InPermHardroid: Android Malware Detection using Intents, Permissions and Hardware Components

A DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF SCIENCE(M.Sc.)

IN

MATHEMATICS

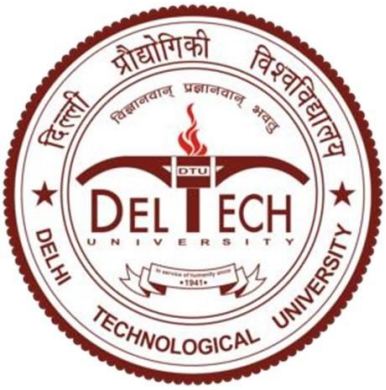
Submitted By:

HIMANSHI

(23/MSCMAT/80)

Under the supervision of

PROF. ANSHUL ARORA



DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

MAY, 2023

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Himanshi, Roll No. 23/MSCMAT/80 student of Master in Science (Mathematics), declaring that the project's dissertation titled InPermHardroid: Android Malware Detection using Intents, Permissions and Hardware Components is original and not copied from any source without proper citation and is presented by me to the Department of Applied Mathematics, Delhi Technological University, Delhi, in partial fulfilment of the requirement for the award of the degree of Master of Science in Mathematics, 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.

Place: Delhi
Date: May 26, 2025

Himanshi
23/MSCMAT/80

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby attest that the project dissertation InPermHardroid: Android Malware Detection using Intents, Permissions and Hardware Components submitted by Himanshi, Roll No. 23/MSCMAT/80 and of Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Masters of Science in Mathematics, is a record of the project work carried out 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: May 26, 2025

Prof. Anshul Arora
Supervisor

ACKNOWLEDGEMENT

My supervisor, Prof. Anshul Arora of the Department of Applied Mathematics at Delhi Technological University, has my sincere gratitude for his meticulous guidance, profound expertise, constructive criticism, attentive listening, and amiable demeanor have been invaluable throughout the process of composing this report. I am eternally grateful for his benevolent and supportive approach, as well as his perceptive counsel, which played a pivotal role in the successful culmination of my project. Furthermore, I would like to express my appreciation to all my classmates who have played a pivotal role in aiding me to complete this endeavor by offering assistance and facilitating the exchange of pertinent information.

Himanshi
23/MSCMAT/80

ABSTRACT

As mobile devices continue to evolve and become integral to daily life, the risks posed by malicious software are increasing. Cybercriminals continuously develop more sophisticated malware, bypassing conventional security mechanisms and compromising user privacy. To counter these threats, it is crucial to establish a highly efficient and precise malware detection framework. This study introduces a creative approach for malware identification and classification. The framework employs Fisher's P-test, a statistical method that isolates the most distinguishing features between benign and malicious applications, eliminating irrelevant attributes early in the process. This initial refinement improves efficiency for subsequent classification. Following this, the Relief algorithm further enhances the data set by identifying the most relevant attributes that contribute significantly to malware detection. The integration of these two techniques results in an optimized dataset, reducing complexity while improving classification accuracy.

Once the features are refined, machine learning models analyze the data to detect and categorize malware. Among the classifiers evaluated, Random Forest proves to be the most effective, utilizing an ensemble of decision trees to enhance predictive accuracy while minimizing overfitting. When applied to a dataset combining permissions, intents, and hardware components, the Random Forest classifier achieves an impressive precision of approximately 97%, demonstrating the effectiveness of a multi-dimensional approach in malware detection.

This research makes substantial contributions to the field of mobile security by designing a more precise and interpretable malware detection system. As threats targeting mobile platforms continue to grow, integrating advanced security techniques is essential for protecting user data and privacy. This study provides valuable insights that pave the way for advancing machine learning-driven cybersecurity strategies, strengthening mobile ecosystems against emerging cyber threats.

Index Terms—Mobile Malware, Android Security, Fisher's P test, Relief Algorithm, Feature Selection

.

Contents

Candidate's Declaration **ii**

Certificate **iii**

Acknowledgement **iv**

Abstract **v**

1. Introduction

1.1 Android OS……1

1.2 Security issues in Android OS……..2

1.3 Detection Analysis……4

1.4 Motivation……5

1.5 Research Contributions……6

1.6 Thesis Framework……7

2. Prerequisites of Permission, Intent and Hardware

2.1 Overview of Permissions……8

2.2 Overview of Intent……8

2.3 Overview of Hardware……..8

3. Related Work

3.1 Literature Analysis……...9

3.2 Merits and Demerits of Existing work……11

4. Methodology

4.1 Data Collection……13

4.2 Data Preprocessing……..13

4.3 Detection with Machine Learning techniques…….17

5. Result

5.1 Permission Evaluation…….20

5.2 Intent Evaluation…….20

5.3 Hardware Evaluation……..21

5.4 Permission and Intent Evaluation……...22

5.5 Permission and Hardware Evaluation……22

5.6 Intent and Hardware Evaluation……22

5.7 Permission, Intent and Hardware Evaluation……23

6. Conclusion

6.1 Thesis Summary……..24

6.2 Future Work…….25

References

List of Tables

4.2 Fishers p test……..15

4.2 Feature Reduction…….16

5.1 Detection Results for permission evaluation…….20

5.2 Detection Results for intent evaluation…….21

5.3 Detection Results for hardware evaluation……...21

5.4 Detection Results for permission and intent evaluation……22

5.5 Detection Results for permission and hardware evaluation…….....22

5.6 Detection Results for intent and hardware evaluation……..23

5.7 Detection Results for permission, intent and hardware evaluation……...23

Chapter 1: Introduction

1.1 Android OS

Android OS has become a go-to operating system, with a focus on touchscreen devices. Built on the Linux kernel, it operates within an open-source framework, giving device makers and developers around the globe the freedom to tweak and enhance the system to fit various needs and specifications. This flexibility, paired with its robust features, has made Android the top mobile operating system worldwide.

One of the standout features of Android is its user-friendly interface, designed for easy navigation. It boasts customizable home screens, interactive widgets, handy notifications, and smooth gesture controls, all of which create a delightful user experience. Plus, Android supports multitasking, letting users switch between apps with ease, and its navigation system keeps getting better with each update, enhancing accessibility and usability.

The success of Android can be largely credited to its vast application environment. The Google Play Store is home to millions of apps spanning a wide array of categories, from games and productivity tools to social networking and educational resources. Developers mainly use Kotlin and Java, along with a comprehensive Software Development Kit (SDK) and powerful development tools, to craft a diverse range of high-quality applications that elevate the user experience.

Additionally, Android offers significant customization options for both manufacturers and users. People can personalize their devices with various custom launchers, themes, wallpapers, and icon packs. This level of personalization allows users to tailor their devices to their individual tastes and needs, making Android especially versatile compared to other operating systems.

1.2 Security Issues in Android OS

Android OS is one of the popular mobile operating systems out there, but it certainly has its fair share of security hurdles. These issues arise from its open-source design, the wide variety of devices it operates on, and the enormous range of apps available. Because of this, Android devices often find themselves in the crosshairs of different types of malware and cyber threats, which can jeopardize user data and the overall integrity of the device.

- Malware Threats: Android devices are especially susceptible to various types of malware, which can have serious consequences for users. Here are a few of the most prevalent forms of malware:
    - SMS Trojans: Unidentified apps may secretly send messages or make calls without the user's awareness, potentially compromising security. This not only increases costs for users but also poses risks of privacy breaches and data theft.
    - Advertising Modules: Some apps come packed with advertising modules that bombard users with intrusive ads. These ads can redirect users to harmful websites or encourage them to download more malicious software, putting device security at risk.
- Exploits for Root Access: Another major security issue is the risk of attackers exploiting vulnerabilities in the Android OS to gain root access. This level of access allows malicious individuals to tamper with system files and settings, leading to serious repercussions, such as:
    - Data Theft: Once an attacker gains root access, they can dig into sensitive information stored on the device, including personal messages, contacts, and financial details.
    - Device Compromise: With root access, attackers can install additional malware, alter system settings, or even make the device unusable.
- Outdated Software: A significant number of Android devices don’t receive timely security updates, leaving them exposed to known vulnerabilities. Many manufacturers and carriers often overlook updates for older devices, which can result in:

    - Increased Vulnerability: Devices that are running outdated software are more susceptible to attacks, as hackers can exploit known vulnerabilities.
    - User Responsibility: It is important for users to regularly check for updates and install them without delay to reduce these risks. Unfortunately, many people often overlook this crucial part of keeping their devices in good shape.
- Risks from Third-Party Apps: One of the great things about Android is the ability to install apps from third-party sources, but this also brings significant security concerns. Users who download apps from untrustworthy sources might unintentionally introduce malware to their devices. To help protect against harmful software, users should:
    - Stick to Reputable Sources: By downloading apps only from the Google Play Store or other trusted app stores, users can greatly lower the chances of malware infections.
    - Read Reviews and Ratings: Before hitting that install button, it's wise to check out reviews and ratings to assess the app's reliability and safety.
- Insecure Network Connections: Android devices frequently connect to unsecured Wi-Fi networks, which can make them exposed to various attacks, including man in the middle attacks. These attacks can intercept data being transmitted over the network, leading to:
    - Data Breaches: Sensitive information like login credentials and personal data can be captured by attackers.
    - Using a VPN: A Virtual Private Network (VPN) can help secure data transmission over public networks, adding an extra layer of protection against potential threats.
- Permissions Mismanagement: Many users tend to give too many permissions to apps during installation, which can lead to unauthorized access to sensitive information. This mismanagement can result in:
    - Privacy Violations: Apps might access personal information such as contacts, messages, and location data without the user's clear consent.
    - Reviewing Permissions: It is really important for users to take the time to regularly check app permissions and only allow access to what’s absolutely necessary for the app to work. This simple habit can go a long way in keeping your personal data safe from unauthorized access.

1.3 Detection Analysis

As Android OS continues to lead the charge in the mobile operating system arena, the importance of robust security measures is more crucial than ever. A vital aspect of Android security is detection analysis, which focuses on spotting and addressing potential threats to safeguard user data. This analysis employs a variety of techniques and strategies aimed at uncovering malware, vulnerabilities, and unauthorized access. The detection methods in Android security can be grouped into several categories:

- Signature-Based Detection: This classic approach relies on known malware patterns or signatures. Security software scans apps and files for these signatures to pinpoint malicious content. While it works well against familiar threats, it often falls short when faced with new or altered malware that doesn't match existing signatures.
- Heuristic-Based Detection: Heuristic analysis takes things a step further by looking at how applications behave. This method assesses the actions of an app to see if they raise any red flags or suggest malicious intent. For instance, an app that asks for too many permissions or tries to access sensitive data without a good reason might be marked as potentially harmful.
- Behavioral Detection: Much like heuristic detection, behavioral detection keeps an eye on how applications behave while they're running. It examines how an app interacts with the system and other apps, searching for unusual activities that could indicate malicious behavior.
- Anomaly Detection: This technique sets a standard for what normal behavior looks like for applications and the operating system. Any significant deviation from this norm is flagged for further scrutiny. Anomaly detection is particularly useful for spotting odd patterns that might signal a security breach or malware infection.
- Machine Learning and AI: The way machine learning and artificial intelligence have been woven into detection systems has truly changed the game for Android security. These smart technologies can shift through massive amounts of data to spot patterns and foresee potential threats. Plus, machine learning models are designed to adapt and get better over time, which boosts their capability to catch new and evolving threats.

1.4 Motivation

The speedy rise of mobile technology, especially the Android operating system, has completely changed the way we interact with digital services. With Android holding about 85% of the global mobile OS market share, it is become an essential tool for everything from communication to banking and social networking. However, this widespread use has also caught the eye of cybercriminals who take advantage of weaknesses in the platform to spread various types of malware like spyware, trojans, and ransomware. These harmful apps can seriously threaten user privacy and security, leading to financial losses and data breaches. As mobile apps increasingly manage sensitive transactions and personal data, the demand for effective malware detection solutions has never been more urgent.

In light of this, the research being proposed aims to create a strong and understandable malware detection framework that merges advanced feature selection techniques with machine learning algorithms. By using Fisher's P-test and the Relief algorithm, the study intends to pinpoint statistically significant and relevant attributes from Android applications. This combined feature selection method not only boosts the accuracy of malware detection systems but also cuts down on computational demands, making the detection process more efficient. Being able to refine the feature set is vital for enhancing model performance and ensuring that possible threats are recognized promptly.

The results of this research are set to make a meaningful impact on the world of Android security by shedding light on effective strategies for detecting malware. Early findings suggest that the proposed model, especially when using the Random Forest classifier, can reach impressive accuracy levels, showcasing its promise for real-time identification of harmful applications. Additionally, this work lays the groundwork for future studies that will delve into more static features and sophisticated detection techniques. By tackling the challenges posed by new threats, this thesis aspires to foster a safer mobile environment for users everywhere, ultimately boosting the overall security of mobile applications.

1.5 Research Contributions

The study presents an innovative feature selection method that capably merges Fisher's P-Test with the Relief Test. This combined approach is tailored to pinpoint the best data set for detecting malware. By harnessing the strengths of both statistical techniques, the research achieves several key results:

- Improved Detection Accuracy: The fusion of Fisher's P-Test, which evaluates the significance of features based on their ability to differentiate between classes, with the Relief Test, which assesses the relevance of features based on their impact on the classification task, leads to a more precise identification of harmful applications. This dual strategy ensures that only the most relevant features are kept, thus boosting the overall accuracy of the detection system.
- Decreased Computational Load: By streamlining the feature set, the proposed method not only enhances detection accuracy but also significantly cuts down on the computational resources needed for processing. This is especially crucial in real-time malware detection scenarios, where speed and efficiency are vital. Therefore, this hybrid feature selection approach offers a solid framework for enhancing the efficiency of malware detection systems without sacrificing performance.
- Extensive Dataset Utilization: This research taps into a rich dataset that highlights the static features of Android apps, such as permissions, hardware and intents. This thorough dataset is vital for a few key reasons:
    - Diverse Feature Representation: By including a broad array of features, the study guarantees that the model is trained on a well-rounded sample of Android applications. This variety is crucial for building a strong malware detection system that can adapt well to new, unseen apps.
    - Successful Feature Reduction: Using techniques like the Relief Test and Fisher's Exact Test, the original set of features is streamlined effectively. This trimming down results in a more efficient and lightweight malware detection system, which has been validated on a large dataset of over 112,000 Android applications. The substantial sample size boosts the reliability and relevance of the findings, making sure that the proposed feature selection method is not just theoretically sound but also practical in real-world scenarios.

1.6 Thesis Framework

The layout of this thesis report is organized into six distinct chapters:

Chapter 2 provides a comprehensive analysis of the key components related to permissions, intents, and hardware in Android.

Chapter 3 presents the literature work that has been done in the field of Android malware detection, focusing on the use of intent, hardware and permissions components as well as their combinations to identify malicious applications.

Chapter 4 presents the proposed approach and methodology for the work.

Chapter 5 presents the detailed results of the evaluations conducted in the Android malware detection framework.

Chapter 6 provides a comprehensive summary of the thesis and discusses potential directions for future research.

References

Chapter 2: Prerequisites about Permissions, Intents and Hardware Components

To really get a solid grasp of the upcoming material, it is important to first understand the key foundational concepts. Building this knowledge base will set you up with the right framework to effectively tackle the discussions that follow.

2.1 Overview of Permissions

In the Android operating system, permissions are a vital security feature that controls how apps access sensitive data and system resources. Essentially, permissions are declarations made by developers in the app's manifest file, outlining what capabilities the app needs to operate correctly. This system is designed to safeguard user privacy, ensuring that apps cannot access sensitive information without the user’s explicit consent. As mobile applications increasingly deal with personal data like contacts, location, and financial details, understanding these permission components is crucial for both developers and users.

2.2 Overview of Intents

In the Android operating system, intents are a key element that enables communiqué between dissimilar parts of an application and even between dissimilar applications. An intent acts as a messaging object that allows components to request actions from one another, such as starting an activity, sending a broadcast, or initiating a service. Intents are essential to the Android architecture, allowing developers to create active and interactive applications that can respond to user actions and system events.

2.3 Overview of Hardware

In the world of Android, hardware components are the physical devices and sensors that come together in mobile gadgets like smartphones and tablets. These components are essential for boosting the functionality of apps by tapping into various device capabilities. Think of hardware components like cameras, microphones, GPS sensors, accelerometers, and gyroscopes. By making the most of these features, developers can craft engaging and interactive applications that truly enhance the user experience.

Chapter 3: Related work

3.1 Literature Analysis

In this section, we take a closer look at the various approaches that have been suggested in the literature for detecting Android malware. A few studies, like those referenced in [12] and [13], have introduced techniques that utilize feature selection methods followed by different base classifiers to boost detection rates. Arora et al. [14] and [15] put forward two models aimed at identifying malicious apps based on network traffic analysis. Meanwhile, the authors of [1], [16], and [20] focused on analyzing permissions and API calls as a means of detecting malware. Additionally, the team behind [2] developed a malware detection system called Significant Permission Identification (SigPID), which employs a three-level pruning process to mine permission data and pinpoint the most crucial permissions that can effectively differentiate between benign and malicious applications. Akbar [3] assessed an app's maliciousness by examining the use of suspicious permissions and key features to achieve high accuracy.

Zhao Xiaoyan [4] utilized PCA and SVM to sort the gathered data into benign or malicious categories. They also suggested that focusing solely on permission features, combined with machine learning techniques, can lead to impressive detection outcomes. The authors in [5] pointed out that intent serves as a semantically rich feature, effectively capturing the intentions behind malware. M. W. Afridi [6] introduced the idea of using the intent generated by applications as a way to pinpoint malicious behavior. In [22], the authors examined 17 different supervised learning methods to conduct a comparative analysis of machine classifiers. M. Kumaran and W. Li [7] demonstrated that a Cubic SVM classifier achieves high accuracy by integrating both permission and intent. The authors in [8] employed Information Gain to rank permission and intent, aiming to identify the optimal set for detecting malware with greater precision. The authors in [9] explored how combining permission and intent filters can help differentiate between malware and benign apps by analyzing their respective patterns. In [10], the authors introduced a system for classifying Android applications based on the permissions they utilize. Lastly, the authors in [11] highlighted the potential of hardware-based approaches,

using machine learning classifiers that leverage Hardware Performance Counters (HPC) for real-time malware detection..

Y. Lu [17] introduced an advanced malware detection technique that utilizes machine learning methods, enhancing Naïve Bayesian Classification with Chi-Square filtering. In [24], the authors implemented a KNN-based Relief algorithm for feature selection, while malware detection was carried out using their optimized SVM algorithm, achieving a True Positive Rate exceeding 0.7. Sharfah Ratibah Tuan Mat [18] developed an Android malware detection system that focuses on permission features through Bayesian classification. The authors in [19] provided a comprehensive overview of the dataset characteristics, along with the processes for feature extraction and selection. Shabtai [23] proposed a Host-based Intrusion Detection System that continuously monitors various features and events from mobile devices, applying Machine Learning techniques to classify malicious applications. Additionally, the authors in [21] utilized feature extraction (AppExtractor) and innovative selection algorithms (FrequenSel) to pinpoint distinguishing features for effective malware detection..

C. P. Chenet [25] explored the link between hardware events and malware applications, even broadening his analysis to include mixed hardware and software strategies for better collaboration and significant improvements in hardware monitoring units. The authors in [26] utilized the malware genome dataset and the Drebin project for static analysis, while they turned to the CICMalDroid dataset for dynamic analysis. Kimberly Tam [27] shared insights on malware statistics and evasion techniques. Meanwhile, the authors in [28] focused on examining the algorithms used in malware detection and conducted a comparative review of existing literature. Lastly, the authors in [29] employed Android permissions and intents as a feature set for malware detection, using Principal Component Analysis for feature selection.

3.2 Merits and Demerits of Existing work

- **Merits of Existing Works**

    - Diverse Methodologies: The existing literature showcases a rich variety of methodologies for detecting Android malware, from feature selection techniques to machine learning classifiers. This range gives researchers and practitioners the flexibility to pick the approaches that best suit their unique needs and situations.

    - Focus on Permissions and API Calls: Many studies highlight the importance of analyzing permissions and API calls, which serve as key indicators of malicious activity. Since permissions define what applications can do, they become a crucial feature for differentiating between harmless and harmful apps.

    - Advanced Techniques: The use of advanced methods like Principal Component Analysis (PCA) and Support Vector Machines (SVM) demonstrates how sophisticated statistical techniques can boost classification accuracy. These methods can effectively reduce dimensionality and enhance the performance of machine learning models.

    - Comprehensive Evaluation of Classifiers: A thorough comparison of various supervised learning techniques offers valuable insights into how effective different classifiers are. This analysis aids in pinpointing the most appropriate algorithms for specific detection tasks, steering future research and development in the right direction.

    - Integration of Intent Features: By delving into the intentions behind application behaviors, researchers can create more refined detection systems that take into account not just permissions but also the context in which they are applied.

    - Hardware-Based Approaches: The focus on hardware-based strategies underscores the potential of utilizing hardware performance counters and mixed hardware-software methods for real-time malware detection. This combination can lead to more resilient detection

systems that function efficiently.

- **Demerits of Existing Works**

    - Complexity and Resource Intensity: When it comes to complexity and resource demands, many current methodologies, especially those that depend on intricate algorithms and a wide range of features, can be quite resource-heavy and take a lot of time to implement.

    - Overfitting Risks: Using complex models can sometimes lead to a situation where the model excels with training data but struggles to perform well with new, unseen applications. This is particularly worrisome in malware detection, where accurately identifying new and previously unknown applications is essential.

    - Evasion Techniques: we cannot ignore the challenge posed by evolving evasion techniques in malware. Many existing detection systems might not be prepared to tackle these advanced evasion strategies, which could leave them vulnerable in the fight against malware.

Chapter 4: Methodology

In the following sections, we provide a comprehensive explanation of the proposed methodology, detailing each aspect in depth.

4.1 Data Collection

- Obtain APKs: To get started with APKs, you'll want to download APK files from AndroZoo, which is a great repository for Android applications.
- Organize Your APKs: First, create a specific directory to store all your APK files. You can use a programming language to read through all the files in this folder.
- Process Each APK: Next, set up a loop to go through each APK file. For every file, follow these steps:
    - Decompile the APK: Use Apktool to decompile the APK file. You can usually do this with a simple command line call from your script.
    - Find the AndroidManifest.xml: Once decompiled, look for the AndroidManifest.xml file in the output directory.
    - Analyze the Manifest: Open the AndroidManifest.xml file to pull out permissions, intents, and hardware requirements.
- Save the Information: Finally, store the extracted data in a structured format, like JSON or CSV, so you can analyze it later.

4.2 Data Preprocessing

- **Feature Reduction**: Feature reduction plays a energetic role in the data preprocessing stage of machine learning and data analysis. It’s all about picking out a relevant subset of data from the actual dataset, which can really boost model performance, help prevent overfitting, and cut down on computational costs. In this document, we’ll dive into the specific techniques we used for feature reduction, such as getting rid of duplicate features.
    - **Removal of Duplicate Features:** Duplicate features can pop up for a variety of reasons, like errors during data collection, combining different datasets, or just having some measurements that are unnecessarily repeated. These duplicates can really throw off the results, as they might distort the model's grasp of the true patterns

hidden in the data. We took a close look at the dataset to find features that had the same values across all instances. If we found that the instances or features were identical, we removed them to cut down on redundancy.

- **Statistical Hypothesis Test**: A statistical hypothesis test is a technique used in statistics to figure out if there's enough evidence in a data sample to back up a specific claim about a larger population. This method allows researchers to make informed decisions based on actual data instead of just assumptions.
    - **Statistical Ranking with Fisher's P-test**: Fisher's P-Test, commonly known as Fisher's Exact Test, is a statistical technique that helps determine the significance of the relationship between two categorical variables. This test delivers an exact p-value, which empowers researchers to make well-informed decisions regarding the connections between variables. If you're looking to apply Fisher's P-Test, here's a method to get started:
        - **Formulating Hypotheses:** Before conducting the test, we need to establish the null and alternative hypotheses:
            - **Null Hypothesis (H0)**: There is no interrelation between the two categorical variables (i.e., the proportions are equal).
            - **Alternative Hypothesis (H1)**: There is a significant interrelation between the two categorical variables (i.e., the proportions are not equal).
        - **Constructing a Contingency Table:** The first step in applying Fisher's P-Test is to create a contingency table that summarizes the frequency counts of the categories. For example, consider a study examining the effect of a new drug on recovery status (Recovered vs. Not Recovered) across two groups (Treatment vs. Control):

Table 4.2: Fisher’s P-test

	Recovered	**Not Recovered**	**Total**
Treatment	a	b	a+b
Control	c	d	c+d
Total	a+c	b+d	n

- **Calculating the Exact Probability:** Fisher's Exact Test calculates the probability of observing the data in the contingency table under the null hypothesis. The probability is given by the hypergeometric distribution:

$$P=(a+b)!(c+d)!(n-a-b)!a!b!c!d!n!$$

Where: n is the total number of observations and a, b, c, d are the counts in the contingency table.

- **Determining the P-Value:** The p-value is calculated by adding up the probabilities of all possible tables that are as extreme or even more extreme than the one we observed, based on the null hypothesis. If this p-value falls below the significance level we've set 0 we reject the null hypothesis. This suggests that there’s a significant relationship between the variables.

- **Interpretation of Results:** The interpretation of Fisher's P-Test results is straightforward:

  - **P-Value <= 0**: Reject the null hypothesis. There is a statistically significant interrelation between the two categorical variables.

  - **P-Value > 0**: Fail to reject the null hypothesis. There is no statistically significant interrelation between the two categorical variables.

Features that exhibit a strong association with the target variable remain in the dataset, while those with weaker associations are removed. The remaining features move forward for the next selection process.

- **Feature selection:** In machine learning, there's a process that helps pick out the most important features to boost how well a model performs. By getting rid of irrelevant or repetitive data, it leads to improved accuracy and efficiency. Plus, it makes things easier to understand by zeroing in on the key variables that really matter. This process is essential for managing high-dimensional data and avoiding overfitting.
    - **Relief Test**: The Relief method is a feature selection approach that evaluates the relevance of each feature by estimating how well it can distinguish between similar instances. It assigns a weight to each feature based on its ability to differentiate between near-by instances, with higher weights indicating more relevant features.
        - **Random Sample Selection:** A random instance is selected from the training data.
        - **Nearest Neighbours Identification:** The algorithm identifies two nearest neighbours for the selected instance: a "nearest hit" (belonging to the same class as the instance) and a "nearest miss" (belonging to a different class).
        - **Feature Weight Update:** For each feature, the algorithm calculates a difference based on the values of the feature in the selected instance, the nearest hit, and the nearest miss. If the feature values are more different between the nearest hit and the instance, the feature is considered less important. Conversely, if the feature values are more different between the instance and the nearest miss, the feature is considered more important. This difference is then used to update the feature weight.
        - **Iteration:** Steps 1-3 are repeated for multiple instances to converge on a more stable feature weight for each feature.
    - **Feature Ranking and Selection:** After going through the data multiple times, the algorithm sorts the features according to their final weights. Features that end up with higher weights are seen as more important and have a better chance of being chosen. Essentially, Relief

operates by repeatedly assessing how effectively each feature can tell apart similar instances, giving weights based on this ability to differentiate, and ultimately using these weights to rank and pick out the most significant features.

Table 4.2: Feature Reduction

FEATURE SET	Initial Count of Features	Removing Duplicate Features	Removing Features through Statistical Hypothesis test	Removing features through Relief Test
PERMISSION	130	129	75	40
INTENT	80	80	45	35
HARDWARE	89	83	13	10
PERMISSION AND INTENT	209	208	122	40
PERMISSION AND HARDWARE	218	211	88	50
INTENT AND HARDWARE	168	162	59	40
PERMISSION AND INTENT AND HARDWARE	297	290	135	50

4.3 Detection with Machine Learning techniques

We utilize machine learning for detection purposes. We introduce a column labeled "apk," where a value of 0 signifies benign applications and 1 indicates malicious ones. After that, we divide the dataset into training and testing sets, and then we apply various machine learning models like Decision Tree, Random Forest, and Naive Bayes for detection. Our goal is to explore different algorithms to identify the most effective technique for comparison. Here are the machine learning methods

we've employed:

- Decision Tree: This is a machine learning algorithm used for both classification and regression tasks. Each internal node represents a feature of the data, while the branches illustrate decision rules based on those features, and the leaf nodes show the final outcome or class label. The algorithm constructs the tree by recursively splitting the dataset into smaller subsets based on the best feature, which is determined using criteria like Gini impurity or entropy, until a stopping condition is reached.

- Random Forest: Random Forest is a powerful ensemble learning technique that's mainly used for tasks like classification and regression. It works by building a bunch of decision trees during the training stage and then combines their outputs—either by taking the most common class (for classification) or averaging the predictions (for regression). The main goal of Random Forest is to boost the accuracy and reliability of predictions by merging the results from several trees, which helps reduce the chances of overfitting that can happen with just one decision tree. The process kicks off with bootstrapping, where various subsets of the training data are created by sampling with replacement. Each tree is skilled on a different subset, and while training each tree, a random selection of features is chosen for splitting at each node. This element of randomness is key to ensuring that the trees are diverse, which is essential for the effectiveness of the ensemble method. After all the trees are trained, predictions are made by combining the results from each tree—using majority voting for classification tasks or averaging for regression tasks.

- Naïve Bayes: Naive Bayes is a group of probabilistic algorithms that rely on Bayes' theorem, mainly used for classification tasks. It works on the premise that the features are conditionally independent when given the class label. This is a simplifying assumption that often holds up in real-world scenarios, even if it’s not always spot-on. Thanks to this independence assumption, Naive Bayes can quickly calculate the posterior probabilities for each class based on the input features. The algorithm first determines the prior probability for each class and then the conditional probability for each feature based on that class. By applying Bayes' theorem, it merges these probabilities to find the later probability for each class. The class with the highest posterior

probability is then chosen as the predicted class for the input data.

Chapter 5: Results

This section presents the results from our proposed Android malware detection framework, which cleverly combines machine learning techniques with sophisticated feature selection strategies. We have organized the findings into clear categories based on an evaluation of essential components: permissions, intents, hardware features, and their various combinations. Moreover, we assess the effectiveness of different machine learning classifiers—like Decision Trees, Random Forest, and Naive Bayes to gauge how well the framework performs.

5.1 Permission Evaluation

- The original dataset kicked off with 130 permission features pulled from a vast array of Android apps. After running Fisher's P-test, we trimmed that down to 75 features that really stood out statistically. Then using this selection approach i.e. Relief algorithm, landing on a final set of 40 key permission features.
- When it came to detecting permission-based malware, the Random Forest classifier really shown, hitting an impressive accuracy of 96.96% post-feature selection. In comparison, the Decision Tree classifier dipped slightly from 96.53% to 96.31%, while the Naive Bayes classifier made a significant leap from 87.35% to 90.80%.

Table 5.1: Detection Results for permission evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	96.53%	96.43%	96.31%
Random Forest	97.32%	97.14%	96.96%
Naive Bayes	87.35%	88.74%	90.80%

5.2 Intent Evaluation

- The original dataset kicked off with 80 intent features, but after running Fisher's P-test, we trimmed it down to 45. Then, by applying the Relief algorithm, we refined it even further to 35 features that were truly relevant for our analysis.
- When we looked at the performance of the classifiers, the Decision Tree's accuracy slide from 88.58% to 87.16%. The Random Forest classifier also took a small hit,

dropping from 88.72% to 87.21%. Meanwhile, Naive Bayes saw a decrease from 76.27% to 75.26%. In the grand scheme of things, while intent features played a role in the detection model, but they did not match up to the effectiveness of permission-based features.

Table 5.2: Detection Results for intent evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	88.58%	87.42%	87.16%
Random Forest	88.72%	88.48%	87.21%
Naive Bayes	76.27%	75.49%	75.26%

5.3 Hardware Evaluation

- The original dataset kicked off with 89 features related to hardware components, but we trimmed that down to just 13 using Fisher's P-test. Then, we took it a step further with the Relief algorithm, which helped us in coming to 10 key features for our analysis.
- When looked at the Decision Tree classifier, it was noticed a slight dip in accuracy, dropping from 67.66% to 66.34%. The Random Forest classifier experienced a similar fall, going from 67.70% to 66.34%. On a brighter note, the Naive Bayes classifier actually improved, jumping from 59.71% to 66.08% after we selected the features. In the grand scheme of things, it turns out that hardware components didn't play a huge role in boosting detection accuracy when compared to permissions and intents.

Table 5.3: Detection Results for hardware evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	67.66%	66.51%	66.34%
Random Forest	67.70%	66.53%	66.34%
Naive Bayes	59.71%	65.82%	66.08%

5.4 Permission and Intent Evaluation

- The dataset included permissions and intents, ended up with 122 features after the feature selection process.
- The Random Forest classifier really shone, achieving an impressive accuracy of 96.96%. Meanwhile, Naive Bayes also made notable strides, boosting its accuracy to 90.80%.

Table 5.4: Detection Results for permission and intent evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	96.53%	96.43%	96.31%
Random Forest	97.32%	97.14%	96.96%
Naive Bayes	87.35%	88.74%	90.80%

5.5 Permission and Hardware Evaluation

- After applying Fisher's P-test and Relief, the permissions and hardware dataset ended up with 88 optimized features.
- The Naive Bayes classifier experienced the biggest accuracy jump, hitting 91.28%, while Random Forest held its ground with a solid performance at 97.12%.

Table 5.5: Detection Results for Permission and Hardware evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	96.67%	96.50%	96.41%
Random Forest	97.45%	97.28%	97.12%
Naive Bayes	87.12%	89.68%	91.28%

5.6 Intent and Hardware Evaluation

- The initial 59 features for intents and hardware were processed through Fisher's P-test and Relief for refinement.
- The Naive Bayes classifier showed improvement, but overall accuracy was lower compared to other combinations, indicating a limited impact of hardware features.

Table 5.6: Detection Results for Intent and Hardware evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	90.08%	88.53%	85.89%
Random Forest	90.39%	88.79%	85.97%
Naive Bayes	71.96%	76.43%	75.17%

5.7 Permission and Intent and Hardware Evaluation

- The dataset combined all three categories, resulting in a total of 135 optimized features.
- The Random Forest classifier attained the maximum accuracy at 97.30%, confirming that integrating multiple feature sets enhances malware detection.

Table 5.7: Detection Results for Permission and Intent and Hardware evaluation

Classifier	Initial Accuracy	After Fisher's P-test	After Relief
Decision Tree	96.92%	96.70%	96.57%
Random Forest	97.78%	97.54%	97.30%
Naive Bayes	90.22%	88.01%	88.64%

Chapter 6: Conclusion

6.1 Thesis Summary

In this research, we tackled the issue of detecting Android malware, a topic that's become increasingly vital with the surge in mobile device usage and the rise of malicious apps. Our focus was on creating a strong and efficient malware detection framework that harnesses machine learning techniques, specifically by using features drawn from permissions, intents, and the hardware components of Android applications.

We kicked off our work by carefully selecting and preprocessing the features. We started with a detailed dataset that encompassed various aspects related to permissions, intents, and hardware specifications. The first order of business was to remove out any duplicate features, which is crucial for keeping the dataset intact and ensuring our models are trained on unique and relevant data. This step not only tidied up the dataset but also set the stage for more precise model training and evaluation.

After clearing out the duplicates, we evaluated the performance of our detection models—Decision Tree, Random Forest, and Naive Bayes—using the initial set of features. This assessment gave us baseline accuracy scores, which were essential for gauging how effective our models were before we applied any feature selection techniques. The results showed that while the models did reasonably well, there was still plenty of room for improvement, especially regarding accuracy and computational efficiency..

To make our feature selection process even better, we turned to Fisher's p-test, a handy statistical tool that helped us assess how significant each feature was in relation to our target variable. By ranking the features according to their p-values, we could spot and eliminate those that didn't hold much statistical weight. This was a crucial step in fine-tuning our feature set, ensuring we kept only the most relevant features for further analysis. Cutting down the number of features not only made the model simpler but also enhanced its interpretability and lowered the chances of overfitting.

Next, we brought in the Relief algorithm to further sharpen our feature selection. By taking samples from the dataset and repeatedly assessing the importance of different features, we crafted a more targeted feature set. The Relief algorithm's knack for

recognizing feature interactions and dependencies gave us a deeper insight into which features were key in telling benign applications apart from malicious ones. This iterative approach, where we tracked feature importance over several rounds, allowed us to systematically remove out the less important features while keeping those that consistently showed high relevance.

After fine-tuning our feature set using both Fisher's p-test and the Relief algorithm, we took a fresh look at how accurate our detection models were. The results were quite encouraging, with the Random Forest model hitting an impressive accuracy rate of around 97%. This notable leap in performance really highlighted how effective our feature selection methods are and emphasized the value of a data-driven strategy to boost model performance. Our study's findings not only add to the existing knowledge in the realm of Android malware detection but also offer practical insights for crafting more robust security solutions.

In summary, our research shows that an approach to feature selection, leveraging statistical techniques like Fisher's p-test and the Relief algorithm, can greatly improve the performance of machine learning models in Android malware detection. By honing in on the most relevant features, we managed to enhance both the accuracy and efficiency of our detection models, ultimately paving the way for a more secure mobile application environment. As mobile devices become increasingly central to our everyday lives, the demand for effective malware detection systems is only set to rise.

6.2 Future Work

Future research in this field could dive into incorporating more static and dynamic features, along with leveraging cutting-edge machine learning techniques, to boost the effectiveness of malware detection systems. It’s also crucial to look into how emerging threats affect our models and to adapt them for new types of malware, ensuring our detection framework remains effective.

Bibliography

[1] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 2013, pp. 300-305, doi: 10.1109/ICTAI.2013.53.

[2] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," in IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3216-3225, July 2018, doi: 10.1109/TII.2017.2789219.

[3] Akbar, F.; Hussain, M.; Mumtaz, R.; Riaz, Q.; Wahab, A.W.A.; Jung, K.-H. "Permission-Based Detection of Android Malware Using Machine Learning," Symmetry, 2022.

[4] Zhao Xiaoyan, Fang Juan and Wang Xiujuan, "Android malware detection based on permission," 2014 International Conference on Information and Communications Technologies (ICT 2014), Nanjing, China, 2014, pp. 1-5, doi: 10.1049/cp.2014.0605.

[5] Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, Guillermo Suarez-Tangil, Steven Furnell, "AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection," Computers and Security, Volume 65, 2017, Pages 121-134, ISSN 0167-4048.

[6] M. W. Afridi, T. Ali, T. Alghamdi, T. Ali and M. Yasar, "Android application behavioral analysis through intent monitoring," 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 2018, pp. 1-8, doi: 10.1109/ISDFS.2018.8355359.

[7] M. Kumaran and W. Li, "Lightweight malware detection based on machine learning algorithms and the android manifest file," 2016 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, USA, 2016, pp. 1-3, doi: 10.1109/URTC.2016.8284090.

[8] K. Khariwal, J. Singh and A. Arora, "IPDroid: Android Malware Detection using intent and permission," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 197-202, doi: 10.1109/WorldS450073.2020.9210414.

[9] F. Idrees and M. Rajarajan, "Investigating the android intent and permission for malware detection," 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Larnaca, Cyprus, 2014, pp. 354-358, doi: 10.1109/WiMOB.2014.6962194.

[10] A. Kapoor, H. Kushwaha and E. Gandotra, "Permission based Android Malicious Application Detection using Machine Learning," 2019 International Conference on Signal Processing & Communication (ICSC), NOIDA, India, 2019, pp. 103-108, doi: 10.1109/ICSC45622.2019.8938236.

[11] Nisarg Patel, Avesta Sasan, and Houman Homayoun, "Analyzing Hardware Based Malware Detectors," in Proceedings of the 54th Annual Design Automation Conference 2017 (DAC '17). Association for Computing Machinery, New York, NY, USA, Article 25, 1–6.

[12] M. Dhalaria and E. Gandotra, "Android Malware Detection using Chi-Square Feature Selection and Ensemble Learning Method," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2020, pp. 36-41, doi: 10.1109/PDGC50313.2020.9315818.

[13] Mahindru, A., Sangal, A.L. "MLDroid—framework for Android malware detection using machine learning techniques," Neural Comput & Applic, 33, 5183–5240 (2021).

[14] A. Arora, S. Garg and S. K. Peddoju, "Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices," 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, Oxford, UK, 2014, pp. 66-71, doi: 10.1109/NGMAST.2014.57.

[15] Anshul Arora and Sateesh K. Peddoju, "Minimizing Network Traffic Features for Android Mobile Malware Detection," in Proceedings of the 18th International Conference on Distributed Computing and Networking (ICDCN '17). Association for Computing Machinery, New York, NY, USA, Article 32, 1–10.

[16] Mahindru, A., Sangal, A.L. "PerbDroid: Effective Malware Detection Model Developed Using Machine Learning Classification Techniques," in: Singh, J., Bilgaiyan, S., Mishra, B., Dehuri, S. (eds) A Journey Towards Bio-inspired Techniques in Software Engineering. Intelligent Systems Reference Library, vol 185. Springer, Cham.

[17] Y. Lu, P. Zulie, L. Jingju and S. Yi, "Android Malware Detection Technology Based on Improved Bayesian Classification," 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control, Shenyang, China, 2013, pp. 1338-1341, doi: 10.1109/IMCCC.2013.297.

[18] Sharfah Ratibah Tuan Mat, Mohd Faizal Ab Razak, Mohd Nizam Mohmad Kahar, Juliza Mohamad Arif, Ahmad Firdaus, "A Bayesian probability model for Android malware detection," ICT Express, Volume 8, Issue 3, 2022, Pages 424-431, ISSN 2405-9595.

[19] L. D. Coronado-De-Alba, A. Rodríguez-Mota and P. J. Escamilla-Ambrosio, "Feature selection and ensemble of classifiers for Android malware detection," 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), Medellin, Colombia, 2016, pp. 1-6, doi: 10.1109/LATINCOM.2016.7811605.

[20] S. R. Tiwari and R. U. Shukla, "An Android Malware Detection Technique Using Optimized Permission and API with PCA," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 2611-2616, doi: 10.1109/ICCONS.2018.8662939.

[21] K. Zhao, D. Zhang, X. Su and W. Li, "Fest: A feature extraction and selection tool for Android malware detection," 2015 IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 2015, pp. 714-720, doi: 10.1109/ISCC.2015.7405598.

[22] J. Y. Ndagi and J. K. Alhassan, "Machine Learning Classification Algorithms for Adware in Android Devices: A Comparative Evaluation and Analysis," 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2019, pp. 1-6, doi: 10.1109/ICECCO48375.2019.9043288.

[23] Shabtai, A., Elovici, Y. "Applying Behavioral Detection on Android-Based Devices," in: Cai, Y., Magedanz, T., Li, M., Xia, J., Giannelli, C. (eds) Mobile Wireless Middleware, Operating Systems, and Applications. MOBILWARE 2010. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 48. Springer, Berlin, Heidelberg.

[24] Varna Priya D, Visalakshi P, "Detecting android malware using an improved filter based technique in embedded software," Microprocessors and Microsystems, Volume 76, 2020, 103115, ISSN 0141-9331.

[25] C. P. Chenet, A. Savino and S. Di Carlo, "A Survey on Hardware-Based Malware Detection Approaches," in IEEE Access, vol. 12, pp. 54115-54128, 2024, doi: 10.1109/ACCESS.2024.3388716.

[26] R. B. Hadiprakoso, H. Kabetta and I. K. S. Buana, "Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection," 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, 2020, pp. 8-12, doi: 10.1109/ICIMCIS51567.2020.9354315.

[27] Kimberly Tam, Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, and Lorenzo Cavallaro, "The Evolution of Android Malware and Android Analysis Techniques," ACM Comput. Surv. 49, 4, Article 76 (December 2017), 41 pages.

[28] E. C. Bayazit, O. Koray Sahingoz and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2020, pp. 1-8, doi: 10.1109/HORA49412.2020.9152840.

[29] A. Sangal and H. K. Verma, "A Static Feature Selection-based Android Malware Detection Using Machine Learning Techniques," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2020, pp. 48-51, doi: 10.1109/ICOSEC49089.2020.9215355.

Details of Candidate's Publication

The Paper titled **" InPermHardroid: Android Malware Detection using Intents, Permissions and Hardware Components"** was accepted in Scopus Indexed conference titled " 2025 The 18th International Conference on Computer and Electrical Engineering (ICCEE 2025)", which will be held on 20^{th} - 22^{th} June 2025.

Acceptance Letter

(Full Paper for Presentation and Publication)

https://iccee.org/

Registration deadline: June 1st, 2025

Paper ID	E-117
Title	InPermHardroid: Android Malware Detection using Intents, Permissions and Hardware Components
Author (s)	Himanshi, Anshul Arora

To whom it may concern,

Congratulations! Now we are pleased to inform you that **your above full paper has been accepted by ICCEE 2025 for presentation and publication** after the reviewing process, monitored by the Technical Program Committee. You are cordially invited to present your paper on the conference. **Your paper will be published in conference proceedings,** when you follow the procedure to finish the registration before the set date. 2025 The 18th International Conference on Computer and Electrical Engineering (ICCEE 2025) will be held in Singapore on June 20-22, 2025. It provides opportunities for the different areas delegates to exchange new ideas and application experiences face to face, to establish business or research relations and to find global partners for future collaboration.

Registration Procedure

1. Revise your paper according to Comments in review form.
2. Format your paper according to the Template attached carefully https://iccee.org/files/IOS/Word-Template.dotx
3. Fill and complete the Registration Form and pay for it online: https://iconf.young.ac.cn/RAUB6
4. Return your registration materials below to iccee_secretary@academic.net before the deadline.
   A. final papers (both doc and pdf version) B. payment proof

Notes:

1. We will feedback registration status in 3 working days after receiving your registration materials. If you don't receive any response, please contact us proactively.
2. For registration fee and cancellation policy, please visit https://iccee.org/registration.html to check.
3. Please do not trust any contact information other than the official website. Be aware of the various types of online fraud and be alert.

Please feel free to contact us if you have any further questions. Look forward to see you in Singapore!

Yours sincerely,
ICCEE 2025 Organizing Committee

CONTACT US
Secretary: Ms. Mila Xiao
Email: iccee_secretary@academic.net
Tel: +86-134 0855 5552
Wechat: Iconf-cs-2

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)

Match Groups

76 Not Cited or Quoted 11%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
5% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **76** Not Cited or Quoted 11%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
5% Publications
7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Carnegie Mellon University on 2024-12-08 <1%
2	Internet	www.researchgate.net <1%
3	Publication	Lecture Notes in Computer Science, 2012. <1%
4	Internet	link.springer.com <1%
5	Publication	Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical ... <1%
6	Internet	www.mdpi.com <1%
7	Submitted works	Heriot-Watt University on 2025-03-27 <1%
8	Internet	medium.com <1%
9	Internet	www.coursehero.com <1%
10	Internet	ir.pdpu.ac.in:8080 <1%

11 Internet
core.ac.uk <1%

12 Internet
vitalflux.com <1%

13 Submitted works
Curtin University of Technology on 2015-06-08 <1%

14 Publication
Garima Kumari, Anshul Arora. "Smartphone Malware Detection using Permission... <1%

15 Publication
Narendra M. Shekokar, Hari Vasudevan, Surya S. Durbha, Antonis Michalas, Tatw... <1%

16 Submitted works
University of Birmingham on 2018-12-21 <1%

17 Internet
repository.tudelft.nl <1%

18 Submitted works
University of Northumbria at Newcastle on 2025-05-19 <1%

19 Submitted works
Wright State University on 2023-04-24 <1%

20 Internet
fr.slideshare.net <1%

21 Internet
zn63.co-aol.com <1%

22 Submitted works
Griffth University on 2024-05-30 <1%

23 Publication
Raden Budiarto Hadiprakoso, Herman Kabetta, I Komang Setia Buana. "Hybrid-B... <1%

24 Submitted works
Roehampton University on 2023-07-26 <1%

25 Internet
neuroquantology.com <1%

26 Publication
Maria Drogkoula, Konstantinos Kokkinos, Nicholas Samaras. "A Comprehensive S... <1%

27 Internet
campatho.files.wordpress.com <1%

28 Internet
stax.strath.ac.uk <1%

29 Internet
theemcoe.org <1%

30 Internet
www2.mdpi.com <1%

31 Publication
"Data Management, Analytics and Innovation", Springer Science and Business Me... <1%

32 Submitted works
535 on 2015-02-11 <1%

33 Publication
Aditya Kapoor, Himanshu Kushwaha, Ekta Gandotra. "Permission based Android ... <1%

34 Publication
Boadu, Bernard O.. "Dual Vision: Enhancing Autonomous Navigation With AutoG... <1%

35 Submitted works
CSU Northridge on 2024-05-03 <1%

36 Submitted works
Chester College of Higher Education on 2023-10-05 <1%

37 Submitted works
Glasgow Caledonian University on 2025-04-17 <1%

38 Submitted works
Indian Institute of Technology, Kanpur on 2016-01-11 <1%

39 Submitted works
Liverpool John Moores University on 2024-02-02 <1%

40 Publication
Sahil Sharma, Bhavna Arora. "Chapter 61 On Static Android Malware Detection an... <1%

41 Submitted works
University of Wales, Lampeter on 2023-09-13 <1%

42 Internet
iku.unirepos.com <1%

43 Internet
www.ej-eng.org <1%

44 Internet
www.nature.com <1%

45 Submitted works
Australian National University on 2025-05-08 <1%

46 Submitted works
Chester College of Higher Education on 2023-12-13 <1%

47 Submitted works
Chester College of Higher Education on 2024-03-27 <1%

48 Publication
H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Co... <1%

49 Publication
Jinxin Liu, Michele Nogueira, Johan Fernandes, Burak Kantarci. "Adversarial Mach... <1%

50 Submitted works
Liverpool John Moores University on 2024-08-26 <1%

51 Submitted works
National Institute Of Technology, Tiruchirappalli on 2023-05-05 <1%

52 Submitted works
University of Auckland on 2024-04-25 <1%

53 Submitted works
University of Lincoln on 2023-09-07 <1%

54 Submitted works
University of Northumbria at Newcastle on 2022-05-05 <1%

55 Submitted works
University of Southampton on 2017-09-08 <1%

56 Publication
Vandana Mohindru Sood, Yashwant Singh, Bharat Bhargava, Sushil Kumar Naran... <1%

57 Publication
William Wolfgang Arrasmith. "Handbook of Systems Engineering and Analysis of ... <1%

58 Internet
aircconline.com <1%

59 Internet
cse.nirmauni.ac.in <1%

60 Internet
etd.auburn.edu <1%

61 Internet
galib19.github.io <1%

62 Submitted works
nith on 2023-05-24 <1%

63 Internet
tel.archives-ouvertes.fr <1%

64 Internet
www.ijert.org <1%

65 Internet
www.rama.mahidol.ac.th <1%

66 Internet
www.researchsquare.com <1%