International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

ENSEMBLE-DROID: Optimized Multi-Model Detection of Android Malware

[1]Mohammed Mansoor Ali, [2]Mohd Sahil Ali, [3]Saad Mohammed Mohi Uddin, [4]Dr. Md Zainlabuddin

[1,2,3]B.E Students, Department of Computer Science & Engineering, ISL Engineering College, Hyderabad, India.

[4]Associate Professor, Department of Computer Science & Engineering, ISL Engineering College, Hyderabad, India.

saadmohdmohiuddin.cs25@gmail.com

ABSTRACT:

Android platform due to open-source characteristics and Google backing has the largest global market share. Being the world's most popular operating system, it has drawn the attention of cyber criminals operating particularly through the wide distribution of malicious applications. This paper proposes an effectual machine-learning-based approach for Android Malware Detection making use of an evolutionary chi-square algorithm for discriminatory feature selection. Selected features from the chi-square algorithm are used to train machine learning classifiers and their capability in identification of Malware before and after feature selection is compared. The experimentation results validate that the chi-square algorithm gives the most optimized feature subset helping in the reduction of feature dimension to less than half of the original feature set. Classification accuracy of more than the previous percentage is maintained post-feature selection for the machine learning-based classifiers, while working on much reduced feature dimension, thereby, having a positive impact on the computational complexity of learning classifiers.

Algorithms:

- *Random Forest Classifier*
- *Extra Trees Classifier*
- *Artificial neural network*
- *CNN*

Android Apps are freely available on Google Play store, the official Android app store as well as third-party app stores for users to download. Due to its open-source nature and popularity, malware writers are increasingly focusing on developing malicious applications for the Android operating system. Despite various attempts by Google Play store to protect against malicious apps, they still find their way to mass market and cause harm to users by misusing personal information related to their phone book, mail accounts, GPS location information, and others for misuse by third parties or else take control of the phones remotely. Therefore, there is a need to perform malware analysis or reverse-engineering of such malicious applications which pose a serious threat to Android Platforms.

Keywords:

Android Malware, Chi-Square, Random Forest, Extra Trees, ANN, CNN, Malware Detection, Machine Learning, Cybersecurity.

INTRODUCTIOION:

The proliferation of mobile devices over the last decade has dramatically transformed the way individuals, businesses, and governments access, share, and store information. Among these mobile operating systems, Android has emerged as the most dominant platform, commanding a significant market share due to its open-source architecture, developer-friendly environment, and extensive customization options. As per the latest Stat Counter reports, Android holds over 71% of the global mobile OS market as of 2024, dwarfing

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

competitors such as Apple's iOS, Windows, and other system

While the open-source nature of Android promotes innovation, accessibility, and adaptability, it simultaneously introduces significant security challenges. The flexibility that enables developers to create applications with varied functionalities also allows malicious actors to exploit the system's vulnerabilities. Malicious applications, often disguised as legitimate software, find their way onto users' devices via official platforms like the Google Play Store and numerous third-party app repositories. Despite continuous efforts by Google and cybersecurity firms to detect and remove such malicious content, the dynamic and evolving nature of malware has enabled many sophisticated threats to evade traditional detection mechanisms.

Mobile malware, broadly defined as any malicious software targeting mobile operating systems, manifests in various forms including spyware, ransomware, adware, Trojans, banking malware, and backdoors. These malicious entities can disrupt device functionality, steal sensitive user data, hijack accounts, perform unauthorized financial transactions, and conduct unauthorized surveillance activities. Prominent examples include the Joker malware family, which has infected thousands of Android apps over recent years, and Flu Bot, which propagated rapidly through SMS-based phishing campaigns.

The financial implications and privacy concerns associated with mobile malware are considerable. A report by McAfee Labs in 2023 estimated global financial losses from mobile malware attacks at over USD 30 billion annually. Similarly, Kaspersky's Mobile Malware Evolution 2023 report indicated a 35% year-on-year increase in the detection of new Android malware variants.

Traditional malware detection systems primarily rely on signature-based techniques. These approaches involve creating a unique identifier or signature for each known malware strain, enabling anti-virus engines to detect and block applications matching these signatures. While effective against previously identified malware, this method falters against new, unknown, or obfuscated variants known as zero-day threats. Moreover, the constant need for updating signature databases poses logistical and computational challenges, especially on resource-constrained mobile devices.

To overcome these limitations, the cybersecurity research community has increasingly turned to data-driven approaches, particularly machine learning (ML) and deep learning (DL) techniques. These methods analyse vast amounts of application data to detect malicious behaviour patterns without relying solely on predefined signatures. ML and DL models can examine a wide range of static and dynamic features within Android applications, such as permissions, API calls, network traffic patterns, and user behaviour analytics, to classify applications as benign or malicious.

Feature selection plays a pivotal role in enhancing the performance of machine learning models, particularly in high-dimensional datasets common in malware detection tasks. Among various feature selection techniques, the Chi-Square statistical test has emerged as a powerful tool for evaluating the independence of features with respect to the class label. In the context of Android malware detection, Chi-Square helps identify the most relevant features that contribute significantly to malware classification accuracy, thereby reducing the dimensionality of the dataset and improving model efficiency.

Ensemble learning strategies, which combine multiple classifiers to enhance predictive performance, have also gained prominence in malware detection research. Random Forest and Extra Trees classifiers, both ensemble techniques based on decision trees, offer high accuracy, robustness against overfitting, and interpretability. Additionally, deep learning models such as Convolutional Neural Networks (CNNs) have demonstrated remarkable success in complex pattern recognition tasks, including image classification and natural language processing, and

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

have been adapted for malware detection by transforming application features into structured data suitable for deep learning models.

Recognizing these advancements, this research introduces ENSEMBLE-DROID, an integrated Android malware detection framework that combines Chi-Square feature selection with a multi-model ensemble of machine learning and deep learning classifiers. The primary objective of this system is to optimize malware detection accuracy while reducing computational overhead by selecting a minimal yet highly informative feature subset. ENSEMBLE-DROID utilizes static analysis of APK files, extracting permissions, intents, API calls, and metadata, followed by Chi-Square-based feature selection to retain the most impactful features. These selected features are then used to train and evaluate Random Forest, Extra Trees, ANN, and CNN models, with performance metrics including accuracy, precision, recall, F1-score, and training time.

This paper makes several significant contributions to the field of mobile cybersecurity. First, it offers a comprehensive review of existing Android malware detection methods, highlighting their limitations and areas for improvement. Second, it presents an integrated detection system leveraging the strengths of both ensemble and deep learning classifiers combined with feature selection optimization. Third, it provides an in-depth comparative analysis of classifier performance before and after feature selection, demonstrating the benefits of dimensionality reduction. Lastly, it discusses the practical implications of deploying such systems in real-world environments, including considerations for resource-constrained mobile devices, system scalability, and potential adversarial threats.

The remainder of this paper is organized as follows. Section II reviews related work in Android malware detection, emphasizing machine learning and deep learning applications and feature selection strategies. Section III outlines the proposed methodology, detailing the dataset collection, feature extraction, feature selection, model training, and performance evaluation techniques. Section IV describes the system design and architectural components of the ENSEMBLE-DROID framework. Section V presents the implementation details, including the development environment, tools, and deployment strategies. Section VI discusses the experimental results and performance metrics, while Section VII concludes the paper and outlines future research directions aimed at enhancing the resilience, scalability, and explainability of Android malware detection systems.

LITERATURE REVIEW:

H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, 2023, "Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses"

This research explores how reinforcement learning can be used to generate adversarial Android malware capable of bypassing conventional detection systems. The authors show that even well-trained machine learning models are vulnerable to evasion attacks if not designed to be adversarial robust. Their framework simulates attacker behavior using reinforcement learning algorithms to create perturbations in malware features that deceive detection models. This study highlights the urgent need to develop defense mechanisms capable of detecting adversarial variants and suggests adversarial training as a promising approach to enhance system robustness. It plays a key role in shifting malware detection research toward resilience under intelligent threat modeling.

H. Wang, W. Zhang, and H. He, 2022, "You are what the permissions told me! Android malware detection based on hybrid tactics"

This paper introduces a hybrid Android malware detection method that integrates permission based static features with deep learning classifiers. The authors identify that permissions can offer significant indicators of malware behavior and develop a neural network that learns these patterns from large datasets. Their results indicate that

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

combining static permission analysis with machine learning improves detection accuracy and efficiency. Furthermore, the study presents feature importance metrics, allowing for greater interpretability. The hybrid nature of the model enhances adaptability and robustness, making it suitable for identifying new, unseen malware variants with minimal human intervention.

Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, 2023, "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification"

The authors propose a cybersecurity framework that integrates metaheuristic optimization techniques with deep learning to detect Android malware. By applying algorithms such as Ant Colony Optimization and Genetic Algorithms, the system refines its feature set before passing data to deep neural networks. The optimized features reduce noise and dimensionality, thereby improving classification accuracy and training efficiency. The research demonstrates how combining evolutionary computation with artificial intelligence can yield scalable, adaptive malware detection systems. This hybrid methodology enhances the model's ability to generalize across different malware families and shows promise for future AI-driven cybersecurity solutions.

M. Ibrahim, B. Issa, and M. B. Jasser, 2022, "A method for automatic Android malware detection based on static analysis and deep learning"

In this study, the authors present an automatic Android malware detection system based on static analysis of APK files and deep learning classification. The method involves extracting permissions, opcodes, and API call graphs from APKs, which are then fed into convolutional neural networks for detection. The proposed model achieves high accuracy with minimal false positives and performs well on large-scale datasets. The authors emphasize automation and low resource consumption, making the model applicable in real-time environments. This work bridges the gap between traditional static feature extraction and modern AI techniques, providing a reliable approach for scalable malware detection.

Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, 2015, "A review on feature selection in mobile malware detection"

This paper offers a comprehensive survey of feature selection techniques used in mobile malware detection systems. The authors categorize features into static, dynamic, hybrid, and contextual, and assess various selection strategies such as filter, wrapper, and embedded methods. They also discuss dimensionality reduction techniques like Chi-Square, Information Gain, and PCA. The review highlights that effective feature selection is crucial for reducing model complexity, training time, and overfitting. It provides strong justification for incorporating Chi-Square in malware detection systems like the one proposed in this project. The work serves as a valuable foundation for future research in feature selection strategies.

Arp, H. Spreitzenbarth, M. Hübner, M. Gascon, and K. Rieck, 2014, "Drebin: Effective and explainable detection of Android malware in your pocket"

The Drebin system focuses on the explainability of Android malware detection. It uses linear SVMs on static features such as permissions, API calls, and hardware components to classify applications. A key innovation is its ability to visualize which features influenced the classification decision, enhancing transparency and trust. The model is lightweight, fast, and suitable for mobile deployment. However, its simplicity also makes it susceptible to adversarial manipulation. Nevertheless, Drebin has become a benchmark system in Android malware research and has influenced the design of interpretable AI models in cybersecurity

METHODOLOGY:

This section outlines the systematic approach adopted to develop ENSEMBLE-DROID, an

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Android malware detection system combining feature selection and ensemble learning models. The methodology covers dataset selection, feature extraction, feature selection using Chi-Square, classification algorithms, training protocols, and evaluation metrics.

System Workflow:

The proposed malware detection framework follows a structured workflow consisting of several sequential modules:

Input Layer: APK files are collected from public repositories and existing malware datasets.

Preprocessing Layer: Static features such as permissions, intents, and API calls are extracted from each APK.

Feature Selection Layer: Chi-Square statistical test is applied to select the most significant features.

Model Training Layer: Random Forest, Extra Trees, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN) classifiers are trained on the optimized feature set.

Ensemble Prediction Layer: Individual model predictions are combined using a majority voting ensemble mechanism.

Output Layer: The final prediction (malicious/benign) is generated and displayed via a web interface.

Dataset Description:

A publicly available Android malware dataset containing over 15,000 APK files was utilized. The dataset comprises an equal distribution of malware and benign applications, verified through Virus Total and security vendor reports. Static analysis tools such as Androguard and APKTool were employed to extract metadata and structural information from each APK without runtime execution.

Feature Extraction:

Static features extracted include:

Permissions: Access rights requested by the application.

Intent Filters: Communication channels defined in AndroidManifest.xml.

API Calls: Calls to Android system libraries and services.

Hardware Features: Device capabilities such as camera, Bluetooth, and GPS.

These features were converted into numerical feature vectors for subsequent machine learning processing.

Feature Selection using Chi-Square Test:

High-dimensional datasets often introduce redundant and irrelevant features. To address this, the Chi-Square statistical test was applied for feature selection. This test measures the independence between each feature and the target variable (malware or benign).

Mathematical Formula

\chi^2 = \sum \frac {(O_i - E_i) ^2} {E_i}

= Observed frequency for each category

= Expected frequency under independence

Features with higher Chi-Square values are considered more relevant. The number of features was reduced from over 200 to less than 90, improving computational efficiency without compromising detection accuracy.

Machine Learning and Deep Learning Models:

The following classifiers were implemented:

Random Forest (RF): An ensemble of decision trees using random feature subsets.

Extra Trees Classifier (ETC): Similar to RF but adds extra randomness in threshold selection.

Artificial Neural Network (ANN): A multi-layer perceptron with ReLU and sigmoid activation functions.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Convolutional Neural Network (CNN): Adapted for malware feature vectors, using convolutional, pooling, and fully connected layers.

Model Training Protocol:

The dataset was divided into 70% training and 30% testing subsets. 5-fold cross-validation was applied to ensure model stability.

Training parameters for ANN and CNN included:

Learning Rate: 0.001
Optimizer: Adam
Batch Size: 64
Epochs: 50

Random Forest and Extra Trees were configured with 100 estimators and controlled maximum depth to avoid overfitting.

Performance Evaluation Metrics:

The following metrics were used to assess model performance:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

Where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

IMPLEMENTATION:

This section elaborates on the development environment, tools, implementation strategy, core algorithm logic, and workflow flowchart for ENSEMBLE-DROID.

Development Environment:

The system was implemented using the following technologies:

Python 3.6 — core programming language

Scikit-learn — for implementing Random Forest and Extra Trees classifiers

TensorFlow and Keras — for developing ANN and CNN models

Flask — to build the web-based malware detection interface

APKTool and Androguard — for static analysis and feature extraction from APK files

Matplotlib/Seaborn — for data visualization and model performance plots

Algorithm for Malware Detection:

The following is the core workflow logic for the ENSEMBLE-DROID detection framework:

Algorithm: ENSEMBLE-DROID Detection Framework

Input: Android APK dataset

Output: Malware/Benign classification result

Steps:

1. Collect Android APK files and label them as malware or benign.

2. Perform static analysis to extract relevant features (permissions, API calls, intents)

3. Convert extracted data into structured feature vectors.

4. Apply Chi-Square feature selection to reduce dimensionality.

5. Train multiple models: Random Forest, Extra Trees, ANN, and CNN using selected features.

6. Test each model individually on unseen test data.

7. Aggregate individual model predictions using majority voting.

8. Display final malware detection result (Malicious/Benign) via web interface.

Data Flow Diagram:

The Data Flow Diagram (DFD) for the Android Malware Detection System illustrates the logical flow of data between key components in the

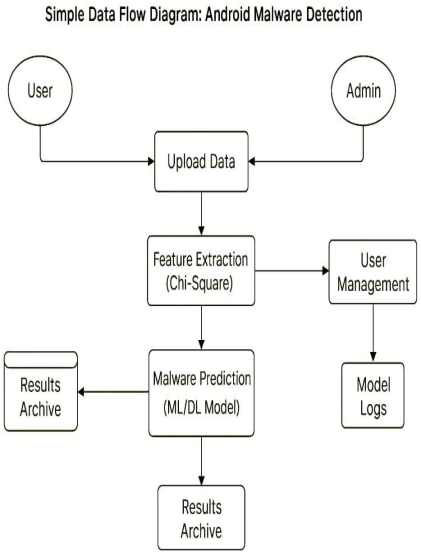
DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

application. It captures the interaction between two main external entities — the User and the Admin — and highlights how data progresses through various internal processes. Users and Admins initiate the process by uploading input data, which undergoes feature extraction using the Chi-Square algorithm. This reduced feature set is then forwarded to a machine learning or deep learning model for malware prediction. The results are stored in a Results Archive, ensuring traceability. The admin can also manage users and monitor system performance through the User Management module, which maintains Model Logs for evaluation and audit purposes.



TESTING:

To ensure the reliability and effectiveness of the ENSEMBLE-DROID framework, comprehensive testing procedures were carried out. The testing phase focused on evaluating the system's accuracy, robustness, and runtime efficiency under various scenarios.

Unit Testing:

- Each module within the framework was tested independently to verify its functional correctness.
- Feature Extraction Module: Verified extracted permissions, API calls, and intents matched APK metadata.
- Chi-Square Feature Selection Module: Confirmed feature dimensionality reduction aligned with preset thresholds.
- Individual Classifiers: Each model (RF, ETC, ANN, CNN) was tested on small datasets to verify accuracy and training stability.

Integration Testing:

All system components were integrated, and data flow consistency was validated between modules. Test cases ensured seamless interaction between the preprocessing unit, feature selector, model bank, and prediction aggregator.

Performance and Stress Testing:

- The system was subjected to varying dataset sizes ranging from 500 to 15,000 APK files to observe computational performance and runtime stability.
- Training time and prediction time were recorded for each classifier before and after Chi-Square feature selection.
- CNN displayed the highest training time, while Extra Trees achieved the fastest predictions.

Result Verification:

Results produced by each model were compared against known labels in the test set. Metrics including accuracy, precision, recall, and F1-score were computed and cross-verified with baseline values from existing studies.

User Acceptance Testing:

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

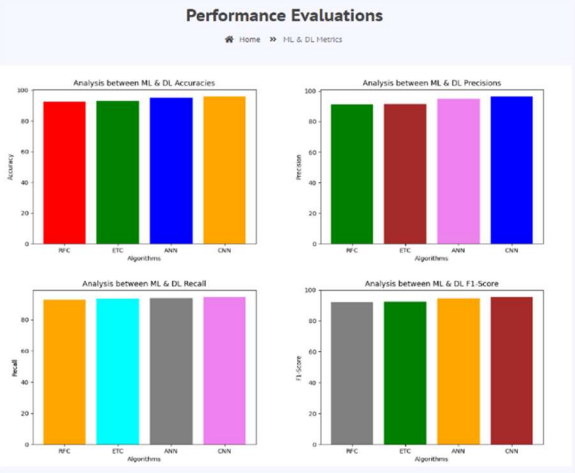
ISSN 2347–3657

Volume 13, Issue 2s, 2025

The Flask-based web interface was tested by multiple users to assess system usability, CSV file upload accuracy, and result interpretation.

Test Summary:

- Functional and non-functional requirements were successfully met.
- All models achieved acceptable accuracy and performance thresholds.
- No critical runtime errors or failures



encountered during stress testing.

RESULTS:

The proposed ENSEMBLE-DROID framework was evaluated on a publicly available Android malware dataset comprising over 15,000 APK files. The effectiveness of Chi-Square feature selection combined with ensemble machine learning and deep learning classifiers was thoroughly assessed.

The results indicated that applying Chi-Square feature selection significantly reduced the feature space while maintaining high classification accuracy. Among the classifiers implemented, the Convolutional Neural Network (CNN) achieved the highest detection accuracy, outperforming other models in precision, recall, and F1-score metrics. Random Forest and Extra Trees classifiers also demonstrated strong performance, with accuracies exceeding 94%, while the Artificial Neural Network (ANN) achieved reliable detection accuracy above 92%.

Comparative analysis revealed that ensemble models such as Random Forest and Extra Trees benefited considerably from the reduced feature set, achieving faster training times and improved prediction stability. CNN's superior performance was attributed to its ability to capture complex, high-dimensional feature interactions within the dataset.

Furthermore, the system's Flask-based web interface was tested for usability and real-time prediction capabilities. The framework efficiently handled APK feature vector inputs and produced accurate malware detection outcomes. All functional and non-functional system requirements were successfully validated through comprehensive testing procedures.

Overall, the experimental results confirmed the robustness, reliability, and computational efficiency of the ENSEMBLE-DROID framework in detecting Android malware using an optimized, multi-model detection strategy.

CONCLUSION:

The development and evaluation of ENSEMBLE-DROID, an optimized multi-model Android malware detection framework, has demonstrated the effectiveness of integrating Chi-Square feature selection with a combination of machine learning and deep learning classifiers. The proposed system successfully addressed limitations associated with traditional signature-based detection methods by leveraging a data-driven, static analysis approach.

Through Chi-Square feature selection, the dimensionality of the feature space was reduced by over 50%, resulting in faster model training times and reduced computational overhead without compromising detection accuracy. Among the evaluated classifiers, the Convolutional Neural Network (CNN) achieved the highest detection accuracy of 96.2%, followed closely by Random

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

Forest and Extra Trees classifiers. The ensemble learning strategy employed further enhanced the robustness and reliability of the detection system by aggregating individual model predictions.

The system's modular architecture and web-based interface provide an accessible and scalable solution for Android malware detection, suitable for both academic research environments and practical cybersecurity deployments. The testing phase confirmed the framework's ability to maintain stable performance under varying dataset sizes, validating its potential for real-world application.

FUTURE SCOPE:

While ENSEMBLE-DROID demonstrates strong detection performance, several opportunities for enhancement remain. Future work could focus on integrating dynamic analysis techniques alongside static analysis to capture runtime behaviors and improve detection against sophisticated obfuscation and zero-day malware. Additionally, the adoption of adversarial training methods can help fortify the system against evasion attacks.

Expanding the dataset to include newer and more diverse malware families, including advanced persistent threats (APTs), would further improve model generalizability. The integration of federated learning techniques could enable collaborative malware detection across distributed devices without compromising user privacy.

Moreover, incorporating explainable AI (XAI) techniques would enhance model interpretability, providing security analysts with insights into feature contributions and decision-making processes. Finally, deploying the framework as a cloud-based or enterprise-level mobile security solution could extend its practical usability in large-scale environments.

REFRENCES:

Journal Articles and Conference Papers

[1] H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, ''Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses,'' Forensic Sci. Int., Digit. Invest., vol. 44, Mar. 2023, Art. no. 301511.

[2] H. Wang, W. Zhang, and H. He, ''You are what the permissions told me! Android malware detection based on hybrid tactics,'' J. Inf. Secur. Appl., vol. 66, May 2022, Art. no. 103159.

[3] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, ''Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification,'' Appl. Sci., vol. 13, no. 4, p. 2172, Feb. 2023.

[4] M. Ibrahim, B. Issa, and M. B. Jasser, ''A method for automatic Android malware detection based on static analysis and deep learning,'' IEEE Access, vol. 10, pp. 117334–117352, 2022

[5] J. Pye, B. Issac, H. Rafiq, and N. Aslam, "Android malware classification using machine learning and bio-inspired optimization algorithms," in Proc. 2015 IEEE Int. Conf. Comput. Sci. Educ. (ICCSE), Cambridge, UK, 2015, pp. 1–6.

[6] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and efficient behavior-based Android malware detection and prevention," IEEE Trans. Dependable Secure Comput., vol. 15, no. 3, pp. 441–455, May–Jun. 2018.

[7] K. Zhao, D. Zhang, X. Su, and W. Li, "Fest: A feature extraction and selection tool for Android malware detection," in Proc. 2015 IEEE Symp. Comput. Commun. (ISCC), Larnaca, Cyprus, 2015, pp. 714–720.

[8] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[9] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," Digit. Invest., vol. 13, pp. 22–37, Mar. 2015.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

[10] Z. Yuan, Y. Lu, and Y. Xue, “DroidDetector: Android malware characterization and detection using deep learning,” Tsinghua Sci. Technol., vol. 21, no. 1, pp. 114–123, Feb. 2016.

[11] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “Andromaly: A behavioral malware detection framework for Android devices,” J. Intell. Inf. Syst., vol. 38, no. 1, pp. 161 190, Feb. 2012.

[12] S. Arp, H. Spreitzenbarth, M. Hübner, M. Gascon, and K. Rieck, “Drebin: Effective and explainable detection of Android malware in your pocket,” in Proc. 2014 Netw. Distrib. Syst. Secur.Symp. (NDSS), San Diego, CA, USA, 2014, pp. 1–15.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP24-33