APP PERMISSION CLASSIFICATION: STATIC AND DYNAMIC METHODS

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
DATA SCIENCE
Submitted by

Praveen Singh Rawal
(2K21/DSC/08)
Under the supervision of
Dr. Divyashikha Sethia
Assistant Professor
Department of Software Engineering

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formely Delhi College of Engineering)
Bawana Road, Delhi-110042
May,2023

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi - 110042

CERTIFICATE

I certify that the Project Dissertation titled **App Permission Classification: Static and Dynamic Methods** which is submitted by **Praveen Singh Rawal (Roll No. 2K21/DSC/08)**, Department of Software Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the Project work carried out by the student 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.

[illegible]

Dr. Divyashikha Sethia
Assistant Professor,
Department of Software Engineering,
Delhi Technological University

Place: Delhi
Date:

DECLARATION

I, Pravven Singh Rawal, Roll No. 2K21/DSC/08, student of M.Tech(Data Science), hereby declare that the project dissertation titled App Permission Classification: Static and Dynamic Methods which is submitted by me to the Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not been submitted anywhere for the award of degree, diploma, fellowship or other similar title or recognition to the best of my knowledge.

Place: Delhi
Date: 13 May,2023

Praveen

Praveen Singh Rawal
2K21/DSC/08

ACKNOWLEDGEMENT

I'm extremely grateful to Dr. Divyashikha Sethia(Assistant Professor, Department of Software Engineering) and all the employees of the Department of Software Engineering at DTU. They all furnished us with enormous help and direction for the venture.

I might likewise want to offer my thanks to the University for giving us the research centers, framework, testing offices, and climate which permitted us to work with next to no checks.

I might likewise want to see the value in the help given to us by our lab partners, seniors, and our companion bunch who supported us with all the information they had regarding different subjects.

Praveen Singh Rawal
2K21/DSC/08

ABSTRACT

It is no secret that Android is one of the most widely used smartphone operating systems globally, boasting a staggering 2.5 billion active users. However, data security has become a crucial aspect of smartphone usage with the increasing reliance on smartphones to store sensitive personal information. Unfortunately, many apps tend to collect user data without the user's knowledge or consent, which can harm data security. To counter this, Android has included an inbuilt security feature called app permission to enable users to control app access. This feature enables users to grant or decline app access to various phone features such as camera, microphone, and location data.

The study proposes two methodology for classification app permissions. Firstly, static novel three-tiered system called APEC (App Permission Classification with Efficient Clustering).The study provides an insightful analysis of app permissions using a dataset of 2 million app permissions and app categories from the Google Play Store. The static novel three-tiered system called APEC (App Permission Classification with Efficient Clustering) aims to determine the safety of app permissions based on their usage frequency within specific app categories. This system categorizes apps into three levels, clustering, approval, and classification, to ensure users can select appropriate permissions and developers can establish the minimum requirements for their apps to function smoothly. To accomplish this, APEC uses DBSCAN clustering to group apps based on their respective categories and evaluate the safety of their permissions. Furthermore, it employs the Decision Tree and Random Forest machine learning algorithms to classify new app permissions as safe or unsafe. The proposed system achieved an impressive accuracy rating of 93.8% and 95.8% using the Decision Tree and Random Forest algorithms, respectively.

Secondly, a dynamic methodology App Permission Classification Dynamic Model (APCM) based on APEC with added features and capabilities. It keeps track of the app permissions requested by various apps using a dataset of 2 million apps from the Google Play Store to train a random forest-based model using DBSCAN clustering. The APCM analyses each permission request's frequency in a category and creates a frequency map accordingly. This model is critical in rating the app permission as safe or unsafe by using DBSCAN clustering and predicting using a random forest machine learning algorithm. The APCM further enhances the results by using PSO-BO optimization over the random forest, which is considered a highly accurate approach. The APCM provides a better experience and ensures that users can use apps safely without any concerns about their security. By analyzing the frequency of each permission request in a category, the APCM can identify safe and unsafe permissions and ensure that users can use apps without any worries. This model considers user preferences while classifying apps using the Dataset, ensuring a more personalized and satisfying user experience. The APCM has achieved an impressive accuracy of 87% in classifying an app's permission as safe or unsafe. Overall, the APCM is a highly innovative and effective approach that can help users ensure their apps are safe and secure. The use of DBSCAN clustering and random forest machine learning algorithms, and PSO-BO optimization ensure that the APCM is highly accurate and can provide a reliable classification of app permissions. This model is also highly user-friendly, ensuring users can use apps safely and securely without concerns about their security or data privacy.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview/Background

In the world of mobile technology, mobile markets play a critical role as distribution centres for new mobile applications. Daily, countless new apps are being developed and added to these platforms, providing people with various options. These apps have become integral to people's daily lives, whether for reading news, shopping, or other tasks. Among the various operating systems available, Android stands out for its user-friendly interface and simplicity, making it the most popular operating system worldwide, with over 2.5 billion users [2]. Google's App Store is the primary platform for Android apps, with over 2 million apps already published. One of the most remarkable features of this platform is that developers can create Android apps without needing certification or validation before releasing them on the store. It means that developers have immense freedom to create and publish apps, leading to many choices for users. Compared to the Apple App Store, the Google Play Store has fewer testing conditions for publishing an app. While Apple's App Store has a stringent review process, Google's Play Store tends to be more lenient. It is another reason many apps are available on the Google Play Store.

The Google Play Store is a popular platform for publishing apps, but this popularity also comes with its fair share of risks. One such risk is the potential for unauthorized data access. The more apps available on the store, the greater the likelihood of malware attacks. Unfortunately, monitoring the activities of all third-party apps on the store can be challenging. While the Google Play Store takes security measures and regularly removes apps that pose a threat, malware publishers often find new ways to republish their apps, creating an ongoing challenge for users and developers.

Android mandates that applications request permission before using specific data and functionality. However, everything is at the user's discretion to permit or restrict permissions [3]. In order to perform their essential functions, more than 70% smartphone apps ask for permission to collect data that is not immediately necessary. On the other hand, users frequently rush through the installation process and provide all permissions. Furthermore, even if users pay attention to the permission request, most lack the expertise to comprehend permissions and their possible consequences. In order to properly provide permissions and prevent privacy violations, practical permission suggestions offer users more information. There is no mechanism to help Android users choose privacy-conscious and safe Android applications. Users must manually verify permissions and rely on their understanding and knowledge of the community's comments and approvals, which rarely address security risks. Users might not anticipate that third-party apps will track their whereabouts (such as their homes or offices), and other private information (such as contact lists or SMS logs) is a significant privacy threat. For example, if a user uses a newspaper app, the app requests contact detail and camera permission which are of no use to the app, but the user ignores it, resulting in the leak of personal data. There are several cases of privacy threats in Android [4], including:

- *Malware:* Malicious software, or malware, can be installed on an Android device without the user's knowledge and collect sensitive information, such as login credentials and financial information.
- *Data Leakage:* Android apps can access a wide range of sensitive information, including contacts, location, and text messages, and potentially leak this information to third parties without the user's knowledge or consent.
- *Phishing:* Android users are the target of phishing attacks, which trick users into giving sensitive information, such as login credentials, to a fake website or app.
- *Location tracking:* Some apps access and track the user's location without their knowledge or consent.
- *Unwanted Ads:* Some Android apps have hidden ads that may pop up when users are not using that app. These ads are undesirable and may contain malicious links.

In Android apps, to access any hardware or software resources, the apps need permission beforehand from the user to access the resources. From Android version 6.0, Google introduced a feature called runtime permission [5]. This feature gives the user more control over the app's permission. There are some issues due to a need for more user awareness, user diligence, and risk assessment. In the Android ecosystem, some apps may request extra permissions that are not directly related to their core functionality. These additional permissions can pose a privacy threat if not correctly managed.

Some common examples of extra permissions that apps generally request include the following:

- *Access to the camera and microphone:* Some apps request permission to use the camera and microphone. These permission are not necessary for the functioning of the app. It enables the app to monitor or record users without their knowledge or consent.
- *Access to SMS and call logs:* Some apps seek access to the device's SMS and call logs. The app's primary functionality does not need this permission. It enables the program to gather personal information about the user's contacts and communication habits. It enables the developers to sell these data to ad agencies and gain profit without the user's knowledge.

- *Location access:* Some apps request permission to access the device's location, but it is not the app's requirement to function. It allows the app to track users' location without their knowledge or consent.
- *Contact access:* Some apps request permission to access the device's contact list, even though the absence of this information does not affect the application's functioning. It gives the app access to personal information about the user's contacts.

The latest Android 12 update has released changes in the permission management system of apps. However, the update sends the notification service for only a few specific permissions. The privacy threat can be induced by other permissions as well. New apps can also develop new malware techniques to threaten user privacy. Hence, there is a need for new services on Android that can alert the user for safe or unsafe and provide the user with a safe space to use applications.

Machine learning models can classify an app's permissions as safe or unsafe. The machine learning models can help in full vigilance of the device for any privacy threats. The previous studies categorize apps using three techniques: Static, dynamic, and hybrid. The static approach does not alter the Dataset and remains constant throughout the model. The dynamic approach periodically updates the Dataset with new data and updates the model. The dynamic approach is the most preferred for real-time service implementation in Android app permission management. The hybrid approach is a mixture of static and dynamic approaches. It inculcates the features of both static and dynamic approaches. The work introduces the APEC model based on a static approach to predict whether the app permissions are safe/unsafe due to low operation and maintenance costs compared to a dynamic approach.

1.2 Motivation

In the modern era, privacy has emerged as a vital issue that demands attention. It empowers individuals to have a say in using their personal information and ensures they are not subject to unwanted scrutiny or judgement. However, the frequency of data breaches has increased, which poses a considerable threat to the security and privacy of users. Privacy is critical in shielding individuals from potential hazards, including identity theft, fraud, stalking, and harassment. Keeping personal details confidential makes it much harder for malicious actors to exploit individuals.

Furthermore, privacy significantly promotes mental well-being by providing a space for introspection, personal contemplation, and relaxation. It enables individuals to rejuvenate, stay in command, and safeguard their mental health. Ultimately, privacy is essential to modern life, and its preservation should be a top priority.

Many Android app users have recently expressed concerns regarding privacy and data breaches. These concerns have led to a general feeling of insecurity among users uncomfortable with the extra permissions granted to these apps. It motivates us to develop a secure system to detect any privacy issues arising from these extra permissions. Our system allows users to control the amount and type of data they share with these apps. This way, users can ensure their data is safe and not misused. With our system, we can identify and eliminate unwanted permissions, making the Android app space much safer for everyone.

1.3 Research Gaps

To effectively classify app permissions, the current clustering model needs improvement. It enhances the model's ability to differentiate between different permission categories, resulting in more accurate classification. Improvements should focus on refining the clustering algorithm to optimize the accuracy of permission grouping. Additionally, the model should use feature engineering techniques to extend its capabilities.

Categorizing application permissions is a crucial aspect of user experience as it provides valuable information to users regarding the permission requests made by apps. However, it is equally important to understand how users perceive and interpret these permission categories. Various research studies investigate user preferences, comprehension levels, and decision-making processes. Such insights can significantly aid in developing more effective and user-friendly permission classification systems, ultimately enhancing the overall app experience for users.

Moreover, there has yet to be an approach that can dynamically classify app permissions. As a user, it is common to encounter permission requests while installing an app or accessing particular features. These permissions can range from accessing user cameras to using user location data. While granting these permissions at first may seem necessary, user preferences may change over time. Therefore, it is crucial to research ways that allow users to modify or revoke permissions after installation. Doing so can provide valuable insights into user privacy preferences and control, ultimately leading to a more secure and personalized user experience. It highlights the need for further research and development in this area to address the growing concerns of users regarding privacy and security in the digital world.

1.4 Problem Statement

The primary goal of this research is to provide an accurate and efficient system for categorizing and labelling the permissions requested by mobile applications. The purpose is to provide consumers with clear and intelligible information about the precise resources and functionality that an app requires access to, allowing for informed privacy and security decisions.

Another goal of dynamic app permission classification is to provide an adaptive and context-aware system that allows users to amend or cancel permissions after initial programme installation. The goal is to give consumers granular control over app permissions while addressing emerging privacy concerns and preserving user privacy and security. It also gives user preferences weightage, making the model more accurate.

1.5 Proposed Solution

To effectively and precisely classify and categorize app permissions, work will employ a machine learning-based approach. This approach will encompass techniques such as feature engineering, clustering algorithms, and user feedback. It will extract the relevant features necessary for capturing the distinguishing characteristics of the different types of permissions from the permission data. The DBSCAN clustering algorithm will group similar permissions based on the category, and it will subsequently rate their extracted features as safe or unsafe. The work aims to provide an effective and efficient app permission classification and categorization solution by employing this machine learning-based approach. The solution will increase user safety and security while using apps.

Static classification methods have been the go-to for classifying apps on the Google Play store for quite some time now. However, these methods have their limitations. One of the significant drawbacks of static classification is that it fails to adapt to new apps added to the store. It can be frustrating for users who want to find and download the latest apps but cannot do so because of the limitations of the classification system.

The work proposes a dynamic app classification model to update frequently, enhancing the user experience based on new app introductions and updates. Moreover, the dynamic model considers user feedback on app permissions, updating the model accordingly. It results in a seamless user experience that feels more human-like and less restrictive.

The dynamic model is far more robust and future-proof than static classification methods. It can adapt to the ever-changing landscape of the Google Play store, ensuring that users can easily find and download the latest apps. The dynamic model provides users a more personalized and responsive experience by leveraging user feedback and constantly updating itself.

Chapter 2

Literature Review

2.1 App Permissions in Android

The Android operating system incorporates various security features that protect user privacy and prevent unauthorized access to sensitive device resources. Among these features, app permissions are of particular importance. App permissions act as gatekeepers and control which apps can access specific device features, data, and functionalities. By explicitly defining and requesting permissions, Android ensures that apps cannot access sensitive resources without the user's knowledge and consent.

The user receives a list of required permissions when installing or updating an app. This list outlines the resources the app seeks to access, and the user can grant or deny these permissions based on their understanding and trust in the app. This user consent process is a critical aspect of app security, as it empowers users to take control of their data and limits the potential for malicious or unwanted access.

Overall, app permissions play a crucial role in maintaining the security and privacy of Android devices, and users are encouraged to carefully review and manage these permissions to ensure the safety of their data. In granting permissions, users can be confident that the app only accesses the resources it needs to function correctly. Conversely, by denying permissions, users can protect their data and privacy by preventing apps from accessing sensitive information or resources. The Android operating system is designed with rigorous permission enforcement measures to ensure that apps comply with defined permissions. This system requires apps to request permission to access protected resources explicitly. The app's access to that particular resource is restricted if the user does not grant permission. This mechanism is put in place to safeguard user data and device functionality from malicious external sources. Furthermore, the permission model is an essential privacy safeguard that prevents unauthorized access to sensitive user information such as contacts, location, or personal files.

Android has implemented a sandboxing security architecture to enhance the permission model further. This security sandboxing mechanism works by isolating apps from each other and the underlying system. This containment mechanism enhances security and reduces the risk of data breaches, identity theft, or misuse of personal information. Overall, the Android permission model and security sandboxing architecture work together to provide a secure and safe environment for users to operate their devices and apps confidently.

When using apps on an Android device, Android invokes specific permissions for the app to function correctly. These permissions can range from accessing basic device features to using sensitive user data. Typically, when an app is installed or updated, it will display a list of its required permissions. Users can then choose to either accept or reject these permissions. The permission in

Developers provide predefined permission types corresponding to specific device features or data in the Android operating system. For instance, an app may need access to the device's camera to capture photos or videos. Alternatively, it might require access to the user's contact list to read, write, or modify contact information. Other common app permission requests include access to the device's GPS or network-based location information, access to the microphone for recording audio or conducting voice-based operations, and the ability to send and receive SMS messages on behalf of the user. Finally, an app may also need access to the device's internal storage to read or write data. By granting these permissions, users allow apps to access the necessary features and data they need to provide a seamless user experience [5].

2.1.1 Types of permission in Android

In Android development, developers categorize permissions into different types to ensure the safety and security of users' data. These types include install-time permissions, runtime permissions, and special permissions. The type of permission granted to the app determines the scope of restricted data the app can access and the restricted actions it can perform. The level of protection afforded to each permission also depends on its type.

Install-time permissions: These are permissions that give the app limited access to restricted data or allow the app to perform restricted actions that minimally affect the system or other apps. Whenever a developer declares install-time permissions in the app, an app store presents an install-time permission notice to the user when they view an app's details page. The system automatically grants the app permission when the user installs the app. Android includes different sub-types of install-time permissions, including normal and signature permissions, each with unique characteristics and levels of access.

Normal Permissions: When utilizing an application, it is imperative to understand the various permissions that dictate the app's access level to data and actions. Users grant normal permissions for standard actions that do not significantly threaten their privacy or other apps.

Signature permissions: Signature permissions are granted to apps signed by the same certificate as the operating system or app that defines the permission. These permissions are crucial for apps with privileged services, such as those dedicated to autofill or VPN services. The system assigns a signature protection level to these permissions to ensure only authorized entities can access them. By implementing such measures, the system guarantees maximum safety and security for its users and their data.

Runtime permissions: Runtime permissions, also known as dangerous permissions, give the app additional access to restricted data or let the app perform restricted actions that more substantially affect the system and other apps. Therefore, the developer must request runtime permissions in the app before the developer can access the restricted data or perform restricted actions. Many runtime permissions access private user data that includes potentially sensitive information. Examples of private user data include location and contact information. The microphone and camera provide access to sensitive information. Therefore, the system helps the developer explain why the app accesses this information. The system assigns the dangerous protection level to runtime permissions.

Special permissions: Certain app operations require special permissions in order to function correctly. It is important to note that only the platform and OEMs can define these special permissions. These permissions safeguard access to highly impactful actions like drawing over other apps. Users can navigate to the Special app access page within their system settings to manage these permissions. This page lists user-toggleable operations, and developers implement many of them as special permissions. Users must exercise caution when granting special permissions to ensure the security and stability of their devices.

Permission groups: When considering app permissions, it is essential to understand that these permissions can be grouped into permission groups. A permission group is a collection of permissions logically related to one another. To provide an example, consider an application that requires the ability to send and receive SMS messages. As these permissions pertain to the app's interaction with SMS, they would likely belong to the same permission group.

One of the primary benefits of using permission groups is that they help to streamline the user experience. Rather than being presented with multiple prompts for each permission, the user can instead be shown a single prompt that encompasses all the permissions belonging to a particular group. Specifically, by grouping related permissions, the system can minimize the number of prompts the user must navigate when an app requests access to certain features.

It is unwise to assume that given permission will always be grouped with others that developers might expect. It is worth noting, however, that permissions can move between groups without warning. As such, it is essential to stay vigilant and pay close attention to the specific permissions requested by each app developer [5].

2.2 Machine Learning Algorithm

2.2.1 DBSCAN clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a widely-used clustering algorithm that is particularly useful for discovering clusters of irregular shapes and varying densities in spatial data. Unlike other clustering algorithms, DBSCAN does not make any assumptions about the shape of the clusters, making it a versatile tool for a wide range of applications.

The fundamental concept behind DBSCAN defines clusters in a dataset as dense regions of points separated by sparser regions. The algorithm requires two essential parameters: epsilon (ϵ) and minPts. Epsilon represents the maximum distance between two points to consider them as neighbours, while minPts indicates the minimum number of points required to form a dense region or cluster.

The DBSCAN algorithm works as follows: First, choose an unvisited data point randomly and retrieve its ϵ-neighbourhood, which consists of all the points within a distance ϵ of the chosen point. If the number of points in the ϵ-neighborhood is greater than or equal to minPts, the point is marked as a core point and assigned to a new cluster. If the number of points in the ϵ-neighborhood is less than minPts, the point is classified as noise and left unassigned.

Once the algorithm identifies a core point, it retrieves all the points in its ϵ-neighborhood and adds them to the same cluster. The algorithm continues to expand the cluster by recursively adding all the points within ϵ distance of each other until no more points can be added.

DBSCAN has several advantages over other clustering algorithms. It can identify clusters of any shape, including non-convex shapes, and handle datasets with varying densities. Furthermore, DBSCAN is robust to outliers and noise, which can be automatically identified and excluded from the clusters [6].

2.2.2 Decision Tree

A decision tree algorithm is a powerful machine-learning tool for classification and regression tasks. It involves constructing a tree-like model that makes decisions and their potential consequences based on the features of input data. The algorithm begins by selecting the best feature to divide the data using either Gini impurity or information gain as measures. The root node is chosen feature, and the Dataset divides into subsets based on the feature's values. This recursive splitting process continues until a stopping criterion is met, such as a maximum tree depth or a minimum number of examples in a node.

Decision trees use pruning to avoid overfitting and remove unnecessary nodes. At each leaf node of the tree, a prediction or decision is made based on the majority class for classification or the average value for regression. This decision represents the output of the decision tree.

One of the main advantages of decision trees is their ability to handle categorical and numerical features, making them versatile in addressing complex relationships between features. Additionally, decision trees are interpretable, allowing users to understand how the model arrived at its decisions. Overall, decision trees are a valuable tool in machine learning, offering a powerful means of data analysis and problem-solving [7].

2.2.3 Random Forest

Random Forest is a highly popular and effective ensemble learning algorithm widely used in various machine learning tasks, including classification and regression. Its primary objective is to combine multiple decision trees to create a more robust and accurate model.

Random Forest requires a labelled dataset in training and testing sets like any machine learning algorithm. Each example in the Dataset has a set of features and a corresponding target value. To achieve this, Random Forest follows a multi-step procedure, starting with data preparation.

Next, the algorithm builds an ensemble of decision trees. Each tree trains on a random subset of the training data, known as bootstrap sampling. Moreover, for each tree, a random subset of features is selected to split at each node, known as feature bagging or subsampling. This randomization helps in reducing overfitting and promotes diversity among the trees.

After the ensemble of decision trees, each tree trains individually. During this stage, the algorithm uses the training data provided for each tree to create a set of rules for each node in the tree. The prediction uses rules to predict the target value for new data points.

Finally, when making predictions on new data points, the algorithm aggregates the predictions of all the decision trees in the ensemble to make the final prediction. This approach helps reduce the variance and increase the model's accuracy as a whole [8].

2.2.4 Bayesian Optimisation

Bayesian optimization is an advanced and highly effective technique to identify the optimal set of parameters for an objective function that is difficult and expensive to evaluate. Bayesian inference and surrogate models are the basis of Bayesian Optimization, which helps to explore and exploit the parameter space intelligently, making it particularly useful when evaluating the objective function is time-consuming or resource-intensive.

To understand how Bayesian optimization works in detail, it is first necessary to define the objective function that needs to be optimized. It could be anything, ranging from the performance of a machine learning model to the accuracy of a simulation or any other metric that needs to be maximized or minimized. The next step in Bayesian optimization is to define the search space of possible parameter configurations. Parameter configuration specifies each parameter's range or set of values.

After defining the parameter space, the next step is to select an acquisition function, which determines the utility or "value" of evaluating a specific point in the parameter space. Many different acquisition functions can be used, but some of the most common ones include Probability of Improvement (PI), Expected Improvement (EI), and Upper Confidence Bound (UCB) [9].

2.2.5 Particle Swarm Optimisation

Particle Swarm Optimization (PSO) is a widely used optimization algorithm that draws inspiration from the collective behaviour of bird flocking or fish schooling. A population-based metaheuristic algorithm searches for the optimal solution in a multidimensional search space. The algorithm simulates the social behaviour of particles that move around the search space to find the global optimum.

The PSO algorithm starts by randomly initializing a population of particles in the search space. Each particle represents a potential solution to the optimization problem and is assigned a position and velocity. The algorithm calculates the position of each particle by computing the objective function value. The objective function determines the fitness or quality of a particle's solution.

After evaluating the position of each particle, the algorithm updates the velocity and position of each particle. The particles adjust their velocities based on their historical best position (pbest) and the best position found by any particle in the swarm (gbest). The velocity update equation is a weighted sum of the particle's current velocity, its historical best position, and the best position found by any particle in the swarm. The position of each particle is then updated based on its new velocity.

The PSO algorithm iterates until it reaches a stopping criterion. The stopping criterion can be a maximum number of iterations, a target fitness value, or a user-specified time limit. Optimization problems such as parameter tuning, feature selection, and function optimization use this algorithm [10].

2.2.6 Kafka and Hadoop

Apache Kafka is a powerful and widely-used distributed streaming platform that can handle high-throughput, fault-tolerant, and real-time data streams. One of the critical benefits of Kafka is its reliability and scalability as a message queue or event streaming platform, which ensures that data is efficiently processed and delivered in a fault-tolerant manner. It makes Kafka an ideal solution for organizations that need to handle large volumes of data and want to ensure that their data processing and delivery systems are both reliable and efficient.

One of the critical features of Kafka is its publish-subscribe model, which allows data to be produced by various sources and consumed by multiple applications or services. Organizations can easily share data across different departments, teams, or companies. Kafka also provides powerful tools and features for managing and monitoring data streams, including support for real-time data processing, automatic data partitioning, and fine-grained access control [11].

Apache Hadoop is an open-source framework that enables distributed storage and processing of large datasets across clusters of computers. At its core, Hadoop consists of two primary components: Hadoop Distributed File System (HDFS) and Hadoop MapReduce.

HDFS provides a distributed file system that can store massive amounts of data across multiple machines, making it an ideal solution for organizations that need to store and manage large datasets. Hadoop also provides various tools and features for managing and processing data, including support for distributed data processing, automatic data replication, and fine-grained access control [12].

Chapter 3

Related Work

Park et al. (2018). [13] suggest categorizing applications into benign, suspicious, and malicious categories based on their APIs and permissions. Level 1 includes overviews of the network and system. Level 2 compares the permissions listed in the classification. Level 3 matches the API's interface, class, public method, and permissions listed in the Level 2 classification. The authors developed the Yet Another Recursive Acronym (YARA) Rules API, based on which they created a permission-based classification system. They extracted the API, class, and public methods from each application's Android Manifest.xml and classes.dex files and matched them with YARA Rules. They eventually increase users' awareness by informing them of the application's behaviours and allowing them to decide whether to download it to their devices.

Niu et al. (2018) [14] provide a dynamic Android permission management method based on machine learning to enhance the permission approval mechanisms used by Android. It provides dynamic administration by maintaining an active entitlement management database that tracks the status of each program entitlement and can only use the permissions granted in the database. The proposed model uses machine learning to classify programs, collect relevant data from databases, and inform users of dangerous permissions. It finally changes the dynamic management database in accordance with the user's choice [14]. Users can stop unauthorized users from accessing private data and sensitive APIs in real-time in this way. This solution broadens authentication management options while enhancing multimedia data's security and dependability on Android devices.

Due to the volume of data that smartphone apps may access, platforms implement permission systems that allow users to control how applications access protected resources. If people are repeatedly requested to make security judgments in harmless circumstances, they may become accustomed to it and approve all future requests without considering the repercussions. To investigate, Wijesekera et al.(2015) [15] analyzed the Android ecosystem and how frequently an Android app collects user data. The paper also analyzed under what circumstances the apps access the data from permission-controlled resources. They experimented with thirty-six volunteers to investigate the frequency of apps that access the protected resources when users do not expect them. The researchers asked the volunteers about the condition in which they would like to control access to protected resources. They found that 80% of the volunteers wanted to prohibit at least a single permission request and that more than a third of the requests were invasive and should be blocked. [15]

Sun et al. (2016). [16] proposed the Significant Permission Identification for Android Malware Detection (SIGPID) method that identifies the crucial subset of the permissions to overcome the malware problem on Android. It accurately identified Android malware detection using significant permissions. These outcomes are favourable compared to those obtained by a method that employs both the dangerous permission list and all 135 permissions. Compared to other cutting-edge malware detection techniques, the SIGPID is more effective. For malicious apps, it shows an accuracy of 93.62%. The authors introduced unknown malware apps to test the model, and SIGPID showed an accuracy of 91.4% in classifying them. SIGPID, when applied to a larger dataset and the most popular sixty-seven supervised machine learning algorithms with a dataset with 310,926 benign labelled apps and 5,494 known malicious apps. The research shows that out of sixty-seven supervised machine learning algorithms, fifty-five achieved an F-measure of a minimum of 85% [16]. F-measure is a performance measure defined as the harmonic mean of precision and recall. It allows the model to evaluate precision and recall using a single score.

SAMADroid is an innovative three-level hybrid malware detection model for the Android operating system proposed by Arshad et al. (2018) [17]. The authors examine and categorize detection methods for several existing Android malware detection approaches. They developed an innovative three-layer hybrid malware detection model for the Android operating system that combines the advantages of three layers: 1) static and dynamic analysis, 2) local and remote host, and 3) machine learning intelligence. Experimental data shows that SAMADroid achieves high malware detection accuracy by ensuring power consumption and memory efficiency [17].

Moussa et al.,(2017) [18] propose to Android Confidentiality Concern User Support Environment (ACCUSE) in this work, a strategy targeted at assisting users in selecting apps based on their confidentiality concerns. It enables the user to assess the danger of installing an app quickly and visually. The work defines three risk factors by combining the frequencies of app necessary permissions in the three classifications. ACCUSE tests on a real-world dataset of 11,576 Android apps and a baseline of approximately 1000 known malware apps. The findings reveal that ACCUSE consistently assigns high risk to known malware programs and exceeds the state-of-the-art [18].

Lindorfer et al.,(2015) [19] propose MARVIN as a system that uses machine learning approaches for calculating the risk of unknown Android apps. They proposed a malicious score using a combination of static and dynamic approaches. They have created one of the most extensive datasets for Android malware classification dataset containing a hundred and thirty-five thousand applications. They added labelled samples of malware apps to the Dataset. They calculated the retrieval interval for updating the Dataset for the model's long-term sustainability. They used MARVIN to have an accuracy of over 98.24% [19].

Through their research, Roy et al.(2020) [20] effectively utilized static analysis to link each app call to a specific feature, subsequently determining the frequency of occurrence per feature by aggregating the results. The study successfully highlighted the temporal bias present in previous malware detection methods and presented a lightweight approach to address it. Utilizing the DREBIN dataset of 972 malicious and 1100 benign apps, the researchers reduced the feature set to 50 using NMF, making the model both fast and lightweight for smartphones. The SVM classifier demonstrated remarkable accuracy of 88.72% in accurately classifying malicious and benign applications. However, the small feature set posed a disadvantage, resulting in low accuracy in detecting malware apps, ultimately limiting its practical application in the industry, particularly in handling new malware variations.

Manzil et al.(2023) [21]introduces a Hufman encoding-based feature vector generation technique to classify and detect banking, adware and sms malware. It uses the frequency of system call as features using machine learning and deep learning (MLP and CNN). The method gives an accuracy of 98.7% using Random Forest.

Martin et al.(2018) [22] developed CANDYMAN, a highly sophisticated tool that employs dynamic analysis and Markov chains to classify Android malware families accurately. The approach CANDYMAN uses entails extracting crucial information from a given malware sample in the form of a series of states through dynamic analysis. CANDYMAN uses a Markov chain to model the transition probabilities between sequence states. Features in the classification process use the probabilities from Markov chains, which helps to identify and classify malware samples accurately. In order to test the proposed technique, CANDYMAN uses the space of features to train traditional Machine Learning, including methods for unbalanced learning and Deep Learning algorithms, across a dataset of malware samples from various families. Unbalanced learning was included in the training process to ensure the classifications were as accurate as possible.

Kabakus et al.(2018) [23] conducted a study that presents mad4a, a novel and highly effective method for analyzing Android malware by integrating static and dynamic analysis techniques. By scrutinizing app permissions and network logs, this approach uncovers the distinctive features of Android malware. The study's findings indicate that malicious programs have a much higher probability of possessing overprivileged permissions than benign applications, with the malware app permission being eleven times more prevalent than benign. It suggests that the Android ecosystem is vulnerable to attacks by malware that exploit app permissions to gain control over sensitive user data.

Alzaylaee et al.(2017) [24] propose a hybrid test input generation approach combining two tools - a state-based tool called DroidBot and a random-based tool known as Monkey. This system significantly improves code coverage and the detection of harmful behaviours in real-world devices. To prove the effectiveness of this approach, the team conducted extensive testing on 2,444 Android apps from the Android malware genome project, including both benign and malware samples. The testing results were promising, with the hybrid system proving to be highly effective in improving code coverage and detecting harmful behaviours. The method significantly enhances dynamic analysis code coverage and Android malware detection. Table 5.2 compares previous related works for their significant contribution and limitations. The previous works show that the Dataset used was very small compared to the variation of the apps in the Google Play Store and was not justifiable. Also, the primary focus of the previous works was on malware detection but not on the privacy concerns that arose from the apps' extra permission requests. Hence a new model is required to address the issue of privacy and data misuse due to the extra permission requests by the apps.

Table 3.1: Comparison Of Previous Work

Research Work	Type	Approach	Contribution	Limitation
Niu et al., (2018) [14]	Classification	Static	According to the relevance of the feature elements, the model converts the APK file into a multidimensional vector.	The model is limited to 2200 applications and selected from the YingYongBa store end
Arshad et al., (2018) [17]	Malware Detection	Static and Dynamic	In terms of static analysis, it outperforms Drebin dataset It has also been found that SAMADroid has a minor performance overhead on the Android smartphone.	At the local host, no malicious activity detection is done. As a result, if the network link fails or there is congestion at Channel, the detection has been adhered too.
Sun et al., (2017) [16]	Malware Detection	Static	Multi-level Data Pruning	The model only considers 23 permissions out of 122 permissions. The reduction can result in false classification.
Lindorfer et al., (2015) [19]	Malware Detection	Static	Employs machine learning techniques and a cross of static and dynamic approach to evaluate the risk posed by untested Android apps.	Marvin does not focus on permission being safe or unsafe but on whether the app is malicious or not.

Chapter 4

Proposed Work

4.1 Preprocessing

4.1.1 Dataset

The Dataset consists of 2 million app data [25]. It contains the names of the apps' permissions declared in their respective manifest files. The Dataset does not contain any information on which category the app belongs. The category is an essential factor in detecting app permission usage patterns. Adding the category becomes essential, so we scrape the category from the Google Play Store using the app name and app ID. The final Dataset merges the 2 million and the category datasets prepared, merging both datasets based on the app i.d. The Final Dataset now contains all the related data.

4.1.2 Feature Extraction

Feature Extraction is an integral part of the data preparation to get good results and accurate predictions. The Dataset contains the app name, category, and permissions requirement of the app. The app permission names are in text, and the text is not processed in machine learning efficiently. The model creates a frequency map in which each unique app permission converts to a column. For each app permission the app uses, the model gives it a value of '1', and permission not used gives a value of '0'. The data contains a category of the app and all app groups according to category. The data includes the apps, but we must include frequency data according to the category. The model calculates the sum of each unique app permission in each category and creates the final Dataset.

4.2 Proposed static APEC model

This work proposes a novel three-tier model APEC (App Permission classification with Efficient Clustering), which uses the static classification model to categorize the permissions as safe and warning using a three-tier architecture based on the app's frequency of permissions. It uses a 2 million dataset comprising the name of the app and all app permissions requested by the applications. The three-tier architecture, as shown in Fig. 4.4 comprises:

- Group category

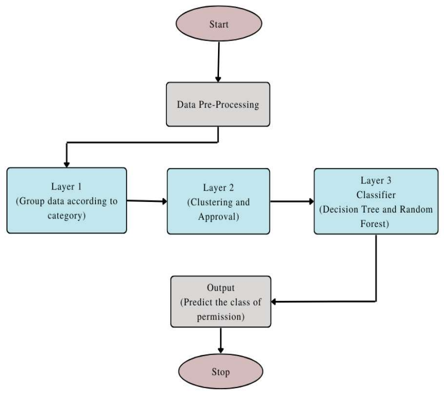

Start
Data Pre-Processing
Layer 1 (Group data according to category)
Layer 2 (Clustering and Approval)
Layer 3 Classifier (Decision Tree and Random Forest)
Output (Predict the class of permission)
Stop

Figure 4.1: Flow diagram of APEC

- Clustering and Approval
- Classifier

4.2.1 Group Category

All apps of a category have similar requirements and permission required to function [26]. The apps in the Dataset are scattered randomly. APEC groups the app based on the category it belongs to. APEC calculates the sum of each unique permission in every category on the basis requested by the apps. APEC creates a frequency map of permissions based on the category containing the sum of each permission.

4.2.2 Clustering and Approval

APEC calculates how many permissions each program has in each group under a particular category. When comparing the minimum and maximum numbers, the sum of the unique permissions exhibits significant fluctuation. Normalizing the sum of the permissions for each unique category helps remove bias. The K-means algorithm and DBSCAN (density-based spatial clustering of applications with noise) are the most popular unsupervised clustering methods. The K-means clustering is effective in the construction of more than one cluster. K-means becomes ineffective if there is only one cluster, hence not considering the K-means algorithm. In APEC, the cluster is all unique categories taken individually. The DBSCAN operates in a density model where the cluster is made based on the density of the point near the core, which is ideal for single cluster clustering and fits the model's requirement.

Fig. 4.2 shows the k-distance graph of the action category. It shows the line approximately curves at a value of $\epsilon = 20$. Hence, in DBSCAN clustering, the ϵ value will be in the neighbourhood of 20 to give an optimal clustering of the app permissions of the action category. The hyperparameter essential for DBSCAN clustering to work is epsilon ϵ. The Epsilon is the max radius of the neighbourhood. In this case, it is the number of most frequent app permission requested by the applications. The k-distance graph calculates the optimal epsilon using different epsilon values to test. When the line curves, the point is considered the optimal epsilon value, and the cluster uses it for evaluation.

Algorithm 1 Algorithm for DBSCAN

```
1: Identify the core points with more neighbours than minPts by finding the points
   in the $\epsilon$(eps) neighbourhood.
2: On the neighbour graph, find the associated components of core points while
   ignoring any non-core points.
3: Allocate every non-core point to the cluster if it is a $\epsilon$(eps) neighbour; if not,
   assign it to noise.
4: Test various values of $\epsilon$(eps) and minPts, and the optimal values form the best
   cluster.
5: Calculate the k-distance for each category. Plot the graph of k-distance values,
   and on the graph, find the point where the line curves offer the best $\epsilon$(eps)
   neighbour value.
6: Calculate the k-distance for each app category and update clusters based on the
   value of k.
```

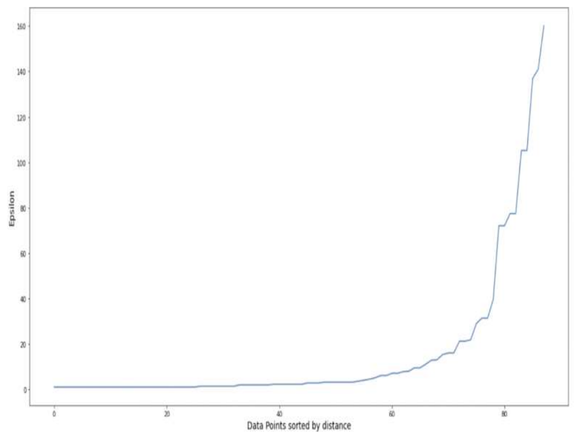


Figure 4.2: K-distance graph of Action category

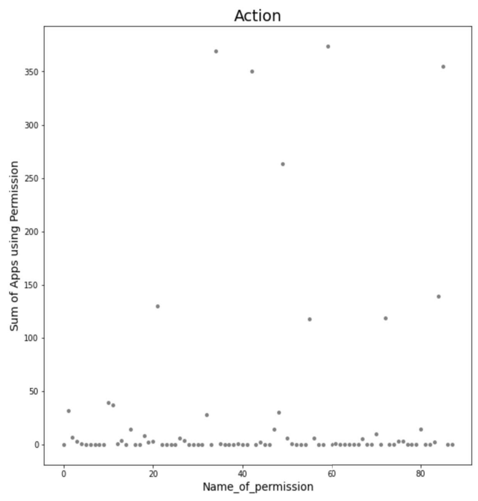


Figure 4.3: DBSCAN clustering of Action category

APEC uses clusters to assign rank based on how frequently apps in each category request permission. The cluster core contains the frequently requested app permissions, and the outlier has fewer requested permissions. The approval is represented by '1' for the cluster's core, showing that the permission is safe, and an unsafe '0' for the cluster's outlier, showing each category's rare permissions requests. The clustering gives the Dataset the ground truth values. The clustering, the unsupervised learning method, does not need prior knowledge and recognizes the common pattern to produce the ground truth values. It gives the approval data from the frequency of the app permissions of each category.

4.2.3 Classifier

Google Play Store publishes new apps daily, and it is hard to keep track of the permissions requests made by the latest apps. The classifier rates the permissions of new apps based on category using the previous category data. The classifier suggests to developers the most appropriate permission for the smooth functioning of the app belonging to a category. The model uses the Decision Tree and Random Forest classifiers to compare a two-dimensional feature containing the approval and the category to which the approval belongs.

4.3 Proposed dynamic APCM method

Fig.4.4 shows the flow diagram of the APCM model comprising of Clustering, Classification, Optimization, Data Streaming, Database and Updation.

4.3.1 Clustering

The Dataset initially does not contain information on whether the app permission is safe or unsafe. The model uses DBSCAN clustering in which the cluster's core is the most used app permission in the category and marks them as safe. The app permission that are outliers of the cluster are used less in the category, and the probability of them being unsafe is very much, so the model marks them as unsafe. DBSCAN clustering needs the k value to be adjusted to get an optimal core of the cluster.

4.3.2 Classification

Introducing the new app daily in the google play store introduces new patterns that the previous data library cannot handle. Hence, a machine learning algorithm must find the pattern and help predict the new app permission class. The classification uses a decision tree and random forest due to the multi-level data. However, the decision could perform better than the random forest algorithm in the initial model testing, so the model uses only random forest for prediction.

4.3.3 Optimization

PSO [10] has caught the attention of many academics, resulting in various variants of the core algorithm and numerous parameter automation schemes. To optimize the function, particle swarm optimization (PSO) requires no gradient knowledge. It is conceptually elementary and uses only primitive mathematical operators. A swarm of particles populated with several random potential solutions is a constituent of classic PSO.

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (p_b(t) x_i(t)) + c_2 r_2 (g_b(t) x_i(t)) \tag{4.1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4.2}$$

where g_b is the global best value, p_b is the personal best of the loop, ω is inertia weight coefficient, c_1 and c_2 are the learning factors, r_1 and r_2 are random variables, v_i is the velocity of the particles and x_i is the particle position.

PSO is generally alone and cannot adjust the hyperparameter values. So, Bayesian optimization determines the optimal value of the function under the assumption that the known finite sample points are present by creating the probability of the function's output. Bayesian optimization is very data efficient and used when function calculation costs are very high.

$$x = argmaxf(x) \tag{4.3}$$

where f(x) is the objective function.

The model uses Particle swarm optimization [10] combined with Bayesian optimization [9] over the random forest to optimize it for the prediction. It adjusts the

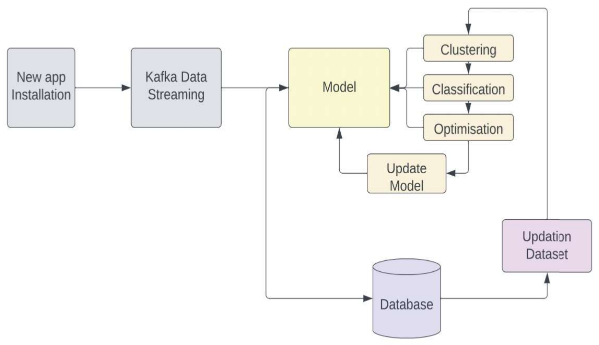


Figure 4.4: Flow diagram of APCM

hyperparameters in the prediction model to produce efficient results. The optimization also works at the updation step. When the model updates, it optimizes the hyperparameters to produce efficient results.

Algorithm 2 Algorithm for PSO-BO

```
1: Initialize the hyperparameter $x_0$.
2: For t = 1, 2, ..., T do:
3: Initialize random position and velocity of particles of the population.
4: Evaluate the fitness function of each particle.
5: Compare Particle's $p_b$ with fitness calculated. If fitness exceeds $p_b$ then replace $p_b$.
6: Compare global best($g_b$) with current value. If the value exceeds $g_b$, replace $g_b$ with the current value.
7: Calculate the value of position and velocity using eq(1) and eq(2).
8: Repeat from Step 4 until the exit criterion is met.
9: Maximize function and evaluate next point:$x_{t+1}$
10: Evaluate objective function value.
11: Update data.
12: End for
```

4.3.4 Data Streaming

The model uses Kafka [11] as the data streaming service to introduce new data to the model for dynamic analysis. Kafka can handle a large amount of data in real time. Kafka also features low latency, high throughput, and high scalability, making it reliable for data streaming services. The model uses Deephaven [27] as the interface to test the model's capability to work as a dynamic model. Deephaven is highly configurable and supports the Kafka API. Deephaven links to a physical database to store the results for the updation of the model. Deephaven supports machine learning algorithms and fits perfectly for use in the proposed model.

4.3.5 Database

The model uses Apache Hadoop [12] as the choice for the database to store the results predicted by the model for the updation of the classification model. The Apache Hadoop is highly scalable and allows distributed data processing, increasing the data's security and the model's efficiency.

4.3.6 Updation

The new data introduction reduces the efficiency of the model. The new data is also unknown to the machine learning model. The database stores newly introduced data with the results and the changes made by the informed user. The new Dataset again trains the classification model and updates the existing model. The updation process repeats itself periodically to make the relevant changes. This process makes the model robust and relevant for the new app introduction.

Chapter 5

Result

5.1 Proposed static APEC method

The clustering algorithm is an unsupervised learning methodology. Our model uses the DBSCAN clustering algorithm to rate the apps. Validation, or the objective and quantitative assessment of clustering findings, is one of the clustering process' most challenging elements. Vendramin et al. (2010) [28] have developed various relative validity criteria for validating clusters. However, global clusters do not make up all data. Density-based clustering algorithms look for partitions with low-density areas that include noise objects and high-density areas containing points (clusters, not necessarily global). Relative validity indices developed for global cluster validation may fail in certain instances.

To avoid the biases in clustering, the work takes a few steps while implementing the DBSCAN algorithm.

- To calculate individual values of ϵ and minPts for every category, the work uses the k-distance graph to increase the reliability of clustering.
- The work implements Density-Based Clustering Validation (DBCV) by Moulavi et al. (2014) [29], explicitly designed for DBSCAN clustering to validate the results. The data is unlabelled. Hence the model uses the internal indices to give a score of clustering.
- Further validation of the result of clustering is done by manually verifying the approval of permission for every category.

The cluster with validated results gives the highest accuracy in assigning the approval to the permission. The validation decreases false true cases, making the model more robust.

Fig. 4.3 shows the DBSCAN clustering of app permissions of the action category of apps. The points represent the frequency of the different app permissions used in the action category apps. The decision of permissions approval (safe/unsafe) is according to the permissions' frequency of the action category apps.

The classification model uses an 80-20 split of the 2 million app dataset. APEC train the models with 80% of the Dataset and tests them with 20% of the Dataset. To predict the class of permissions from 100,000 app data points, the classification

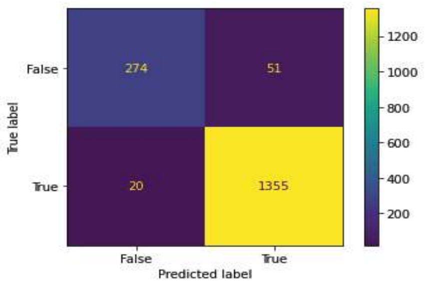


Figure 5.1: Confusion matrix of APEC

Table 5.1: Comparison of accuracy of APEC with related work

Model	Accuracy	Type
APEC	95.8%	App permission
Niu et al. (2018) [14]	93.5%	
Arshad et al.(2018) [17]	93.62%	Malware detection
Sun et al.(2017) [16]	93.62%	
Lindorfer et al.(2015) [19]	98.24%	

model employs the Decision Tree and Random Forest algorithms. The hyperparameters need to be adjusted to give the best-performing versions of the model. The decision tree model adjusts max_depth and max_leaf; in the random forest, max_depth, and min_sample_split. APEC adjusts the hyperparameters by using random values and running the algorithm till the accuracy decreases linearly. On the testing, the Decision Tree achieves an accuracy of 93.8% for permission classification, and Random Forest achieves an accuracy of 95.8% for permission classification. Table 5.1 compares APEC's accuracy with the previous work. With 95.8 % accuracy, the APEC leads the classification class algorithms. It performs well in comparison to the malware detection algorithm as well.

Only malware patterns and common traits are found in apps by malware detection algorithms. However, the presence of privacy threat permissions in a non-malicious application is also possible, which the malware detection algorithm cannot detect. The APEC can detect the application's privacy and extra permissions and outperform malicious algorithms. A large dataset also plays an essential role in APEC performance. The larger the Dataset, the more significant the algorithm encountered various cases while training, resulting in more robust results. Fig.5.1 shows the confusion matrix using the random forest method. The performance evaluation of the classification model uses other metrics, as accuracy can often be subjective and a false representation of the results. Here precision and recall are metrics to evaluate the correct prediction of the classifier using the confusion matrix in Fig.5.1.

On calculation, the precision is 0.985, and the recall is 0.9637. These metrics show that the classification model correctly classifies the app permission as safe/unsafe, with high precision and recall value. It signifies that 98% of the predicted safe permissions are actual safe permissions, and 96% of the total safe permissions are predicted as safe using the model.

The classification empowers the user to enable and disable the extra permissions requested by the apps. The approach also protects the honest developer, and the model tells the developer the most popular apps in the category use which app permissions. The model also helps determine the minimal permissions required for the app to perform smoothly in a category.

A sample app from the Google app store is used to test the model. The app belongs to the business category. The app acts as a personal sales tool, letting the user know whether the user is at the desk or on the move. Streamline user workday by adopting a mobile system. The app handles user requirements on the go and helps boost mobile sales productivity with features like search, call, email, check-in, locate leads nearby, and notifications. The app is perfect for field sales with a highly interactive, user-friendly interface, offline access, and the ability to sync data across all user devices automatically. According to the permission requested by the app, a few concerning permissions are likely to produce privacy concerns if not handled carefully. A few risky permissions are asked: find accounts on the device, record audio, and read contacts on the mobile device. The APEC model using the sample app data is used to predict the safe and unsafe permissions, where the safe permission is '1', and the unsafe permission is '0'.

5.1.1 Comparison with Android 12 update [1]

Android released a new update with significant changes in the app permission management in Android. Here is a comparison of updates in the app permission management with the proposed model.

- When an app uses the microphone or camera on an Android 12 or higher version, an icon displays in the status bar. Compared to Android 12, the proposed model does not restrict the notifications to only the microphone and camera. It specifies permissions accessed by the app which are not commonly used by apps in the selected category, ensuring robust security for the user.

- The Android 12 developer can query how the system organizes platform-provided permissions into permission groups. The platform-provided permissions are only a tiny part of the permissions accessed by the apps. The runtime and special permissions also act as a threat to the user data. The proposed model ensures the type of permission does not affect the permission approval as the platform-provided are necessary for the fluid function of the apps. The apps in the category will use that permission with high frequency.

5.2 Proposed dynamic APCM method

The cluster with validated results has the highest accuracy in assigning authorization approval. Validation reduces false positives, making the model more resilient.

The prediction model uses random forest. The training of the random forest uses 70 per cent of the data. The testing phase uses 30 per cent of the data. The testing data is divided into three parts to emulate a dynamic model interface. In testing the update function, the parts of the 30 per cent data are updated periodically in the model. The combination of all the data gives the final result.

Table 5.2: Comparison of different methods on APCM

Model	Accuracy	Precision	Recall
APCM without optimisation	82%	86.9%	86.5%
APCM with BO optimisation	85%	89.7%	89.4%
APCM with PSO-BO optimisation	87%	92.3%	90.8%

5.2.1 The APCM model without the optimization

The model uses 70 per cent of data training data to train the model. The model uses 30 per cent of the data for testing. Kafka then feeds the 10 per cent data to the model, which is 200,000 app data, and the model gives an accuracy of 80 per cent. The model stores the data in the database, and the updation step updates the model using the data. The next part of the data again follows the same steps. The model achieves an accuracy is 81.5 per cent. The APCM initiates the updation stage. It retrains the model with the new data and updates the classification model with the new data.

The model introduces the final half of the Dataset, giving an accuracy of 82 per cent. The pattern in the testing phase; after introducing and updating new data in the model, the model becomes more efficient. Compared to the industry standard accuracy, the model could be more accurate.

5.2.2 The APCM model with the BO optimization :

Bayesian optimization uses a probability model on the function to produce efficient hyperparameters. The model uses Bayesian optimization to optimize the hyperparameters of the random forest. The model's training uses 70 per cent of the dataset stream by Kafka. The model trains with Bayesian optimization optimizing the hyperparameters. The model tests 10 per cent of the data, and the model achieves an accuracy of 83 per cent. Testing the model using the 10 per cent remaining 30 per cent of data gives an accuracy of 82.5 per cent. Now the model retrains using the additional 10 per cent of data used for testing. Now the model retrains on testing data. The model again tests on 10 per cent of the remaining data. The model gives a final accuracy of 85 per cent.

5.2.3 The APCM model with the PSO-BO optimization :

The model performs well with BO optimization, but there is scope for improvement. There is evidence that PSO with BO optimizes well in Comparison to only BO. The PSO is particle swarm optimization which optimizes based on the natural phenomenon. The model with PSO-BO optimization trains on 70 per cent of the Dataset. The model gives accuracy in Comparison with the industry standards. The model tests on 10 per cent of the data on the first test. The model gives an accuracy of 84 per cent. The model retrains on the training data used and tests with 10% of the data. The model gives an accuracy of 86 per cent. On retraining, the model in final testing gives an accuracy of 87 per cent.

TABLE 5.2 compares different optimization methods used by APCM. PSO-BO optimization performs the best. TABLE5.3 compares the related work and APCM.

Table 5.3: Comparison of accuracy of APCM with related work

Model	Type	Accuracy	Precision	Recall
APCM with PSO-BO optimisation	Dynamic	87%	92.3%	90.8%
Martin et al. (2018) [22]	Dynamic	76.1%	75.5%	76%
Lindorfer et al.(2015) [19]	Static	98.24%	-	-
Arindaam Roy et al. (2020) [20]	Static	88.7%	89.5%	81.9%
Manzil et al. (2023) [21]	Dynamic	98.7%	98.7%	98.7%

APCM outperforms many dynamic and static methods. The performance of APCM is low compared to [21]; since the latter uses methods on malware app detection, APCM works on the app permission classification problem with significant variability in the Dataset.

Chapter 6

Conclusion and Future Work

Ensuring a safe environment on Android devices is crucial, and app permissions play a vital role in protecting users' privacy. However, excessive permission requests can cause significant privacy concerns. The proposed methodology rates app permissions and suggests the most appropriate permission for users and developers. This methodology, called App Permission Classification with Efficient Clustering (APEC), employs a novel three-tier architecture to categorize app permissions based on their frequency. This categorization helps identify the apps' category and recommend the appropriate permission for their smooth functioning. APEC achieves an accuracy of 93.8% using the decision tree and 95.8% using the random forest in predicting the app permission as safe and unsafe. Moreover, APEC is a user recommendation system that predicts extra permission based on the category trend. On the developers' end, this methodology helps them to minimize the required permission by analyzing the trends in the category. Thus, APEC is an efficient tool that ensures privacy and security while providing a seamless user experience.

The work presents a unique and practical approach to enhancing the safety of the digital ecosystem. The proposed App Permission Classification dynamic Model (APCM) operates with the user to classify apps and evaluate their permission requests. The model utilizes a frequency map and DBSCAN clustering to rate each app permission request based on the category. To assist the user in selecting safe permission, APCM deploys the random forest algorithm with PSO-BO optimization. One of the significant advantages of APCM is its dynamic nature, which allows it to update data periodically using KAFKA and Apache Hadoop database. This feature ensures that the model remains future-safe and can accommodate newly published apps seamlessly. In addition to considering user preferences, APCM also updates the model based on user suggestions, providing a smooth experience for the user. APCM evaluates accuracy using random forest with PSO-BO optimization, achieving an impressive accuracy rate of 87%. Overall, this innovative approach is a promising solution for improving the security of the digital ecosystem, demonstrating how dynamic methods can work symbiotically with users to create a more secure online environment.

Developing an app or service for Android using the model can effectively reduce security threats for Android users. However, the current research cannot suggest app permissions in real-time, which can hinder users seeking a seamless and secure experience. We can implement a real-time approach with the model, which provides users with real-time suggestions during app installation. Users can make informed decisions regarding app permissions and ensure that user data remains protected. Furthermore, the model can also be utilized as a security layer in the Google Play Store while publishing apps. This feature can enable the scanning of app permissions demanded by the developer and ensure that only safe and secure apps are available to users. Overall, implementing APCM can enhance Android devices' security and provide users with a much smoother and more secure user environment.

Bibliography

[1] Google, “Features and APIs Overview,” Oct. 2021. [Online]. Available: https://developer.android.com/about/versions/12/features

[2] “IDC - Smartphone Market Share - Market Share.” [Online]. Available: https://www.idc.com/promo/smartphone-market-share

[3] S. Pichai, “For the next five billion: Android One,” Sep. 2014. [Online]. Available: https://blog.google/products/android/for-next-five-billion-android-one/

[4] Y. Yang, X. Du, and Z. Yang, “Pradroid: Privacy risk assessment for android applications,” in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*. IEEE, 2021, pp. 90–95.

[5] “Permissions on Android.” [Online]. Available: https://developer.android.com/guide/topics/permissions/overview

[6] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, “DBSCAN: Past, present and future,” in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, Feb. 2014, pp. 232–238.

[7] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, “An introduction to decision tree modeling,” *Journal of Chemometrics*, vol. 18, no. 6, pp. 275–285, Jun. 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/cem.873

[8] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[9] P. I. Frazier, “Bayesian optimization,” in *Recent advances in optimization and modeling of contemporary problems*. Informs, 2018, pp. 255–278.

[10] F. Marini and B. Walczak, “Particle swarm optimization (PSO). A tutorial,” *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015.

[11] “Apache Kafka.” [Online]. Available: https://kafka.apache.org/

[12] “Apache Hadoop.” [Online]. Available: https://hadoop.apache.org/

[13] J. Park, H. Chun, and S. Jung, “API and permission-based classification system for Android malware analysis,” in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 930–935.

[14] S. Niu, R. Huang, W. Chen, Y. Xue, and others, "An improved permission management scheme of android application based on machine learning," *Security and Communication Networks*, vol. 2018, 2018.

[15] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, "Android permissions remystified: A field study on contextual integrity," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 499–514.

[16] L. Sun, Z. Li, Q. Yan, W. Srisa-an, and Y. Pan, "SigPID: significant permission identification for android malware detection," in *2016 11th international conference on malicious and unwanted software (MALWARE)*. IEEE, 2016, pp. 1–8.

[17] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: a novel 3-level hybrid malware detection model for android operating system," *IEEE Access*, vol. 6, pp. 4321–4339, 2018.

[18] M. Moussa, M. Di Penta, G. Antoniol, and G. Beltrame, "ACCUSE: Helping users to minimize android app privacy concerns," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 2017, pp. 144–148.

[19] M. Lindorfer, M. Neugschwandtner, and C. Platzer, "Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis," in *2015 IEEE 39th annual computer software and applications conference*, vol. 2. IEEE, 2015, pp. 422–433.

[20] A. Roy, D. S. Jas, G. Jaggi, and K. Sharma, "Android malware detection based on vulnerable feature aggregation," *Procedia Computer Science*, vol. 173, pp. 345–353, 2020.

[21] H. H. R. Manzil and S. Manohar Naik, "Android malware category detection using a novel feature vector-based machine learning model," *Cybersecurity*, vol. 6, no. 1, p. 6, 2023.

[22] A. Martín, V. Rodríguez-Fernández, and D. Camacho, "CANDYMAN: Classifying Android malware families by modelling dynamic traces with Markov chains," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 121–133, 2018.

[23] A. T. Kabakus and I. A. Dogru, "An in-depth analysis of Android malware using hybrid techniques," *Digital Investigation*, vol. 24, pp. 25–33, 2018.

[24] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2020.

[25] "Android Permissions Dataset." [Online]. Available: https://www.kaggle.com/datasets/gauthamp10/app-permissions-android

[26] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proceedings of the eighth symposium on usable privacy and security*, 2012, pp. 1–14.

[27] "Real-time query engine | Deephaven." [Online]. Available: https://deephaven.io/

[28] L. Vendramin, R. J. Campello, and E. R. Hruschka, "Relative clustering validity criteria: A comparative overview," *Statistical analysis and data mining: the ASA data science journal*, vol. 3, no. 4, pp. 209–235, 2010.

[29] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 2014, pp. 839–847.

DECLARATION

I hereby certify that the work which is presented in Major Project-II entitled App Permission Classification: Static And Dynamic Methods in fulfilment of the requirement for the award of the Degree of Master of Technology in Data Science and submitted to the Department of Software Engineering, Delhi Technological University, Delhi is an authentic record of my own, carried out during a period from January to June 2023, under the supervision of Dr. Divyashikha Sethia.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/ communicated in SCI/SCI expanded/SSCI/Scopus indexed journal OR peer-reviewed Scopus indexed conference with the following details:

Title of the Paper: APEC: App Permission classification with Efficient Clustering

Author names (in sequence as per research paper): Praveen Singh Rawal, Divyashikha Sethia

Name of Conference/Journal: Computational Intelligence for Information, Security and Communication Applications (CIISCA)

Conference Dates with venue (if applicable): 22nd & 23rd JUNE 2023 In Bengaluru, India

Have you registered for the conference (Yes/No): Yes

Status of paper (Accepted/Published/Communicated): Accepted

Date of paper communication: May 15, 2023

Date of paper acceptance: May 25, 2023

Date of paper publication: N/A

Title of the Paper: App Permission Classification dynamic Model (APCM)

Author names (in sequence as per research paper): Praveen Singh Rawal, Divyashikha Sethia

Name of Conference/Journal: The 14th International Conference on Computing, Communication and Networking Technologies (ICCCNT)

Conference Dates with venue (if applicable): July 6th - 8th, 2023 In New Delhi, India

Have you registered for the conference (Yes/No): No

Status of paper (Accepted/Published/Communicated): Communicated

Date of paper communication: April 30, 2023

Date of paper acceptance: N/A

Date of paper publication: N/A

Praveen Singh Rawal

2K21/DSC/08

SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I, further certify that the publication and indexing information given by the students is correct.

Divya Sethia

Place: Delhi

Date: 31May,2023

SUPERVISOR

Dr. Divyashikha Sethia

Assistant Professor

Department of Software Engineering

DTU
Delhi Technological UNIVERSITY

praveensinghrawal 2k21dsc08 <praveensinghrawal_2k21dsc08@dtu.ac.in>

CIISCA 2023 notification for paper 654

1 message

CIISCA 2023 <ciisca2023@easychair.org> Thu, May 25, 2023 at 7:17 PM
To: Praveen Singh Rawal <praveensinghrawal_2k21dsc08@dtu.ac.in>

Dear Praveen Singh Rawal,

Congratulations!

On behalf of the CIISCA-2023 Technical Program Committee, we are pleased to inform you that your submitted paper ID: 654, entitled "APEC: App Permission classification with Efficient Clustering" has been accepted for presentation in CIISCA-2023 to be held in Global Academy of Technology, Bangalore, Karnataka, India, during June 22– 23, 2023 online and in-person mode.
Paper ID: PID-654
You are requested to use Paper ID for further communication regarding this paper.
Thanks for your support to CIISCA-2023.
Please note that the accepted papers will be indexed in IEEE Xplore database if
• The final/Camera Ready paper uses the proper format according to the template provided by IEEE.
• The final/Camera Ready paper passes through IEEE compliance in terms of PDF Express.
• The final/Camera Ready paper passes IEEE Plagiarism test (<20%).
• The final/Camera Ready paper (both word and pdf format) must be mailed on or before 27th May 2023 to ciisca@gat.ac.in. Please write the Title of the mail as per the following format: PID-654_Camera ready paper
• The IEEE electronic Copyright Form is filled and ECF link will be shared with separate mail after completing registration and camera-ready paper is received.
• One of the Author must register and present the paper in the conference.
• The reviewer's comments (if any, in the bottom of this email) must be addressed properly in the final/Camera Ready of the paper.
Note:
• The registration link: https://forms.gle/GQASDNR3qwat2iiDA
• Payment details:
Account Holder Name: Industry Institute Interaction Cell- GAT
Account Number: 143510100047637
Bank and Branch: Union Bank of India, RR Nagar Branch
IFSC Code: UBIN0814351
• Complete the registration by 27th May 2023.
• If you want plagiarism report then write to us to ciisca@gat.ac.in Please write the Title of the mail as per the following format: PID-654_plagiarism report required
We also like to extend our sincere thanks to the reviewers for handling review of this paper. Thanks for submitting your paper to CIISCA-2023. We look forward to see you during the Conference.
Best regards,
Dr. Galiveeti Hemakumar Reddy
Technical Program Committee Chair(s), CIISCA-2023

SUBMISSION: 654
TITLE: APEC: App Permission classification with Efficient Clustering

----------------------- REVIEW 1 ---------------------
SUBMISSION: 654
TITLE: APEC: App Permission classification with Efficient Clustering
AUTHORS: Praveen Singh Rawal and Divyashikha Sethia

----------- Overall evaluation -----------
SCORE: 1 (weak accept)
----- TEXT:
- Avoid paragraphs in the abstract.
- Improve the quality of the figures as text in the figures is not visible.
- the pages to 6 pages.
- Avoid using citation in the conclusion
----------- Recommendation for best paper awards -----------
SELECTION: no

GLOBAL ACADEMY OF TECHNOLOGY, BENGALURU

(An Autonomous Institute, Affiliated to VTU, Belagavi, Recognized by Karnataka, Approved by AICTE, New Delhi and NBA Accredited CSE, ECE, EEE, ISE, and ME programs)

First International Conference on

"Computational Intelligence for Information, Security and Communication Applications"

Organized by:

Department of Artificial Intelligence and Data Science

IMPORTANT DATES

15th May, 2023	31st May, 2023	5th June, 2023	6th June, 2023
Last date for paper submission	Notification of Acceptance	Camera Ready Paper Submission	Last date for registration

HELD ON **22nd-23rd June 2023**

Published by

IEEE COMPUTER SOCIETY CONFERENCE PUBLISHING SERVICES

Endorsed By

CIISCA ENABLE-EMPOWER-INSPIRE

Academic Partner (In Progress)

Indian Institute of Information Technology, Allahabad

TRACKS

- Electronics Devices, Circuits and Systems
- Network Technologies and Systems
- Signal Processing
- Data Mining and Big Data Analysis
- Communication Technologies and Systems
- Bioinformatics and Machine Learning Algorithms
- Multimedia Processing Technologies and Systems
- Language Technologies and Information Retrieval
- Robotics and Artificial Intelligence
- Power Electronics and Control Engineering
- Security, Privacy and Digital Forensics
- Internet of Things

REGISTRATION FEE

UG/ PG Students/ Ph.D Scholars:	₹ 9000/-
Academicians	₹ 9500/-
Industry Delegates	₹ 10000/-
International Delegates	$ 200

CONTACT

Dr. Shoaib Kamal, 9986912260, shoaib.kamal@gat.ac.in

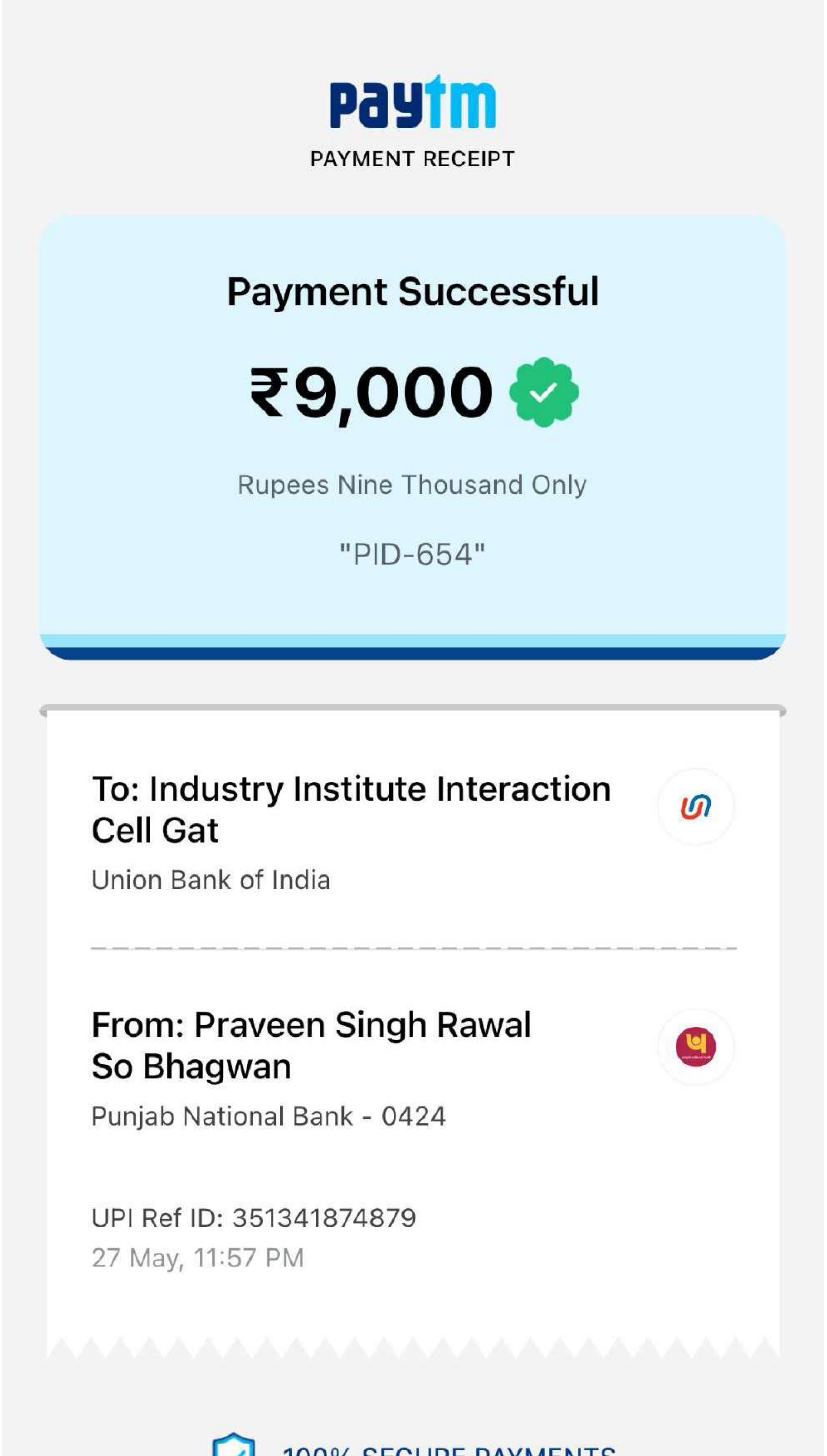
Dr. Ashwini Kodipalli, 9663332643, dr.ashwini.k@gat.ac.in

Dr. G. Hemakumar Reddy, 9704638836, ghkr220@gmail.com

NOTE : ALL PRESENTED PAPERS WILL BE SUBMITTED TO IEEE XPLORE DIGITAL LIBRARY AND SUBMITTED FOR INDEXING TO SCOPUS

Submission Link:
https://easychair.org/conferences/?conf=ciisca2023

For Registration:
https://gat.ac.in/ciisca2023/



Paytm

PAYMENT RECEIPT

Payment Successful

₹9,000 ✓

Rupees Nine Thousand Only

"PID-654"

To: Industry Institute Interaction Cell Gat

Union Bank of India

From: Praveen Singh Rawal So Bhagwan

Punjab National Bank - 0424

UPI Ref ID: 351341874879

27 May, 11:57 PM

100% SECURE PAYMENTS

praveensinghrawal 2k21dsc08 <praveensinghrawal_2k21dsc08@dtu.ac.in>

14th ICCCNT 2023 submission 1253

1 message

14th ICCCNT 2023 <14thicccnt2023@easychair.org> Sun, Apr 30, 2023 at 11:59 PM
To: Praveen Singh Rawal <praveensinghrawal_2k21dsc08@dtu.ac.in>

Dear authors,

We received your submission to 14th ICCCNT 2023 (14th International Conference on Computing Communication and Networking Technologies -2023):

Authors : Praveen Singh Rawal and Divyashikha Sethia
Title : App Permission Classification dynamic Model(APCM)
Number : 1253

The submission was uploaded by Praveen Singh Rawal <praveensinghrawal_2k21dsc08@dtu.ac.in>. You can access it via the 14th ICCCNT 2023 EasyChair Web page

https://easychair.org/conferences/?conf=14thicccnt2023

Thank you for submitting to 14th ICCCNT 2023.

Best regards,
EasyChair for 14th ICCCNT 2023.

14th ICCCNT 2023

THE 14th INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT)

July 6th - 8th, 2023
IIT - Delhi, Delhi India.

Hybird Conference - Online / Onsite
Presentations Accepted

Welcome to 14th ICCCNT

The 14th International Conference on Computing, Communication and Networking Technologies (ICCCNT) is a premier conference which is being organized during July 6th - 8th, 2023 at IIT - Delhi, Delhi, India. The computing, communication and networking are the most compelling areas of research because of its rich applications. So far ICCCNT form successfully completed 13 version of conferences in different locations across globe. It continuously receives research papers from different countries and different level of colleges / Universities. All the conferences papers were successfully published in IEEE Digital Library Xplore® and indexed in Scopus. It is a prestigious event organized every year with a motivation to provide an excellent platform for the leading academicians, researchers, industrial participants and budding students to share their research findings with the renowned experts.

The accepted papers with extended version will be considered for publication

NEWS AND UPDATES

Paper submission deadline is extended

Submit your paper: Submission link (https://easychair.org/my/conference?conf=14thicccnt2023)

Author Registration Link is open: Register (14icccnt.com/register/)

in various SCI/ SCIE/Scopus Journals.

(important-dates.php)
(important-dates.php)

Call for Papers

(call-for-papers.php)

Important Dates

(important-dates.php)

Venue

(venue.php)

(venue.php)

(venue.php)
(venue.php)
(venue.php) (venue.php)HOME (https://www.14icccnt.com/) | SITEMAP (http://www.ieee.org/sitemap.html) | ACCESSIBILITY (https://www.ieee.org/accessibility-statement.html) |

TERMS (https://www.ieee.org/about/help/site-terms-conditions.html) | IEEE PRIVACY POLICY (https://www.ieee.org/security_privacy.html) |

Hit Counter (https://visitorshitcounter.com/)