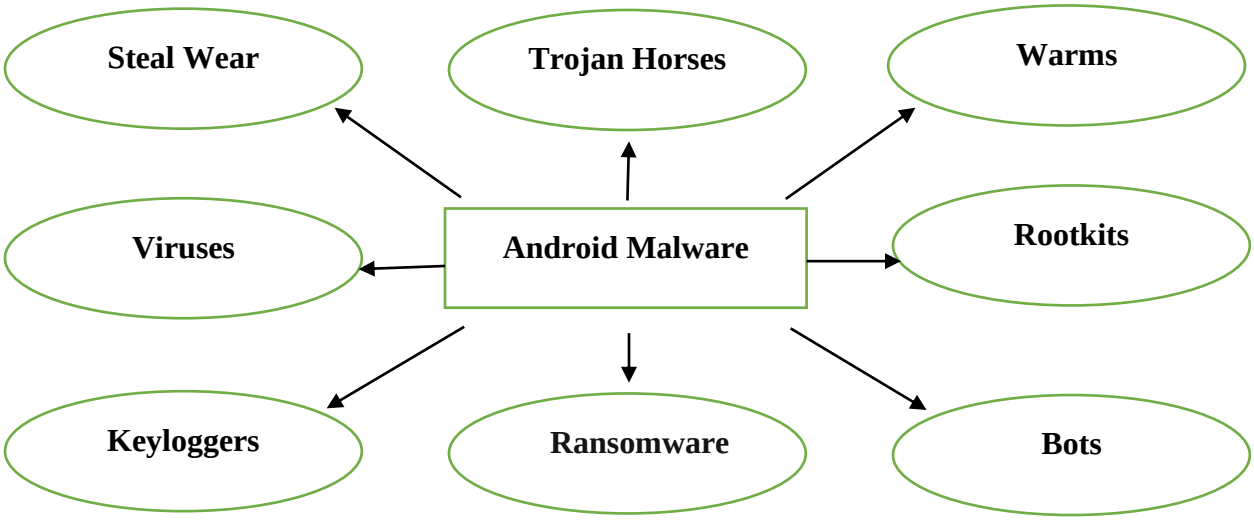

Fig.1.1 Types of Android Malware

Such malicious apps can damage the system, leak sensitive information, send SMS messages in the background without the user's knowledge, etc. So, it is necessarily needed to detect these malicious applications.

In Android malware detection, the study is done in three approaches static, hybrid, and dynamic. In the static approach, malware is pulled apart into a source code and specific features are removed. In the dynamic approach, malware is examined at run-time in the virtual environment. In both cases, the classification models of the dataset have been made by machine learning algorithms. Then the detection of malicious apps is done by using these classification models.

Android app stores organize 32 app categories in the google play store such as Social media apps, Educational apps, Entertainment apps, music apps, sports apps, etc. Also, these categories have their features. Also, the applications which have similar categories share similar functionalities. One of the most important features is permission.

Permissions are a type of data that is asked by the application from the user so that the application can run smoothly.

Normal Permissions: Permissions that pose a low risk to the user, system, or device and are granted by default during the installation process.

Dangerous Permissions: Permissions that are dangerous and pose a significant risk due to their ability to access private information and the device's most critical sensors.

On the other hand, malicious apps request anomalous features to do their malicious activities.

In this regards, this paper describes the detection of malware applications by the ranking of the permissions of the applications using statistical techniques. We apply statistical tests to rank the permissions so that irrelevant permissions can be deleted from the detection method. Further, on the ranked permissions list, we delete the lower-ranked permissions and apply machine learning algorithms for malware detection.

1.2 MOTIVATION

As the growth of android benign apps is increasing day by day the growth of Android malware apps is also increasing day by day. According to the researchers, the malicious attacks on smartphone users exhibited “exceptional” peaks during the start and the end of February 2022. Now, hackers are shifting their focus toward discrete infections through email and IoT. Their focus is on enterprise businesses and governments versus average web users.

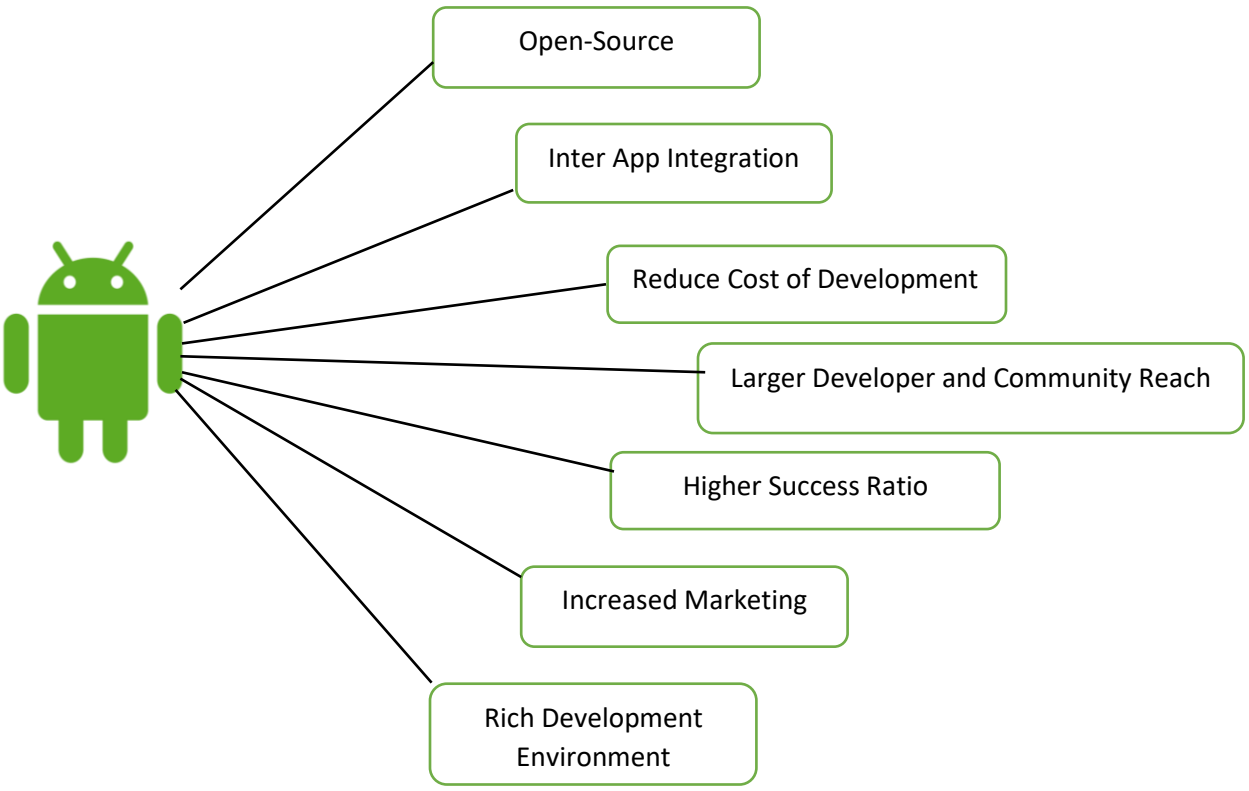

Fig. 1.2 About Android

Being an open-source system, Publishing of Android applications is increasing day by day. On the report of statistics (Statista), the number of applications accessible on Google play store is around 3,298,329 in 2022 [45].

Currently, Machine learning algorithms are used for detecting malware on Android. Mostly, the researchers have their focus on machine learning classifiers to classify the malware to a known malware family. Also, determine a new one by using semi-supervised learning. Machine learning algorithms predict extraordinary accuracy rates at perceiving malicious apps based upon the superiority of the features.

1.3 THESIS STRUCTURE

The structure of this thesis report is systemized into 6 chapters.

Chapter 2 presents a brief introduction to Android OS such as background, architecture, android security features, and Android malware.

Chapter 3 presents related work that has been done for malware perceived in Android OS.

Chapter 4 presents the proposed ranking methods of this research which include structure, extracting permissions, selecting permissions using ranking techniques, and detection process.

Chapter 5 shows the outcomes, the significant findings, and the performance of the classifiers, conclusion, and future work.

References.

CHAPTER 2 ANDROID

2.1 BACKGROUND

Android was constructed based on the Linux kernel and developed by Google. It is released on September 23, 2008. It is an open-source operating system for smartphones, tablets, TVs, cars, and wearable devices. Being an open environment of Android, many companies and manufacturers use it for their products as a platform. Moreover, it allows companies to establish it to meet their needs.

Android provides a warm development environment through numerous kits: Android NDK, Android SDK, Android Developer Tools (Eclipse), and Android Debug Bridge (ADB). As a new version of Android is released Android Software Development Kit (SDK) goes updated. it provides inclusive packages of Java framework classes, libraries, and debuggers for programmers. Android Native Development Kit (NDK) is a set of libraries written in programming languages such as C, C++, and more, that may be filled into Java code by System.loadLibrary call.

Android Debug Bridge (ADB) consists of three components: client, server, and daemon. It is a command-line tool in a client-server form. ADB authorizes an organizer to check their apps for bugs using terminal commands by linking the software running device to a PC.

Android Developer Tools (Eclipse) is an Integrated Development Environment (IDE) used to develop Android applications and provides many functionalities through command lines/ GUI. Also, the Google Play Store is the official distribution center for Android Apps. It provides app installation and updates.

Unfortunately, an open-source environment of Android attracts developers and cybercriminals to establish their apps. Android markets use a variety of procedures to catch and extract malicious applications. Also, Google Play Store uses “Bouncer” to scan the uploaded applications and apply security evaluation before letting the application be published.

2.2 ANDROID ARCHITECTURE

Android stack consists of 6 layers shown in Fig. 2.1.

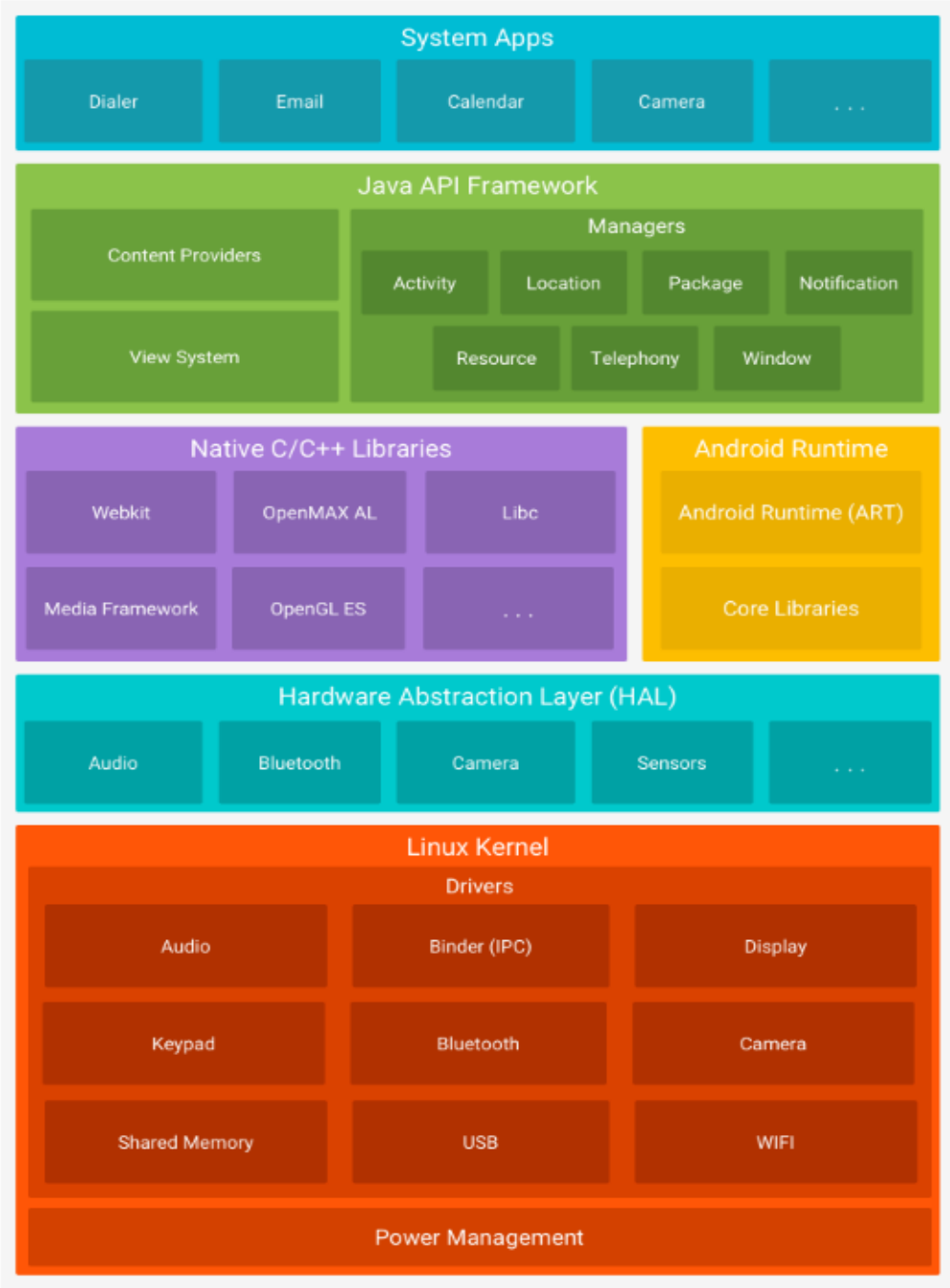

Fig.2.1 Architecture of Android

These layers control the entire system starting from power management to the system application. Each layer provides particular services.

The first layer, The Linux kernel; The foundation of Android OS is based on the Linux kernel. It manages the hardware's functionalities such as power, drives, memory, security settings, hardware abstraction, and shared libraries. The second layer, Hardware Abstraction Layer (HAL); It contains a verity of libraries, each of which is responsible for a particular type of hardware component. It provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The third layer, Native libraries; it is a set of guidelines that are responsible for the data process. It gives open-source libraries like Webkit, Media framework, OpenMAX AL, Libc, and OpenGL-ES. It also provides the Android runtime libraries which include the Dalvik VM and the core libraries. The fourth layer is, Java API Framework; APIs are interfaces for the development of Android applications. This layer organizes the basic functionalities on the system and links with the running applications. It consists of programs content providers, view systems, Activity managers, Location managers, Package managers, Notification managers, Resources managers, Telephony managers, and Window managers. The last layer is System apps; Finally, it is the step where the phone's functionalities are provided to the user. It consists of a set of core apps such as SMS messaging, Emails, Calendars, Contacts, and so on [46].

2.3 ANDROID SECURITY FEATURES

Android security focuses on the smartphone's hardware, user data, and system. Android's security features are as follows: App sandbox, App signing, Authentication, Biometrics, Encryption, Keystore, Security-Enhanced Linux, Trusty Trusted Execution Environment (TEE), and Verified Boot.

App sandbox identifies and isolates app resources by using Linux user-based protection. It allocates a unique user ID (UID) to each app. This UID is used to set up a kernel-level app sandbox. App signing permits a user to recognize the app's author and to update their application without any kind of complications. The user-authentication-gated cryptographic key which requires a

cryptographic key, service provider and user authentication is used by the Android for authentication purposes. Users can sign in to the derives by their fingerprints. A device can have more than one fingerprint. For the transection purpose, high android versions use a BiometricPrompt API. With this, application developers may affiliate biometric authentication into their applications. Encryption makes the data unreadable that is if the data is accessed by an unauthorized user still, he/she will not be able to read that data. Android provides a hardware-backed Keystore. It provides a key generation. Security-Enhanced Linux (SELinux) is used by Android to impose mandatory access control (MAC) over the process. Trusty is an OS and enables a Trusted Execution Environment. Trusty is isolated and works as same as the android. Verified Boot ensures that all the executed codes come from a trusted source but not from any attacker or hacker [47].

2.4 ANDROID MALWARE

The below table represents android malware and its work.

Table 2.1 Android Malware

Type	**Definition**
Worm	It launches by itself and makes copies and spreads over a network's node.
Trojan	It shows like a benign app and offers the user to use its useful functionalities but performs its malicious activities in the background by hiding from the user.
Spyware	It collects the user's information and sends them to the linked remote device. Also, it sends users' data like messages, location, photos, and more to a remote server.

Backdoor	It accesses high-level user access by exploiting the system's authentication mechanism. It is undetected and enables remote access.
Bot	It enables remote control over the device from a server called Bot-master. It is used to launch an attack called DDoS.
Ransomware	It makes the system inaccessible and encrypts the data.
Adware	It sends customized advertisements found on a user's collected data such as location.

CHAPTER 3 LITERATURE REVIEW

3.1 RELATED WORK

Earlier research, such as [5], and [6], looked at permissions to detect harmful behavior within regular apps. By evaluating permissions and API requests, Grace et al. [5] assessed potential dangers connected with in-app advertisement libraries. Based on a combination of permissions, the Kirin model [6] has established security criteria to identify high-risk apps. The authors of these investigations looked into normal in-app permissions for evidence of nefarious conduct. The analysis did not include malware samples

Sanz et al. [7] used machine learning classifiers to extract top permissions in malicious and benign apps. The permissions and other properties of the manifest file were retrieved by the MAMA model [8]. The authors of each of these papers concentrated on obtaining permissions that are commonly utilized by malware and normal applications.

Permissions were retrieved from the apps by Talha et al. [9]. The score for each permission was calculated by comparing the amount of malware that contained the permission to the total number of malwares. Tao et al. [10] used permissions, APIs, and the links between them to detect malware. For detecting malicious samples, Cen et al. [11] used a probabilistic discriminative model on decompiled source code and permissions. Peng et al. [12] used probabilistic generative models such as Naive Bayes to assess the dangers of Android apps based on the permissions requested.

To discover problematic permission patterns, the permission patterns mining method is used in [13]. The model, on the other hand, did not take into account common apps and did not present any detection models. Similarly, the dangerous permissions were ranked using feature ranking methods in the model provided in [14]. When using a set of 40 permissions, SVM produces good accuracy, but Random Forest produces superior results with ten permissions.

The DroidMat [15] model used K-means clustering to detect permissions, intents, components, and API requests. To detect malware, Arp et al. [4] looked at permissions, hardware components, API calls, and network addresses. Permissions and intentions were utilized to detect malware in

the study proposed in [16]. Kim et al. [17] built an app vector out of static features like manifest file components, strings, and API calls.

To depict the association between the permissions of regular apps, Solokova et al. [18] employed graphs. They categorized apps in the same category and calculated metrics for each graph, such as node degree, weighted degree, and page rank score. However, they only looked at the apps that were classified as normal and not the malicious samples. To generate different permission pair graphs, the suggested method in this study considered both malware and normal apps.

Zhu et al. [19] developed a mechanism for evaluating an app's hazards based on its permissions. They built a bipartite network, with the first set of vertices representing apps and the other set of vertices representing permissions. Similarly, a new Android malware detection method based on static analysis and graph neural networks was proposed in [43] . The source code of Android applications was analyzed to extract high-level semantic information, which raised the detection barrier. To represent an Android program, an estimated call graph from function invocation links was generated, and then intra-function data was extracted, such as necessary permission, security level, and statistical instructions information, to form node attributes within graph structures. After that, a graph neural network (GNN) was employed to produce a vector representation of the application, which was then used to classify malware

Lu et al. [20] presented a two-layered malware detection methodology based on permission. The analysis for the first layer detection used an upgraded random forest technique. The fuzzy sets created by the first layer detection were analyzed using sensitive permission rules matching in the second layer detection. The findings demonstrated that using sensitive permission rules improved the detection accuracy of Android malware.

The authors in [21] proposed a permission weight approach using which each permission was assigned a distinct score. The algorithms K-nearest Neighbor (KNN) and Naive Bayes (NB) were then used.

The proposed system in [22] used a reduction of features to find the most influential permissions. Random Committee, Sequential Minimal Optimization (SMO), Multilayer Perceptron, J48, and Randomizable filtered classifiers were used for feature reduction, and

Random Committee, Sequential Minimal Optimization (SMO), Multilayer Perceptron, J48, and Randomizable filtered classifiers were used for feature evaluation. The trials demonstrated that five permissions can reach near-complete feature accuracy, allowing the malware detection system to be improved

The performance of some machine learning methods, such as naive Bayes, J48, Random Forest, Multi-class classifier, and Multi-layer perceptron, was investigated in [23]. For normal apps, Google Play store data from 2015 and 2016 was used, and typical malware data sets were used in the evaluation. In terms of classification accuracy, the multi-class classifier outperformed the other algorithms. In terms of model development time, the Naive Bayes classifier had outperformed.

The authors in [24] proposed a hybrid technique based on network traffic and permission bit-vectors to detect malware. A decision tree classifier was built to detect the android malware. The results showed that combining network traffic analysis and permission bit-vector was highly efficient, with a detection accuracy of 95.56 percent. Kuo et al. [44] proposed a malware detection system that used a hybrid analysis model and machine learning methods (SVM or Random Forest). The combination of the Permissions characteristic from the static analysis approach and API from the dynamic analysis method is the main feature of the hybrid analysis model. The accuracy rate and TP (true positive) rate employing the proposed technique were 88 percent and 89 percent, respectively, according to the experimental results.

A machine learning-based malware detection method was suggested in [25] to differentiate Android malware from benign apps. Using a linear regression-based feature selection approach, the proposed malware detection system's feature selection stage aimed to reduce unneeded features. When the study's findings were examined, the highest score of 0.961 was attained utilizing the F-measure metric and at least 27 features.

For Android malware detection, Firdaus et al. [26] used static analysis and a genetic search-based strategy to find the best features. Five machine learning classifiers were employed to evaluate the best features determined by GS: J48, Naive Bayes (NB) random forest (RF), multilayer perceptron (MLP), and functional trees (FT). With only six features, FT provided the highest accuracy (95%) and true positive rate (TPR) (96.7%) among these classifiers.

Wang et al.[27] proposed a novel method for extracting different permission patterns, which were used to evaluate the variations in permissions between Android benign apps and malware, and to identify Android malware using these differences. Then, using the support-based permission candidate method, unique required or utilized permission patterns were mined for the detection of Android malware. This method used permission patterns from Android malware to detect a mixed Android app dataset in an experiment. The results revealed that the proposed method has a 94 percent accuracy rate, a 5% false-positive rate, and a 1% false-negative rate.

The Android-based malware detection and classification framework were built using eXtreme Gradient Boosting (XGBoost) in [28]. APK permission categories were retrieved from Android apps and used in the framework. The comparison of modeling results revealed that the XGBoost is particularly well-suited to Android malware classification, achieving a F1 score of 74.40 percent with real-world Android application sets.

The authors in [29] presented a two-phase static Android malware analysis approach based on bloom filters. Phase I involved two filters for a bloom that classified a sample as malware or benign only based on the features allowed. Phase II used a Naive Bayes Classifier with permission and a code-based mixed feature set to assess the escaped harmful samples from Phase I. The addition of Phase I classification reduced the computing complexity of the technique, while Phase II classification improved the overall accuracy of the suggested model.

Congyi et al.[30] proposed a method for detecting and distinguishing Android malware. First, a static analysis of the AndroidManifest file in APK was conducted to extract system characteristics such as component calls, permission calls, and intents. Then, to detect malware programs, an ensemble learning implementation XGBoost approach was employed.

The authors in [31] proposed a novel static approach to detect fraudulent Android programs by offering a set of Android program properties, including sensitive API calls, sensitive permissions, and utilizing Extreme Learning Machine. Tiwari et al. [32] used permissions and APIs for detecting Android malware. Common and combined feature vectors were created. Using logistic regression,97.25 percent accuracy was achieved for common characteristics and 96.56 percent accuracy was achieved for composite features. Furthermore, to reduce categorization training and

testing time, a reduction of features to 131 by deleting low variance features was done, with which 95.87 percent accuracy was achieved.

In[33] Latex semantic Indexing was applied to identify malware applications.

The authors in [34] presented the EDMDroid malware detection system, which is based on frequency of Android permission. The goal of this framework was to increase detection rates by ensuring that classifiers of ensemble base are diverse. To achieve variety, random feature subspace and bootstrapping technology were utilized to build data subsets, then non-negative matrix factorization (NMF) technology was applied to produce new data sets based on subsets of the data. The integrated predicted outcomes of the model trained by the decision tree algorithm were used to generate the final prediction results using the integrated strategy voting technique. The results of the tests showed that EDMDroid is a good tool for detecting Android malware.

The authors in [35] used permission-based techniques to identify applications as malware and classify them as benign using filtering function selection algorithms and machine learning algorithms such as Random Forest, SVM, and J48.

In [36] , an ensemble model that distinguished between dangerous and benign apps based on permission combinations was created. Combining classifiers into an ensemble model improved accuracy over using a single classifier. The malware detection rate came out to be 99.3 percent.

For malware comprehension, detection, and classification, a precise semantic model of Android malware based on Deterministic Symbolic Automaton (DSA) was proposed in [37]. It demonstrated that DSA could capture both the malware variations and the malware family's shared harmful behaviors. In[37], researchers developed the SMART automatic analysis framework, which learned DSA by recognizing and summarising semantic clones from malware families and then extracted semantic properties from the learned DSA to classify malware based on attack patterns.

NTPDroid, a hybrid detection approach that extracted Network Traffic features and Permissions from applications was proposed in [38]. To build frequent patterns consisting of traffic features and permissions, the FP-Growth algorithm was used to train and test the proposed model. The detection accuracy of the experimental results achieved was 94.25 percent, which was higher than the detection accuracy of frequent patterns acquired separately.

Using a deep neural network model, the study in [39] focused on detecting malware that can infiltrate through permissions on Android. The suggested method detected permission-based malware in real-time Android apk files with an accuracy of above 85%.

The authors in [40] developed an Android malware dataset in which the second half of the dataset comprised API calls as dynamic features and permissions and intents as static features. The malware scanner for Android was two-layered. According to our findings, at the first layer, 95.3 percent precision in Static-Based Malware Binary Classification was achieved, 83.3 percent precision in Dynamic-Based Malware Category Classification, and 59.7% precision in Dynamic-Based Malware Family Classification at the second layer was achieved.

Information Gain was used by the authors in [41] to rank permissions and intents in order to find the optimum set of intents and permissions for more accurately identifying Android malware. A combination of machine learning algorithms such as Random Forest, SVM, and Naive Bayes was used to discover the optimal set. The best set consisted of 37 features, including 20 intents and 17 permissions, according to the trial data, and the Random Forest classifier had the best accuracy of 94.73 percent.

Shyong et al. [42] proposed a method to improve Android Malware Detection by Using Static Permissions and Dynamic Packet Analysis. The malware's network traffic was used to extract numerous sorts of features in the dynamic analysis stage, and machine learning was employed to achieve malware family categorization.

CHAPTER 4 PROPOSED METHODOLOGY

4.1 STRUCTURE

In this proposed methodology, firstly, the APK files of benign and malware applications are extracted. `The flowchart of the proposed methodology is shown below in Figure 4.1.

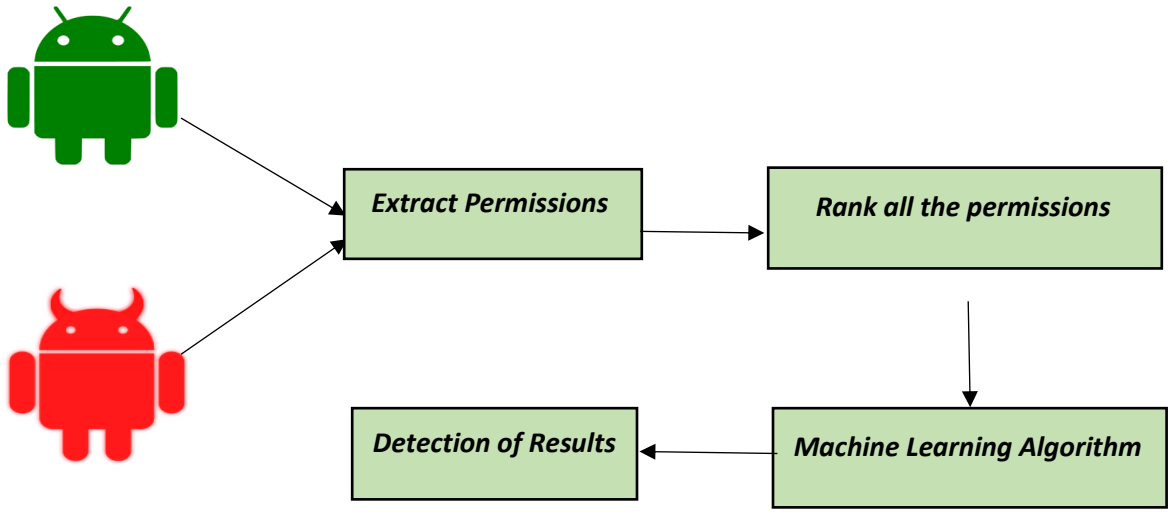

Fig. 4.1 Methodology Process

After, the extraction of the APK file there are enormous folders and files present in the application's folder. There is a file namely manifest.xml which contains a list of all the permissions needed for the application. In this work, the dataset consists of 12,000 benign applications and 5,000 malware applications from Genome [20], and Drebin [21]. By, the extraction of this data, all the permissions were collected. Then, the occurrence of these permissions was counted using the Python and a sheet of these permissions with a label was made. Permissions that are present in benign applications are labeled by 1 and permissions that are present in malware applications are labeled by 0.

To identify distinguishing/significant permissions, we ranked the permissions based on statistical techniques of the ***Spearman ranking test*** and the ***Mann Whitney U test.***

The Spearman Ranking test is a nonparametric measure of rank correlation. It is a version of the Pearson correlation. To apply the spearman ranking test we need two variables or two datasets. This method is appropriate for both discrete and continuous variables.

The Mann-Whitney U test is a nonparametric test for randomly selected values of variables. It is also called Mann-Whitney Wilcoxon. Mainly this method is used for continuous variables.

Permissions present only in malicious applications are permissions with high risk. On the other hand, permissions present only in benign applications are permissions with low risk. Some permissions are common to both malware and benign application and hence those permissions have to be ignored, since they may confuse the malware detection process. Therefore, we aim to rank the permissions to find the significant permissions. This paper discusses two methods to rank the dataset of permissions. Here, our database consists of the following columns:

1. Permissions' Name
2. Number of times the permission has occurred in benign applications
3. Number of times the permission has occurred in malware applications

4.2 FIRST RANKING METHOD

This ranking is inspired by Spearman Ranking Test. In the first method, the rank of the permissions of both benign and malware applications is done separately.

Rank R_1 (Rank of the permissions of benign applications) is found as follows:

1. The permission which is present in the least number of benign applications is assigned as rank 1 and the permission which is present in the second-least number of benign applications is assigned as rank 2 and the process continues.

2. The permissions that are present the same number of times in benign applications are ranked differently.
   The ranking of such permission is given by the formula below:

$$\frac{Sum\ of\ the\ values\ of\ ranks\ for\ all\ the\ duplicate\ values\ in\ benign\ application\ column}{Number\ of\ times\ the\ duplicates\ are\ present\ \ in\ benign\ applications}$$

3. Similarly, Rank R_2 (Rank of permissions in malware applications) is found.
4. The final rank of the permissions in our dataset is then found by R_1-R_2

When the process is done the range of almost all common permissions will come in some interval and the range of almost all distinct benign permissions will become positive outside of this interval and the range of almost all distinct malicious permissions will become negative outside of this interval. Then $|R_1 - R_2|$ is found.

Our database is then sorted in descending order by the column namely $|R_1\text{-}R_2|$ so that almost all the distinct permissions come at the top of the sheet and common permissions between benign and malware applications come at the bottom. Removal of lower-ranked permissions one by one from the labeled database is done until the set of best permissions is left. In addition, the accuracy of this dataset is done through the machine learning algorithms: Decision Tree, SVC, and Random Forest.

4.3 SECOND-RANKING METHOD

In this method, inspired by Mann Whitney U Test, the rank of permissions in both benign and malware applications is calculated simultaneously.

1. While ranking, both the columns of benign and malware applications are considered together. The permission which is least present in either benign or malware applications is assigned as rank 1 under the column of R1 or R2 where R1 is the column for the rank of benign applications and R2 is the column for the rank of malware applications and the

permission which is second least present in either benign or malware applications is assigned as rank 2 and the process continues.

2. The permissions that are present the same number of times in both benign and malware applications are ranked differently:
   The ranking of such permission is given by the formula below:

$$\frac{Sum\ of\ ranks\ for\ all\ \ duplicate\ values\ present\ \ in\ both\ \ benign\ and\ malware\ application\ column}{Number\ of\ times\ the\ duplicates\ are\ present\ \ in\ both\ benign\ and\ malware\ applications}$$

Note that, if the values in both the columns namely benign and malware contain duplicates, then the average rank is assigned to each set of duplicates simultaneously.

3. The ranking of the dataset is done by finding $\frac{R_1-R_2}{N}$. where N represents the total number of permissions.

When the process is done the range of almost all common permissions will come in some interval and the range of almost all distinct benign permissions will become positive outside of this interval and the range of almost all distinct malicious permissions will become negative outside of this interval. Then, $\left|\frac{R_1-R_2}{N}\right|$ is found.

The database is then sorted in descending order by the column namely $|\frac{R_1-R_2}{N}|$ so that almost all the distinct permissions come at the top of the database and common permissions between benign and malware applications come at the bottom.

4.4 DETECTION PROCEDURE

Once we get the ranked permissions, we aim to remove irrelevant permissions which may decrease the detection accuracy. Hence, we aim to remove the lower ranked permissions from both

the rankings. Removal of lower-ranked permissions one by one from the labeled database is done until the set of best permissions is left. In addition, the accuracy of this dataset is done through the machine learning algorithms of Decision Tree, SVC, and Random Forest Tree. In the next section, we discuss the results obtained from the proposed approach.

CHAPTER 5 RESULTS AND CONCLUSION

5.1 MACHINE LEARNING ALGORITHMS

A **machine learning algorithm** is a type of computer algorithm. These algorithms build a part of the dataset (generally 75% of the dataset and called the training dataset) to train the machine, to make predictions/ decisions without being explicitly programmed to do so. Some algorithms are Decision Tree, SVC, Random Forest, etc.

The **Decision Tree algorithm** is used for classification and regression problems. But preferably it is used for classification. It contains two nodes one is Decision Node and the second is Leaf Node. The decision node has multiple nodes and makes the prediction whereas Leaf nodes are the output of those predictions.

The **Support Vector Classifier** (SVC) is put a decision boundary to categorize the data. The best decision boundary is known as a hyperplane. It uses an extreme vector to create the hyperplane. These vectors are called support vectors.

The **Random Forest** is proposed from ensemble learning, which is used to solve a complex classifier by the combination of multiple classifiers. It consists of numerous decision trees on various subsets of the dataset and predicts the accuracy of the dataset.

5.2 CLASSIFICATION AND ACCURACY

In order to check the accuracy of these methods, machine learning algorithms have been used.

Here, the dataset of 12k benign applications and 5k malware applications is provided. The dataset of benign samples is downloaded from the google play store. A total of 293 permissions have been taken in the experiment. Table 5.1 summarizes the top 10 permissions obtained from both the ranking techniques.

Table 5.1: Top 10 Permissions from both the ranking techniques

Rank	**Top 10 Permissions from the First Ranking Method**	**Top 10 Permissions from Second Ranking Method**
1	BIND_DEVICE_ADMIN	BIND_GET_INSTALL_REFERRER_SERVICE
2	RECEIVE_WAP_PUSH	RECEIVE
3	ACCESS_MTK_MMHW	FOREGROUND_SERVICE
4	BROADCAST_SMS	C2D_MESSAGE
5	RECEIVE_MMS	READ
6	BIND_GET_INSTALL_REFERRER_SERVICE	WRITE
7	RECEIVE	BROADCAST_BADGE
8	GLOBAL_SEARCH_CONTROL	CHANGE_BADGE
9	BROADCAST_WAP_PUSH	UPDATE_SHORTCUT
10	FOREGROUND_SERVICE	UPDATE_COUNT

These permissions are ranked by the first-ranking method. After ranking, the lowest-ranked permission is removed from the labeled excel sheet. After deleting lower-ranked 221 permissions one by one, the highest accuracy of 99.45% is achieved. Also, if more permissions are removed, the accuracy starts reducing so the process is not continued further.

The following results are achieved with the first ranking method, as summarized in Table 5.2.

Table 5.2: Detection Results when lower-ranked permissions

are deleted from the first ranking technique

Machine Learning Algorithms	**Detection Accuracy (in %)**		
	Before Ranking (no permissions deleted)	**After Ranking (lower ranked permissions deleted)**	**If more permissions are deleted**
Decision Trees	80.75	99.40	99.2
SVC	73.04	94.54	94.1
Random Forest	80.75	99.22	99.2

The above table summarizes that the dataset had the highest accuracy of 80.7% before the ranking technique was applied. When the ranking technique was applied, the highest accuracy reached 99.4%. In this process, we removed 221 lower-ranked permissions from the dataset. Also, the last column tells that if we remove more permissions then the accuracy goes down.

Also, these permissions are ranked by the second-ranking method. After ranking, the lowest-ranked permission is removed from the labeled excel sheet. After deleting 119 permissions one by one, the highest accuracy of 99.25% is achieved. Also, if more permissions are removed, the accuracy starts reducing so the process is not continued further. The following results are achieved with the second-ranking method, as summarized in Table 5.3:

Table 5.3: Detection Results when lower-ranked permissions

are deleted from second-ranking technique

Machine Learning Algorithms	**Detection Accuracy (in %)**		
	Before Ranking (no permissions deleted)	**After Ranking (lower ranked permissions deleted)**	**If more permissions are deleted**
Decision Trees	80.75	99.25	99.2

SVC	73.04	91.02	90.8
Random Forest	80.75	99.25	99.2

The above table summarizes that the dataset had the highest accuracy of 80.7% before the ranking technique was applied. When the ranking technique was applied, the highest accuracy reached 99.25%. In this process, we removed about 119 permissions from the dataset. Also, the last column tells that if we remove more permissions then the accuracy goes down.

Hence, from the results, we can conclude that permissions ranking is quite useful in improving the detection accuracy of Android malware detection. We get the highest accuracy of 99.40% when we apply the Decision Trees classifier to the ranked permissions obtained from the first ranking technique.

5.3 CONCLUSION AND FUTURE WORK

Since smartphones are vital to our lives it is very important to protect our privacy when we use them. In this paper, statistical-based ranking algorithms were proposed to rank the permissions that are requested by the application when the application is downloaded. Such a ranking can help in eliminating irrelevant permissions that can further improve detection accuracy. The best permissions set was constructed using two proposed ranking algorithms. In addition, machine learning algorithms were used to detect malicious Android apps. The highest accuracy achieved was 99.4% using the first ranking method and 99.2% using the second-ranking method.

In the future, we will incorporate other static techniques and may include intents and hardware components with permissions.

REFERENCES

[1]. https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[2]. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[3]. https://gs.statcounter.com/os-market-share/mobile/worldwide

[4]. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware inYour Pocket", NDSS, 2014.

[5]. M. C. Grace, W. Zhou, X. Jiang, and A.'R. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements", 5th ACM WiSec, 2012.

[6]. W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification", 16th ACM CCS, 2009.

[7]. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. A´ lvarez, "Puma: Permission usage to detect malware in android, International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions, Springer Berlin Heidelberg, 2013.

[8]. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. A´ lvarez Maran, "MAMA: manifest analysis for malware detection in android", Cybernetics and Systems, 44(6-7), 469-488, 2013.

[9]. K. A. Talha,D. I. Alper, and C. Aydin, APK Auditor: "Permission-based Android malware detection system", Digital Investigation, 13, 1-14, 2015.

[10]. G. Tao, Z. Zheng, Z. Guo, and M. Lyu, "MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs", IEEE TRANSACTIONS ON RELIABILITY, 67(1), 355-369, 2018.

[11]. L. Cen, C. Gates, L. Si, and N. Li, "A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code",
IEEE Transactions On Dependable And Secure Computing, 12(4), 400-412, 2015.

[12]. Peng et al. "Using Probabilistic Generative Models for Ranking Risks of Android Apps", ACM CCS 2012.

[13]. V. Moonsamy, J. Rong, and S. Liu, "Mining permission patterns for contrasting clean and malicious android applications", Future Generation Computer Systems, 36, 122-132, 2014

[14]. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zang, “Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection”, IEEE Transactions on Information Forensics and Security, 9 , 1869-1882, 2014.

[15]. D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, “Droidmat: Android malware detection through manifest and api calls tracing”,
Seventh Asia Joint Conference on Information Security (Asia JCIS), 2012.

[16]. F. Idrees, M. Rajarajan, M. Conti, T. Chen, and Y. Rahulamathavan, “PIndroid: A novel Android malware detection system using ensemble learning methods”, Computers & Security, 68, 36-46, 2017.

[17]. T. Kim, B. Kang, M. Rho, S. Sezer, and E. Im, “A Multimodal Deep Learning Method for Android Malware Detection Using Various
Features”, IEEE Transactions on Information Forensics and Security, 14 (3), 2019.

[18]. K. Sokolova, C. Perez, and M. Lemercier, “Android application classification and anomaly detection with graph-based permission patterns”, Decision Support Systems, 93, 62-76, 2017.

[19]. H. Zhu, H. Xiong, Y. Ge, and E. Chen, “Mobile App Recommendations with Security and Privacy Awareness”, ACM KDD 2014.

[20]. T. Lu ,S. Hou, “ A Two-Layered Malware Detection Model Based on Permission for Android”, IEEE International Conference on Computer and Communication Engineering Technology (CCET), 2018

[21]. D.O.Sahin,O.E.Kural,S. Akleylek, E.Kilic, “New results on permission based static analysis for Android malware”, IEEE 6th International Symposium on Digital Forensic and Security (ISDFS) , 2018

[22]. S.J.K, S. Chakravarty, R.K.V.P , “Feature Selection and Evaluation of Permission-based Android Malware Detection.”,4th International Conference on Trends in Electronics and Informatics (ICOEI) ,2020

[23]. R.K.V.P, K. P.Raj, K.V.S.Raju, “Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms”, IEEE International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud,2017

[24]. S.Kandukuru, R. M. Sharma, “Android malicious application detection using permission vector and network traffic analysis.” IEEE 2nd International Conference for Convergence in Technology (I2CT) ,2017

[25]. D.O.Sahin,O.E.Kural,S. Akleylek, E.Kilic , “A novel permission-based Android malware detection system using feature selection based on linear regression”, Neural Computing and Applications ,2021

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

[27]. C .Wang, Q.Xu , X.Lin, S.Liu, “Research on data mining of permissions mode for Android malware detection” Cluster Computing, 2018

[28]. T. N. Turnip, A.Situmorang,A.Lumbantobing ,J.Marpaung ,S.I. G. Situmeang, “Android malware classification based on permission categories using extreme gradient boosting”, International Conference on Sustainable Information Engineering and Technology,190-194,2020

[29]. P.M.Kate,S.V. Dhavale, “Two Phase Static Analysis Technique for Android Malware Detection” (2015). ACM International Symposium on Women in Computing and Informatics, 650–655, 2015

[30]. D. Congyi ,S Guangshun, “Method for Detecting Android Malware Based on Ensemble Learning”,ACM International Conference on Machine Learning Technologies,28-31 ,2020

[31]. Y.Sun,Y.Xie,Z. Qiu,Y. Pan,J. Weng,S. Guo,“ Detecting Android Malware Based on Extreme Learning Machine”,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, 47–53,2017

[32]. S. R. Tiwari , R. U. Shukla, “An Android Malware Detection Technique based on Optimized Permissions and API” ,International Conference on Inventive Research in Computing Applications (ICIRCA), 2018

[33]. H.Shahriar,M. Islam,V.Clincy, “Android malware detection using permission analysis” IEEE SoutheastCon ,1–6, 2017

[34]. H Fang, H.J. Zhu,“,EDMDroid:Ensuring Diversity to improve Android malware detection based on permissions .” International Conference on Internet of Things and Intelligent Applications (ITIA), 2020

[35]. S.Ilham ,G. Abderrahim ,B.A.Abdelhakim , “ Permission based malware detection in android devices”, ACM International Conference on Smart City Applications,1–6,2018

[36]. E. Amer, “Permission-Based Approach for Android Malware Analysis Through Ensemble-Based Voting Model ”, International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC),2021

[37]. G. Meng ,Y. Xue, Z. Xu, Y.Liu ,J.Zhang ,A. Narayanan , “ Semantic modeling of Android malware for effective malware comprehension, detection, and classification”, International Symposium on Software Testing and Analysis, 306–317,2016

[38]. A.Arora ,S.K.Peddoju, “NTPDroid: A Hybrid Android Malware Detector Using Network Traffic and System Permissions”, IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) , 808–813 ,2018

[39]. P.Sirisha, K., P. B, A. K.K, A. T, “Detection of Permission Driven Malware in Android Using Deep Learning Techniques” IEEE International conference on Electronics, Communication and Aerospace Technology (ICECA ,941–945,2019.

[40]. L. Taheri, A.F.A. Kadir, A. H.Lashkari, “Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls”, IEEE 2019 International Carnahan Conference on Security Technology (ICCST) 1–8,2019

[41]. K.Khariwal ,J.Singh ,A.Arora, “IPDroid: Android Malware Detection using Intents and Permissions.” IEEE Fourth World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4),197–202,2020

[42]. Y.C. Shyong , T.H. Jeng, Y.M. Chen, “Combining Static Permissions and Dynamic Packet Analysis to Improve Android Malware Detection”,IEEE International Conference on Computer Communication and the Internet, 75–81, 2020.

[43]. P.Feng , J.Ma ,T.Li ,X.Ma, N.Xi, “Android Malware Detection based on Call Graph via Graph Neural Network”, International Conference on Networking and Network Applications (NaNA),2020

[44]. W.C. Kuo,T.P.Liu, “Study on Android Hybrid Malware Detection Based on Machine Learning”, IEEE 4th International Conference on Computer and Communication Systems,31-35,2019

[45].https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/

[46]. https://developer.android.com/guide/platform

[47]. https://source.android.com/security/features