KWG-RFE: A Tri-Stage Hybrid Feature Reduction Framework for Android Malware Detection

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the award of the degree

MASTER OF SCIENCE (M.Sc.)
in
MATHEMATICS

Submitted by

Kunikaa Dwivedi

(2K23/MSCMAT/29)

Under the supervision of

Dr. Anshul Arora

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

MAY, 2025

Candidate's Declaration

I, Kunikaa Dwivedi, a student of Master of Science (Mathematics) with roll number 2K23/MSCMAT/29, state that the dissertation I turned in to Delhi Technological University's Department of Applied Mathematics, with the title **KWG-RFE: A Tri-Stage Hybrid Feature Reduction Framework for Android Malware Detection**, is completely authentic and free of any copies of other sources without the required citation. Part of the prerequisites for earning a Master of Science in Mathematics degree are met by this.

Place: Delhi
Date: May 26, 2025

Kunikaa Dwivedi
2K23/MSCMAT/29

Certificate

I certify that the project dissertation **KWG-RFE: A Tri-Stage Hybrid Feature Reduction Framework for Android Malware Detection**, which was turned in by Kunikaa Dwivedi, Roll No. 2K23/MSCMAT/29 of the Department of Applied Mathematics at Delhi Technological University, Delhi, as a partial fulfilment of the requirements for the award of the Masters of Science in Mathematics degree, is a record of the work completed by the student under my guidance. To the best of my knowledge, neither this university nor any other has accepted this work in whole or in part for a degree or diploma.

Place: Delhi
Date: May 26, 2025

Dr. Anshul Arora
Supervisor

Acknowledgement

My supervisor, Dr. 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 will always be appreciative of his kind and encouraging demeanour as well as his wise advice, which were crucial to the accomplishment 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 endeavour by offering assistance and facilitating the exchange of pertinent information.

Kunikaa Dwivedi
2K23/MSCMAT/29

Abstract

Mobile computing has been transformed by the quick development and broad use of Android smartphones, but this has also given cybercriminals a larger attack surface. Among these, mobile malware poses the greatest threat to data, privacy, and system integrity for both individuals and companies. As a result, reliable, effective, and precise malware detection techniques that are exclusive to the Android ecosystem are desperately needed. This thesis introduces KWG-RFE, a hybrid feature selection strategy that offers a fresh approach to Android malware detection. This strategy combines three complementing techniques: Recursive Feature Elimination (RFE), Graph-based feature analysis, and the Kruskal-Wallis statistical test. These elements function in concert to filter and rank features according to their significance and effect on classification performance.

The proposed method was validated using a large dataset of over 111,000 Android applications, which included both malicious and benign samples. Each program had a number of components removed, including hardware-related parts, intent filters, permissions, and API calls. In order to reduce dimensionality while maintaining crucial information pertinent to malware identification, these features were subsequently put through the KWG-RFE selection procedure.

Both the full and reduced feature sets were used to train and assess a number of machine learning classifiers, such as Random Forests, Decision Trees, and Support Vector Machines. With a full feature set of 97.75% and a competitive 94.50% accuracy after applying the KWG-RFE feature reduction, the Random Forest classifier showed the highest detection accuracy among them. The findings show that, without compromising classification performance, the suggested hybrid feature selection approach is quite successful at removing superfluous and unnecessary characteristics.

All things considered, this study shows how effective it is to combine algorithmic, structural, and statistical feature selection methods when it comes to Android malware detection. The suggested KWG-RFE approach is a useful addition to the field of mobile cybersecurity since it maintains a high level of accuracy while increasing detection efficiency by lowering computational overhead.

Contents

List of Tables

List of Figures

Introduction

The digital landscape has been completely transformed by the widespread use of Android smartphones, which give billions of users access to business, entertainment, and communication apps at their fingertips. According to recent statistics, Android is the most popular mobile operating system worldwide, powering more than 70% of mobile devices. Although this open-source platform gives developers flexibility and scalability, there are serious security issues with it. The exponential growth in malware specifically designed to target Android is one of the most urgent issues, as it jeopardises the availability, confidentiality, and integrity of user data.

By incorporating malicious code into applications that appear to be harmless and are frequently shared via third-party app stores or even the official Google Play Store, cybercriminals take advantage of the Android ecosystem. These malicious apps have the ability to track user behaviour, install more malware, and steal private data, leading to serious privacy violations and monetary losses. Intelligent, data-driven approaches must be adopted because traditional signature-based detection techniques are frequently insufficient to detect new or evolving malware.

In cybersecurity, machine learning (ML) has become a potent tool, especially for malware detection. ML models are capable of efficiently differentiating between malicious and benign applications by identifying patterns in vast amounts of labelled data. The high dimensionality of the input data, however, presents a significant obstacle to this strategy. Large feature spaces created by static features like intents, hardware access, permissions, and API calls add noise and redundancy, which lowers the classifiers' accuracy and efficiency.

This thesis proposes a novel tri-stage feature selection framework, **KWG-RFE** (Kruskal-Wallis, Graph-based filtering, and Recursive Feature Elimination), to address this problem. By keeping only the most pertinent and discriminative features, the framework seeks to decrease the dimensionality of Android application data. KWG-RFE efficiently reduces big feature sets into a compact subset appropriate for classification by successively implementing statistical, structural, and classifier-based selection techniques.

This thesis's experimental investigation combines malicious samples from publicly accessible malware repositories with benign samples from the Google Play Store, utilising a large dataset of more than 111,000 Android applications. Permissions, hardware indicators, and intents were among the features that were statically extracted and fed into the hybrid feature selection framework. To assess the detection performance on both the full and reduced feature sets, several classifiers were used, such as Decision Tree, Naïve Bayes, and Random Forest.

When trained on the entire feature set, the Random Forest classifier had the highest accuracy of any of the tested models (97.75%), and it continued to maintain a strong

accuracy of 94.50% even after applying KWG-RFE for dimensionality reduction. These outcomes demonstrate how well the suggested feature reduction framework works to preserve a high level of malware detection accuracy while greatly increasing computational efficiency.
In conclusion, this study advances the field of Android malware detection by putting forth a feature selection framework that is highly accurate, computationally efficient, and statistically supported. In addition to improving detection capabilities, KWG-RFE provides a scalable solution that can be integrated into mobile security platforms and used for real-time malware analysis.

1.1 Background

In the last few years, smartphones have become a key element in everyday life, changing the way we communicate, entertain ourselves, be productive, and connect to one another. As we become more reliant upon smartphones to interact with banking, healthcare, shopping, and governmental services, security is of increasing concern.As of January 2025, there were about 2.06 million apps available on the Google Play Store, the main marketplace for Android apps. Due to its dominating 72.15% market share for smartphones worldwide, Android has emerged as a top target for cybercriminals looking to take advantage of security flaws. Concerns regarding malware threats have increased as a result of the growing reliance on mobile devices for sensitive tasks like banking and payments. A substantial number of new apps were added to the Google Play Store in 2024; in December alone, about 41,000 new apps were released.These dangers, which at first frequently result in illegal access, financial fraud, and data breaches, range from basic adware and spyware to more complex attacks involving ransomware, trojans, and botnets.

Researchers have created a number of detection techniques that can be broadly divided into static, dynamic, and hybrid analysis in order to combat mobile malware. Static analysis provides a rapid and resource-efficient way to detect malware by examining an application's code and metadata without running it. In contrast, dynamic analysis uses a controlled environment to observe how the application behaves. To improve detection accuracy, hybrid analysis blends the two methods. Permissions, intents, and hardware components are among the static features that are frequently examined in Android malware detection because they are important markers of potentially harmful activity. It is difficult to effectively distinguish malware because, as prior research has demonstrated, many of these characteristics are shared by malicious and benign applications. To find the most distinctive qualities, a thorough feature selection and ranking procedure is required due to this overlap.

1.2 Motivation

The shortcomings and difficulties of the current static analysis-based malware detection methods served as the catalyst for this investigation. First off, there are over a hundred different permissions in the Android permission infrastructure, which quickly introduces high dimensionality. This results in overfitting, longer training times, and poor detection model generalisation. Additionally, some permissions depend on the context; for instance, a permission that might seem unwanted in one kind of application (like `READ_CONTACTS`

in a game) might be perfectly acceptable in another (like a messaging app). Adopting a more sophisticated feature selection methodology that permits nuance in feature selection is therefore imperative.

This paper proposes a hybrid feature selection model that uses three feature selection methodologies: the classifiers' Recursive Feature Elimination (RFE) technique, Graph-based filtering to evaluate structure and relational attributes among features, and the Kruskal-Wallis test to evaluate relevant statistical relevance. By combining these techniques, the hybrid model will be able to identify the best subset of permissions and other characteristics that distinguish malicious and benign applications.
Following the removal of features which are redundant or carry little informative significance, overall model performance should be enhanced while maintaining computational complexity at an acceptable level. The outcome of this study will produce a lightweight, interpretable and accurate Android malware classifier model capable of deployment across existing mobile security products.

1.3 Contribution

This thesis presents the following significant contributions:

- Presents a novel hybrid feature selection framework that sequentially uses the Kruskal-Wallis statistical test for filtering the features, Graph-based correlation analysis is used for removing redundant features, and finally, Recursive Feature Elimination (RFE) is used for refining the feature set under classifier instruction.
- Provides the building blocks for a feature pruning pipeline that is capable of leveraging features in a systematic manner while smartly balancing the three axes of relevance, diversity and predictive power, overcoming the challenges associated with a single-stage approach to feature selection.
- The feature extraction of a comprehensive feature set of static characteristics, including permissions, hardware components and intents, from over 112,000 Android applications, demonstrates a significant coverage of the endless static characteristics.
- Provides compelling evidence based on the analysis of multiple static malware detection models that the inclusion of as few as 10 selected features provides a level of classification accuracy that is strong and in line with what could be achieved using a full feature set, thereby allowing the possibility for lightweight malware detection.
- Provides a comprehensive contextual evaluation of the hybrid method in comparison to selecting features using individual and combined feature selection methods all using a number of machine learning models.
- Provides a clear understanding that using a reduced hybrid-selected feature set, the Random Forest model, achieved a peak accuracy of 94.50%.

1.3.1 Why This is a Novel Approach

What is novel about this approach is that it brings together three independent, but complementary feature selection approaches into one cohesive pipeline that can be applied in an Android malware detection setting. Each of the individual approaches, the Kruskal-Wallis test (statistical classification), graph-based filtering (topological) and RFE (classifier-based elimination) have been used in isolates in the past literature but have not yet been studied together in a mobile malware detection study.

This paper establishes:

- a sequential application of these approaches to ensure both statistical significance and independence of the features;
- a process that reduces over 297 features to just 10 while maintaining classification accuracy; and
- adapts this tri-stage feature selection process to an exhaustive Android dataset of over 112,000 applications. The hybrid engineered engineered model presented here allows for improved performance, while ensuring both interpretability, scalability, and deployability throughout lightweight mobile security application development.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

- An overview of the suggested KWG-RFE hybrid feature selection method is given in Chapter 2, along with an introduction to the fundamental ideas of Android malware and important features like permissions, intents, and hardware components.
- A review of related work is given in Chapter 3, which covers permission-based detection, hybrid approaches, feature selection strategies, and static and dynamic analysis techniques.
- The dataset and the pre-processing procedures used to extract and prepare features from Android applications are covered in detail in Chapter 4.
- The suggested methodology is described in Chapter 5, which also goes into detail about the machine learning classifiers used for detection and each step of the hybrid feature selection pipeline (KWG-RFE).
- A thorough analysis of the experimental findings is given in Chapter 6, which also compares different feature combinations and highlights important model performance findings.
- A summary of the results and research contributions is provided in Chapter 7, which brings the thesis to a close.
- The future scope of the work and its possible social impact are examined in Chapter 8, especially as it relates to improving Android security.

Foundations of Android Malware Detection using Hybrid Feature Selection (KWG-RFE)

It is essential to have a basic understanding of the fundamental ideas pertaining to Android malware detection before diving into the technical implementation and outcomes of our suggested approach. This chapter presents:

1. What is Android Malware?
2. What are Permissions, Intents, and Hardware Features in Android?
3. What is Feature Selection and Why is it Important?
4. What are the Kruskal-Wallis Test, Graph-based Feature Analysis, and Recursive Feature Elimination?

2.1 What is Android Malware?

Malware, which stands for "malicious software," is any software that is purposefully made to harm a client, server, network, or device. Malware frequently infiltrates Android devices through untrusted apps, taking advantage of security flaws in the system or abusing permissions that have been granted.

Types of Android Malware

- **Trojans:** Appear authentic while carrying out nefarious activities in the background.
- **Ransomware:** Locks or encrypts data until a ransom is paid.
- **Spyware:** Secretly collects sensitive user data.
- **Adware:** Shows intrusive advertisements, frequently with ulterior motives.
- **Worms and Botnets:** Malware that replicates itself and is used to remotely control devices.

Android is a prime target for these threats due to its openness and broad adoption.

2.2 Android Features: Permissions, Intents, and Hardware

Permissions

Android permissions control who can access private information and hardware. Applications that want to access a user's contacts, location, or microphone must ask for permission. For instance:

- `READ_CONTACTS`
- `ACCESS_FINE_LOCATION`
- `RECORD_AUDIO`

Intents

Components communicate with one another using intents, which are messaging objects. Intent filters can be used by malicious apps to intercept or reroute operations.

Hardware Components

If device hardware (such as the camera, GPS, and accelerometer) is not properly controlled, it can be misused for surveillance or data leakage.

2.3 Feature Selection in Malware Detection

Many of the thousands of features in a dataset are redundant or unimportant. Feature selection improves interpretability, boosts model performance, and decreases dimensionality.

2.4 The KWG-RFE Method

KWG-RFE is a hybrid feature selection framework combining:

- **Kruskal-Wallis Test:** A statistical test that is non-parametric and ranks features according to how well they can differentiate between classes.
- **Graph-Based Feature Analysis:** Constructs a feature affinity graph to find features that are influential and highly connected.
- **Recursive Feature Elimination (RFE):** Based on the performance of a trained model, iteratively eliminates the least significant features.

This hybrid method reduces the amount of high-dimensional Android app data (hardware, intents, and permissions) to a manageable subset for classification.

Further chapters detail the application of KWG-RFE in Android malware detection.

Related work

Android malware detection has been largely investigated using static and dynamic analysis approaches. A classified review of the primary areas of research into the prominent avenues is presented, emphasizing permission-based detection, feature extraction using the hybrid approach, and feature selection techniques.

3.1 Static vs Dynamic Analysis

Over the past ten years, mobile security research has placed a lot of emphasis on detecting Android malware. Static, dynamic, and hybrid analysis techniques are the three main categories into which approaches can be separated; each has unique advantages and disadvantages.

Static analysis is a fast and highly scalable method of analysing applications without actually running them. Permissions, manifest components, opcodes, control flow graphs, and API calls are among the features that are frequently extracted [1, 6, 25]. For instance, it was shown by DREBIN [1] that highly interpretable malware classification using linear SVMs could be accomplished with lightweight static features. Similarly, to find zero-day vulnerabilities in Android apps, RiskRanker [6] created a scalable static framework. Compact and semantically rich feature spaces for detection are also provided by opcode-based methods [25] and manifest analysis [23].

On the other hand, **dynamic analysis** runs programs in sandbox settings to watch for runtime events like network activity, file system changes, and system calls. Such behavioural traces are captured by programs like DroidScope and ANDRUBIS [4]. Dynamic analysis has scalability problems and is vulnerable to evasion if the malicious payload is only activated under certain circumstances, even though it frequently provides higher resistance to code obfuscation.

Numerous researchers have suggested **hybrid approaches** that combine static and dynamic features to achieve balanced detection capabilities because of the trade-offs involved. These methods seek to improve resilience against evasion techniques by combining complementary perspectives on app behavior [13, 22]. Due to its effectiveness, ease of use, and compatibility with current antivirus software and app stores, static analysis is still the most widely used technique in large-scale frameworks in practice [20, 4].

3.2 Permission-Based Malware Detection

Application permissions stand out among the many static features used in malware detection because they are present in the manifest file and have obvious privacy and security

implications.

Permissions like `READ_SMS`, `WRITE_EXTERNAL_STORAGE`, and `ACCESS_FINE_LOCATION` are strongly associated with malicious activity, according to a number of studies [1, 8, 26]. These permissions allow dangerous actions like changing files and accessing location data, or they grant access to private user information. Rule-based analysis was employed by early systems such as Kirin [2] to identify risky permission combinations.

These concepts were expanded upon in later work. In their analysis of permission abuse in practical applications, Grace et al. [6] underlined the necessity of more precise control mechanisms. VetDroid [14] used policy analysis and selective permission tracking to shed light on app behaviour. Subsequent research used statistical models and graph-based representations to investigate permission correlation. For example, GNN-based models like the one in [18] captured the structural relationships between permissions and behaviour patterns, while PermPair [7] identified collusive permission requests.

Permissions are now modelled in conjunction with other features using sophisticated hybrid techniques to improve accuracy. These include integrating permissions with contextual app metadata [24, 27, 28], intent filters [3], and API call graphs [10]. Detecting implicit permission leaks that might not be apparent from direct usage patterns is another benefit of multilevel permission modelling.

3.3 Hybrid Feature-Based Approaches

While specific features, like permissions or API calls, provide insightful information, combining different feature types improves robustness and generalisation. Hybrid methods combine behavioural and static data to create detection models that are more thorough.

Systems such as DroidAPIMiner [3] and IPDroid [3] have shown how useful it is to combine static features like manifest declarations, app metadata, and API usage. To create enriched feature sets, DroidMalwareDetector [9] used both static and dynamic analysis.

Semantic feature extraction using control flow graphs and data dependency analysis was introduced by deep learning frameworks like DroidSieve [10] and Apposcopy [11]. MalPat [5] and Andro-profiler [12] concentrated on context-aware modelling and sensitive API patterns, providing behavioural fingerprints that are challenging to obfuscate.

While hybrid detection pipelines [22, 30] use convolutional and recurrent neural networks to capture both spatial and sequential patterns in app behaviour, multi-view learning frameworks [13, 19] align multiple feature sets to learn joint representations. These systems achieve superior detection accuracy on a variety of malware datasets and provide improved resistance to popular obfuscation techniques.

3.4 Feature Selection Techniques

Feature selection has become a crucial preprocessing step to improve classifier efficiency and interpretability as feature sets become more dimensional.

Statistical filters such as Information Gain and Chi-square tests are examples of classical techniques that rank features according to their discriminative power [4, 17]. These filters perform well on big datasets and are computationally efficient. Although they require more computing power, wrapper techniques like Recursive Feature Elimination (RFE) [19, 21] use classifier feedback to iteratively evaluate subsets of features.

Mutual information and feature correlation are incorporated into structural feature selection methods. To identify redundant and irrelevant features, for example, [16] investigated contrasting permission patterns. In order to balance performance and computational efficiency, hybrid selection strategies, like those in [17, 24, 13], combine multiple criteria (filter + wrapper, for example).

Semi-supervised techniques and graph-based feature selection have been investigated recently [29]. By combining RFE for optimal subset selection, graph-based filtering for redundancy reduction, and Kruskal-Wallis statistical tests, our suggested KWG-RFE framework expands on these concepts. This pipeline minimises overhead while guaranteeing interpretability and generalisation.

3.5 Summary

To sum up, the literature on Android malware detection covers a broad range of approaches, from deep learning models that make use of hybrid representations to static analysis that employs manifest-based features. Multi-view learning, API usage mining, and permission analysis have all helped to increase detection accuracy.

Choosing features is still a crucial step in creating effective and scalable models. Although a number of filter, wrapper, and hybrid approaches have been put forth, interpretable and computationally balanced methods are still required. In order to close this gap, we present a unified selection framework (KWG-RFE) that improves performance and transparency by combining recursive elimination, redundancy pruning, and statistical significance testing into a single pipeline.

Dataset and Preprocessing

Two different sources were used to create a balanced dataset: malicious samples were taken from the AndroZoo project, and benign Android apps were gathered from the Google Play Store. 111,010 APK files in all, equally divided between 55,505 samples of malware and 55,505 samples of benign software, were used.

To extract its manifest file (`AndroidManifest.xml`), each APK file was processed using `Apktool`. To determine the requested permissions, `<uses-permission>` tags were parsed from this file. Throughout the dataset, 129 distinct permissions were found. The first feature matrix was created by encoding the existence or lack of a permission as binary values (1 or 0) for every application.

- Benign Apps: 56,000 apps from the Google Play Store
- Malware Apps: 56,00 apps from the AndroZoo dataset.

4.1 Feature Extraction

Features were extracted statically from the AndroidManifest.xml of each application. The extracted categories were:

- Permissions: 129 unique permissions
- Intent Filters: Event-driven components
- Hardware Features: GPS, Camera, Bluetooth, etc.

Each feature was expressed in binary, where 1 represents the feature being present and 0 represents it absent.

4.2 Preprocessing Steps

- To facilitate the analysis of the APK files, we initially employed a series of decompilation tools, including Apktool and Androguard.
- We then processed the extracted code into feature matrices using scripts written in Python.
- A straightforward label encoding scheme assigned 1 to malware and 0 to benign samples.
- Finally, the extracted dataset was normalized, and the data was split into training and separate testing datasets for evaluation purposes.

Methodology

This section presents an overview of the end-to-end methodology implemented in Python using Google Colab notebooks. The workflow encompasses the following stages: data loading, cleaning, merging, hybrid feature selection, and model performance evaluation using various machine learning algorithms. The implementation of these steps leverages the following Python libraries: **pandas**, **scikit-learn**, **seaborn**, **matplotlib**, and **networkx**.

5.1 Hybrid Feature Selection Pipeline

The key logic from the implementation included:

- Merging Datasets: The permissions, intents, and hardware datasets were loaded from .xlsx files and merged on a common apk identifier. Data were checked for duplicates and null values.
- **Label Creation:** A binary label `Malware_Label` was generated by checking whether the APK ID started with '1'. This indicated malware, while all other cases were considered benign.
- **Step 1: Kruskal-Wallis Test:** A custom function iterated over each feature and applied the Kruskal-Wallis H-test to compare the distributions between benign and malware groups. Features with p-values < 0.01 were selected.
- **Step 2: Graph-Based Filtering:** A correlation matrix was computed among selected features. Features with correlations > 0.7 were connected as nodes in a graph. For each connected component, only one representative feature was retained.
- **Step 3: Recursive Feature Elimination (RFE):** The reduced feature set was passed through RFE using a RandomForestClassifier to select the top 10 features that most contributed to classification accuracy.

The entire process was implemented in a series of modular functions, and visualizations, such as confusion matrices and accuracy bar plots, were provided to better compare models. To enhance detection capability while alleviating dimensionality, we applied a hybrid feature selection pipeline featuring:

- **Kruskal-Wallis Test:** A non-parametric statistical test for assessing feature relevance across the malware and benign cohorts,

- **Graph-Based Feature Filtering:** A method that constructs a correlation graph of all features, retaining central-node features.
- **Recursive Feature Elimination (RFE):** A backward feature selection method to iteratively remove at least useful features based on the model's performance measurements.

The hybrid pipeline allows for selecting the top 10-50 features, based on the combination being tested.

5.2 Hybrid Feature Selection Pipeline (KWG-RFE)

In order to mitigate redundancy while maximizing model performance, a three-stage hybrid feature selection process was applied:

1. **Kruskal-Wallis Test:**
   - To identify features which have the most significant distributional differences between the malware and benign classes, we performed a non-parametric statistical test. We selected features that resulted in a p-value ≤ 0.01 for further analysis.
   - The test statistic H is given by:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1)$$

     where:
     - N = total number of observations,
     - k = number of groups (in this case, 2),
     - R_i = sum of ranks in group i,
     - n_i = number of observations in group i.
2. **Graph-Based Filtering:**
   - A correlation matrix C is computed:

$$C_{i,j} = \frac{\text{cov}(X_i, X_j)}{\sigma_{X_i} \sigma_{X_j}}$$

   - Feature pairs with Pearson's correlation ≥ 0.7 were clustered into groups using graphical representation, and for each group, we selected one feature based on centrality.
3. **Recursive Feature Elimination (RFE):**
   - The features from the previous steps were passed to RFE using Random Forest as the underlying estimator, followed by selecting a final group of 10 features using variable importance weights.

- The accuracy A of the model is evaluated using:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

This pipeline allowed us to confirm statistical relevance, reduce multicollinearity, and maintain features with the highest predictive power, which overall helped to improve model performance.

5.3 Classification Models

To evaluate model robustness and consistency across feature subsets, the following classifiers were trained using 5-fold Stratified Cross Validation:

- Decision Tree: Creates interpretable decision paths by dividing data according to feature importance.
- Logistic Regression: Models the probability that an app is malware based on a linear combination of selected features.
- Random Forest (n=50 estimators):An ensemble model that improves accuracy by reducing overfitting by combining several decision trees.
- K-Nearest Neighbors (KNN): Classifies based on the majority label of nearest training samples in feature space.
- Gaussian Naive Bayes: Applies Bayes' theorem assuming feature independence and Gaussian distributions.

Mean accuracy, standard deviation, and confusion matrices were used as evaluation metrics. Results were displayed both in tabular form and as bar plots using Seaborn and Matplotlib libraries.

Results and Discussion

6.1 Feature Combination Comparison

Table 6.1: Individual Accuracies of Top 10 RFE Permission Features

Feature	**Accuracy**
MOUNT_UNMOUNT_FILESYSTEMS	0.8577
READ_PHONE_STATE	0.8519
GET_TASKS	0.8471
CHANGE_WIFI_STATE	0.8467
SYSTEM_ALERT_WINDOW	0.8039
WRITE_SETTINGS	0.7878
READ_LOGS	0.7504
RECEIVE	0.6030
BIND_GET_INSTALL_REFERRER_SERVICE	0.5919
FOREGROUND_SERVICE	0.5686

Table 6.2: Individual Accuracies of Top 10 RFE Hardware Features

Feature	**Accuracy**
touchscreen	0.5897
camera	0.5761
location.network	0.5560
telephony	0.5362
screen.portrait	0.5355
screen.landscape	0.5340
vulkan	0.5269
bluetooth	0.5181
nfc.hce	0.5179
NFC	0.5055

Combinations of the three fundamental static feature groups— permissions, intents, and hardware — were assessed with and without the hybrid feature selection pipeline (KWG-RFE). The number of features prior to and after each selection phase is presented in the table below: The models were evaluated in combinations of feature groups (Intents, Permissions, Hardware) with and without feature selection.

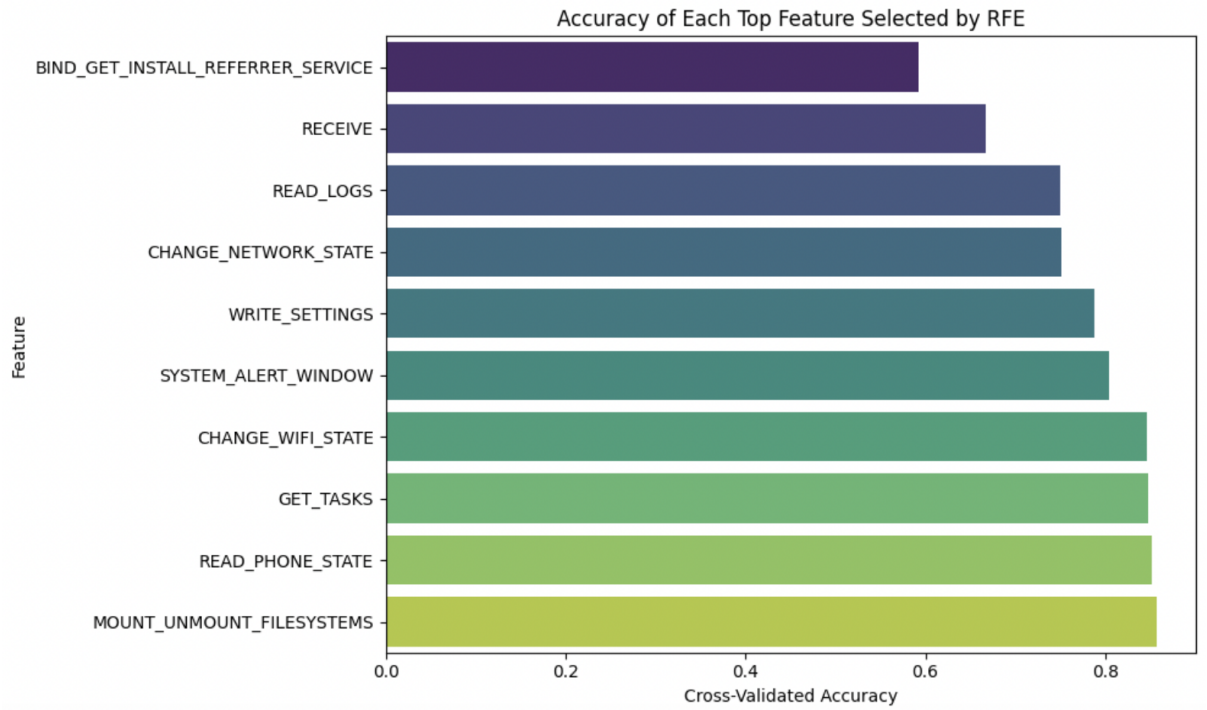

Figure 6.1: Individual Accuracies of Top 10 RFE Permission Features

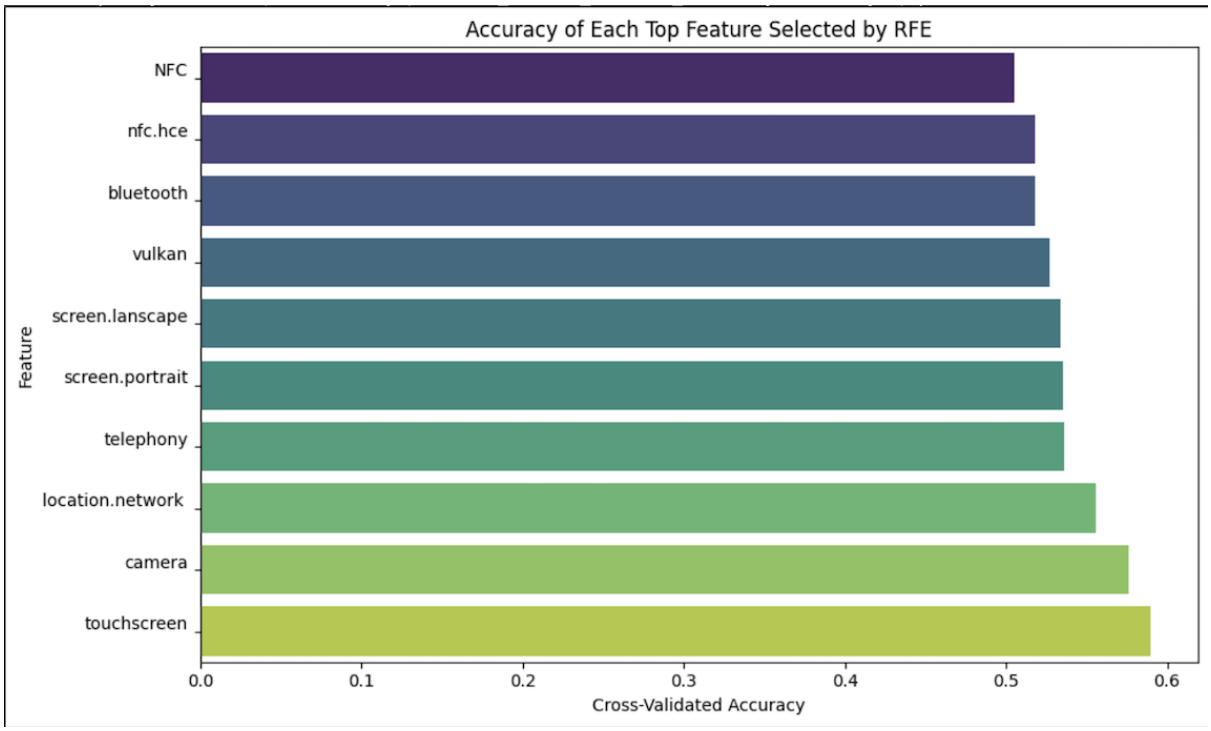

Figure 6.2: Individual Accuracies of Top 10 RFE Hardware Features

6.2 Feature Set Combinations

We have analyzed the various combinations of the three primary static feature categories—permissions, intents, and hardware—using both the hybrid feature selection pipeline, KWG-RFE, and without it. Below we show the original number of features, in

Table 6.3: Individual Accuracies of Top 10 RFE Intent Features

Feature	**Accuracy**
DEFAULT	0.7178
DaemonService	0.6109
RECEIVE	0.6030
BOOT_COMPLETED	0.5905
MEDIA_MOUNTED	0.5647
LEANBACK_LAUNCHER	0.5493
START_FROM_AGOO	0.5477
MY_PACKAGE_REPLACED	0.5348
PUSH	0.5274
ELECTION_RESULT_V4	0.5116

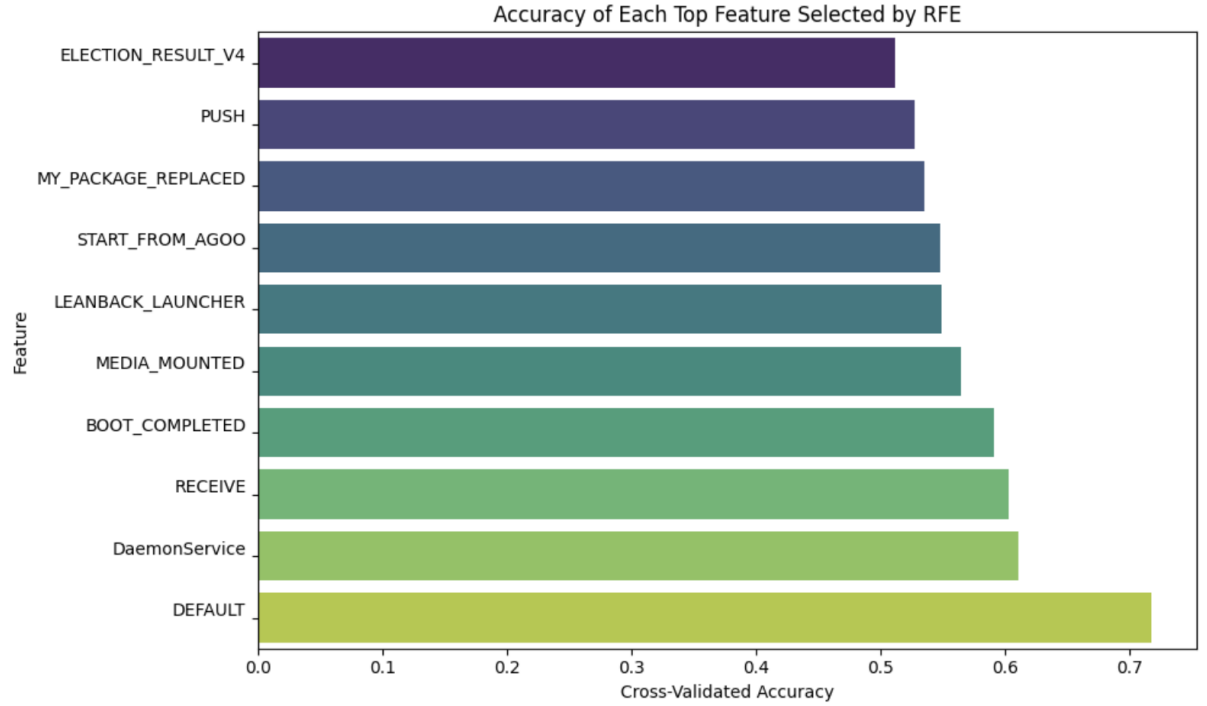

Figure 6.3: Individual Accuracies of Top 10 RFE Intent Features

addition to the number of features derived from each selection technique stage:

6.3 Key Observations

The accuracy rates across feature selection stages and classifiers are summarized below:

We assessed the stand-alone accuracy of the top 10 features in order to gain a better understanding of the predictive power of each feature chosen by RFE. Table 6.6 lists the accuracies of a Random Forest classifier that was trained independently using each feature. The most important characteristics influencing the overall model performance are highlighted by these findings.

Random Forest was used to assess each RFE-selected feature separately. Their cross-validated malware detection accuracy is displayed in the figure below.

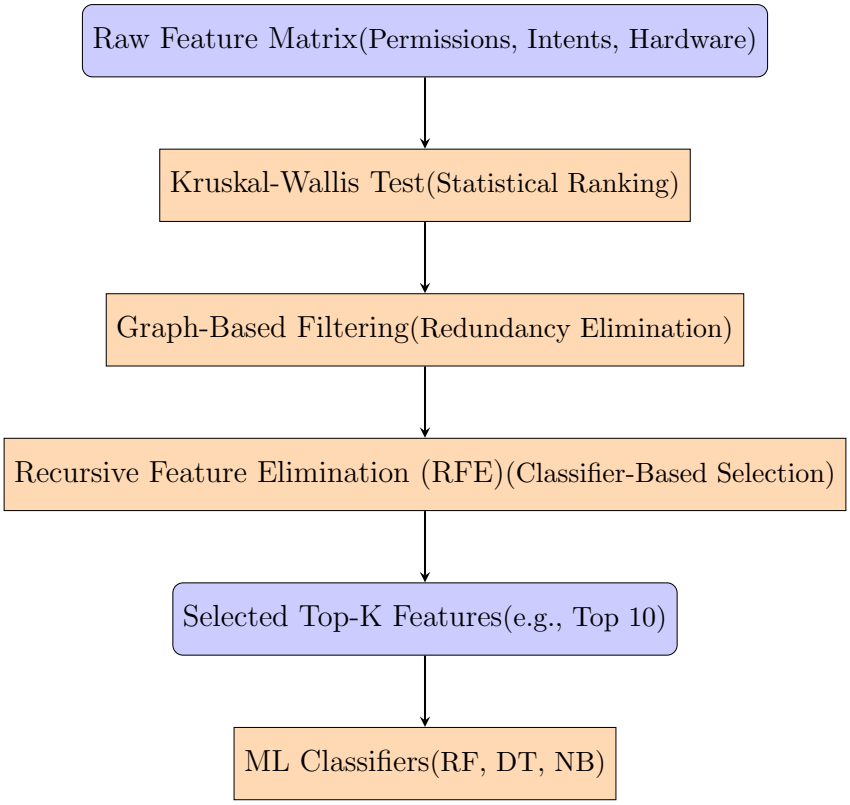

Figure 6.4: Hybrid Feature Selection Pipeline

The findings suggest that the KWG-RFE strategy maintained strong detection accuracy across a variety of models, while significantly reducing feature dimensionality. The Random Forest classifier yielded the highest performance, with other models following closely behind.

- Robust Accuracy Across Models: Random Forest maintained an impressive detection accuracy of 94.50% using only the top 10 features chosen by KWG-RFE. Decision Tree (94.49%), Logistic Regression (94.16%), and K-Nearest Neighbors (94.16%) were the next closest contenders.
- Dimensionality reduction: Up to 297 features (combined Permissions, Intents, Hardware) were included in the full feature set without feature selection. This was reduced to just 10 features after using the KWG-RFE pipeline, and the classification

Table 6.4: Feature Reduction Across Combinations

Feature Set	**Raw**	**Kruskal**	**Graph**	**RFE**
Intents + Permissions	209	206	138	10
Intents + Hardware	168	143	86	10
Permissions + Hardware	218	191	151	10
All Combined	297	270	184	10

Table 6.5: Accuracy Across Feature Reduction Stages

Features Used	**Feature Count**	**Model**	**Mean Acc. (%)**	**Std Dev**
All Features	297	Naive Bayes	90.78	0.0041
		Decision Tree	96.93	0.0011
		Random Forest	97.75	0.0010
		KNN	96.87	0.0017
		Logistic Regression	96.00	0.0014
After Kruskal	270	Naive Bayes	89.18	0.0038
		Decision Tree	96.78	0.0010
		Random Forest	97.73	0.0007
		KNN	96.96	0.0008
		Logistic Regression	96.02	0.0011
After Graph	184	Naive Bayes	90.04	0.0036
		Decision Tree	96.67	0.0006
		Random Forest	97.64	0.0006
		KNN	96.94	0.0013
		Logistic Regression	95.85	0.0008
After RFE	10	Naive Bayes	93.93	0.0015
		Decision Tree	94.49	0.0012
		Random Forest	94.50	0.0013
		KNN	94.16	0.0013
		Logistic Regression	94.16	0.0016

Table 6.6: Individual Accuracies of Top 10 RFE Combined Features

Feature	**Accuracy**
MOUNT_UNMOUNT_FILESYSTEMS	0.8577
READ_PHONE_STATE	0.8519
GET_TASKS	0.8471
CHANGE_WIFI_STATE	0.8467
SYSTEM_ALERT_WINDOW	0.8039
WRITE_SETTINGS	0.7878
READ_LOGS	0.7504
RECEIVE	0.6030
BIND_GET_INSTALL_REFERRER_SERVICE	0.5919
FOREGROUND_SERVICE	0.5686

performance was barely affected.

- Stage-wise Feature Pruning:
  - The Kruskal-Wallis Test found statistically significant features that distinguish between malicious and benign applications, reducing the original feature set by roughly 10–30
  - By eliminating highly correlated features, graph-based filtering removed re-

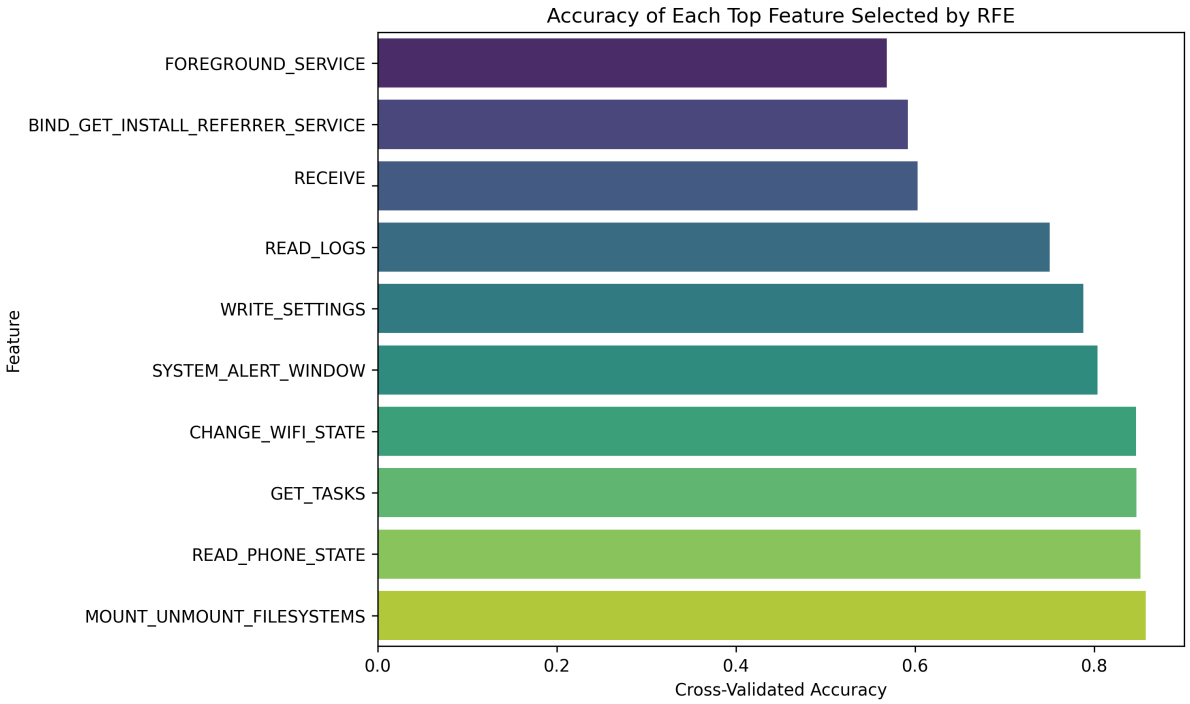

Figure 6.5: Cross-validated accuracy of top 10 features selected using RFE

dundancy.

  - The most predictive features were selected using Recursive Feature Elimination (RFE).

- Consistency and Generalization: Even after vigorous feature reduction, the KWG-RFE method demonstrated good generalization across all classifiers with little loss in accuracy.

- Effectiveness in Practice: Google Colab experiment results verified that the approach maintained interpretability and robustness while lowering computational complexity.

- For the entire feature set (permissions + intents + hardware), an example reduction would be:
(Kruskal-Wallis) → (Graph-Based) → (RFE) — *resulting in 94.50% accuracy using Random Forest.*

6.4 Summary

In order to reduce noise, remove redundant features, and keep only the most informative features, the suggested KWG-RFE (Kruskal-Wallis, Graph-based filtering, and Recursive Feature Elimination) method worked incredibly well. This technique achieved high detection accuracy with a significantly smaller set of features by drastically reducing the feature space while maintaining critical classification power. In addition to speeding up the model training process, the final reduced feature set showed remarkable suitability for real-time malware detection on resource-constrained environments, like embedded systems and smartphones. The method strikes a balance between efficiency, interpretability,

and accuracy, making it a workable option for scalable implementation in actual Android malware detection systems.

Conclusion

Here, we present a scalable and reliable framework for detecting Android malware. It is based on a hybrid feature selection pipeline called KWG-RFE, which combines three potent methods: Recursive Feature Elimination (RFE), graph-based correlation analysis, and the Kruskal-Wallis statistical test. By ensuring that only the least redundant and statistically significant features are kept for classification, this multifaceted approach enhances the model's interpretability and efficiency.

To find the most informative subset, the suggested approach methodically reduces a large number of static features, such as hardware components, intents, and permissions. Our model maintains a high level of accuracy while drastically reducing computational overhead by narrowing the feature space to just 10 highly discriminative features. An extensive dataset of 112,000 Android applications, balanced between benign and malicious samples, was used to create this optimised subset.

Our system demonstrated its effectiveness and practicality for large-scale malware detection by achieving a detection accuracy of 94.50% despite this significant reduction in dimensionality. The findings demonstrate that rigorous feature selection, informed by graph theory and statistical testing, can produce malware detection systems that are both lightweight and highly effective, making them appropriate for implementation in actual mobile security frameworks.

Future Scope and Social Impact

8.1 Future Scope

The suggested KWG-RFE framework has shown great promise in raising the precision and effectiveness of Android malware detection. Nonetheless, there are still a number of encouraging avenues for further study:

1. **Deployment on Mobile Devices:** Improving Android devices' real-time malware detection framework can provide users with proactive security. Mobile deployment will require optimisation for computational efficiency and battery consumption.

2. **Integration of Dynamic Analysis:** Future research can include dynamic behavioural features (such as runtime behaviour and network activity) to increase resilience against advanced malware techniques like code obfuscation and polymorphism, even though this study is based on static features.

3. **Cross-Dataset Validation:** The model should be validated using malware samples from various datasets and sources, such as zero-day exploits and emerging threats, in order to evaluate generalisability.

4. **Deep Learning and Transfer Learning:** To further improve detection performance, particularly in identifying malware variants that have not yet been discovered, future research may investigate deep learning techniques or transfer learning.

5. **Explainable AI (XAI):** Security analysts will gain a better understanding of model decisions and increase confidence in automated malware detection systems by incorporating explainability into the detection framework.

6. **Adversarial Robustness:** Future studies should concentrate on strengthening the system's resilience against hostile inputs and changing threat landscapes as attackers create ways to avoid detection.

8.2 Social Impact

The KWG-RFE framework offers substantial advantages to society in addition to advancing technology:

1. **Enhanced User Security:** The framework helps shield users from identity theft, financial fraud, data theft, and unauthorised surveillance by enhancing malware detection on Android devices.

2. **Preservation of Digital Trust:** Since smartphones are becoming essential for banking, healthcare, education, and communication, accurate and effective malware detection helps to maintain public confidence in mobile technologies.

3. **Reduced Economic Loss:** Effective detection systems can help people, companies, and governments reduce the billions of dollars in economic losses caused by mobile malware every year.

4. **Contribution to Cybersecurity Research:** This work offers a new and reproducible hybrid feature selection technique that strikes a balance between efficiency and performance, which supports cybersecurity efforts in academia and industry.

5. **Support for Law Enforcement and Policy:** Reliable malware detection tools help law enforcement and the creation of public policy by helping cybercrime units identify malicious apps and track down the origins of attacks.

Bibliography

[1] S. Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens, "DREBIN: Effective and explainable detection of Android malware in your pocket," in NDSS, 2014.

[2] Enck, William & Ongtang, Machigar & McDaniel, Patrick. (2009). "On Lightweight Mobile Phone Application Certification". Proceedings of the ACM Conference on Computer and Communications Security. 235-245.

[3] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-level features for robust malware detection in Android," in International Con- ference on Security and Privacy in Communication Systems, Springer, 2013.

[4] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. v. d. Veen and C. Platzer, "ANDRUBIS – 1,000,000 Apps Later: A View on Current Android Malware Behaviors," 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), Wroclaw, Poland, 2014, pp. 3-17

[5] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: A behavioral malware detection framework for Android devices," Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161–190, 2012.

[6] Grace, Michael & Zhou, Yajin & Zhang, Qiang & Zou, Shihong & Jiang, Xuxian. (2012). RiskRanker: Scalable and Accurate Zero-day Android. Proc. of the 10Th International Conference on Mobile Systems, Applications, And. 10.1145/2307636.2307663.

[7] Amalina, Fairuz & Feizollah, Ali & Anuar, Nor & Gani, Abdullah. (2014). Evaluation of machine learning classifiers for mobile malware detection. Soft Computing. 20. 10.1007/s00500-014-1511-6.

[8] Odat, Esraa & Yaseen, Qussai. (2023). A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3244656.

[9] Wu, Wen-Chieh & Hung, Shih-Hao. (2014). DroidDolphin. 247-252. 10.1145/2663761.2664223.

[10] Suarez-Tangil, Guillermo & Dash, Santanu & Ahmadi, Mansour & Kinder, Johannes & Giacinto, Giorgio & Cavallaro, Lorenzo. (2017). DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware. 10.1145/3029806.3029825.

[11] Feng, Yu & Anand, Saswat & Dillig, Isil & Aiken, Alex. (2014). Apposcopy: Semantic-based detection of android malware through static analysis. 576-587. 10.1145/2635868.2635869.

[12] S. H. Moghaddam and M. Abbaspour, "Sensitivity analysis of static features for Android malware detection," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2014.

[13] Ibrahim, Mohammed & Abdullahi, Abdullahi & Ahmad, Muhammad Aminu & Mustapha, Rabi & Ng, & Ibrahim, Mohammed. (2023). "A Comparative Analysis of Android Malware Detection with and without Feature Selection Techniques using Machine Learning". SLU Journal of Science and Technology. 10.56471/slujst.v6i.371.

[14] J. Zhang, M. Yang, B. Xu, Z. Wang, and Y. Liu, "VetDroid: A selective permission analysis tool for smartphone applications," in Proceedings of ACM CCS, 2013.

[15] S. Zhang, J. Yan, L. Chen, H. Xu, and G. Gu, "Vetting Undesirable Behaviors in Android Apps with Permission Use Analysis," in Proceedings of the 21st USENIX Security Symposium, 2012.

[16] P. Xiong, X. Wang, W. Niu, T. Zhu and G. Li, "Android malware detection with contrasting permission patterns," in China Communications, vol. 11, no. 8, pp. 1-14, Aug. 2014.

[17] N. Tiwari, I. Gupta, D. Sethi, S. Marwaha and V. Jha, "A Review on Ransomware Detection in Android Using Feature Selection and Machine Learning," 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS), Gurugram, India, 2024, pp.

[18] P. Feng, J. Ma, T. Li, X. Ma, N. Xi and D. Lu, "Android Malware Detection Based on Call Graph via Graph Neural Network," 2020 International Conference on Networking and Network Applications (NaNA), Haikou City, China, 2020, pp. 368-374.

[19] Kumar, Harsh & Chopan, Aijaz. (2024). Hybrid ML-DL Approach for Android Malware Detection. 10.21203/rs.3.rs-5358924/v1.

[20] H. Yin, "Android Malware Detection Using Convolutional Neural Networks and Light Gradient Boosting Machine: A Hybrid Method," 2024 6th International Conference on Internet of Things, Automation and Artificial Intelligence (IoTAAI), Guangzhou, China, 2024, pp. 75-79

[21] Srastika, N. Bhandary, S. R. S, P. Honnavalli and S. E, "An Enhanced Malware Detection Approach using Machine Learning and Feature Selection," 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2022, pp. 909-914.

[22] A. Prayoga, R. B. Hadiprakoso, R. N. Yasa and Girinoto, "Deep Learning for Android Malware Detection and Classification Using Hybrid-Based Analysis: A Comparative Study," 2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs), Bogor, Indonesia, 2023, pp. 309-313.

[23] X. Li, J. Liu, Y. Huo, R. Zhang and Y. Yao, "An Android malware detection method based on AndroidManifest file," 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), Beijing, China, 2016, pp. 239-243.

[24] L. Ma, Y. Yang, X. Wang and J. He, "Ultra-Lightweight Malware Detection of Android Using 2-Level Machine Learning," 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, China, 2016, pp. 729-733.

[25] V. Sihag, A. Mitharwal, M. Vardhan and P. Singh, "Opcode n-gram based Malware Classification in Android," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 645-650, doi

[26] S. Jadhav, T. T. Oh, J. P. Jeong, Y. H. Kim and J. N. Kim, "An Assistive System for Android Malware Analysis to Increase Malware Analysis Efficiency," 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, Taiwan, 2017, pp. 370-374, doi

[27] J. Wang, B. Li and Y. Zeng, "XGBoost-Based Android Malware Detection," 2017 13th International Conference on Computational Intelligence and Security (CIS), Hong Kong, China, 2017, pp. 268-272, doi

[28] R. D. Prayogo and S. A. Karimah, "Hybrid Feature Selection with K-Nearest Neighbors for Optimal Heart Failure Detection," 2022 12th International Conference on System Engineering and Technology (ICSET), Bandung, Indonesia, 2022, pp. 101-105, doi

[29] Shangzhi Zheng and Hualong Bu, "A novel semi-feature selection method based on hybrid feature selection mechanism," 2010 2nd International Conference on Advanced Computer Control, Shenyang, China, 2010, pp. 590-593, doi

[30] C. Ryan and M. Diviya, "Malware Image Classification with Enhanced Feature Extraction Using VGG16 and Spatial Pyramid Pooling," 2024 IEEE 8th International Conference on Information and Communication Technology (CICT), Prayagraj UP, India, 2024, pp. 1-6, doi

[31] 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

[32] A. Arora and S. K. Peddoju, "NTPDroid: A Hybrid Android Malware Detector Using Network Traffic and System Permissions," 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 2018, pp. 808-813, doi

[33] A. 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, 2017.

[34] A. Arora, S. K. Peddoju, V. Chouhan, and A. Chaudhary, "Hybrid Android Malware Detection by Combining Supervised and Unsupervised Learning", In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18). Association for Computing Machinery, New York, NY, USA, 798–800, 2018.

[35] A. Arora, S. K. Peddoju and M. Conti, "PermPair: Android Malware Detection Using Permission Pairs," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1968-1982, 2020.