

Hybrid Random Forest and XGBoost Approach for Android Malware Detection

Abdul Museeb^{1*}  Yaman Hamed²  Hao Louis Tan³ 

Department of Applied Sciences, Universiti Teknologi Petronas, 32610 Seri Iskandar, Perak, Malaysia

*Corresponding Author: Abdul Museeb: e-mail: abdul_24001711@utp.edu.my

Available online at: <https://journal.mas.edu.sy/MASV02E01D02202605>

Received: 25 November 2025

Revised: 27 December 2025

Accepted: 10 January 2026

Abstract— Android malware continues to pose a serious threat to mobile security, frequently evading traditional signature-based detection techniques and compromising sensitive user data. While signature-based approaches are effective against known malware, they struggle to identify obfuscated and evolving threats. To address these limitations, this paper proposes a hybrid machine-learning framework for Android malware detection that integrates Random Forest and XGBoost classifiers. The model relies on static features, specifically API calls and application permissions, which are highly indicative of malicious behaviour. Experiments are conducted using the CIC-AndMal2017 dataset, consisting of labelled benign and malicious Android applications. A systematic feature-selection process is applied to retain the most informative API and permission features. Dimensionality reduction using t-Distributed Stochastic Neighbour Embedding (t-SNE) is employed to improve computational efficiency while preserving meaningful patterns in the data. Experimental results demonstrate that the proposed hybrid model achieves an accuracy of 91.20%, outperforming individual classifiers. The findings highlight the effectiveness of ensemble learning, optimised feature selection, and dimensionality reduction in building scalable and accurate Android malware detection systems.

Keywords— Android Malware Detection; Hybrid classification; API Calls; Permissions; Malware Classification; t-SNE; Android Security.

I. INTRODUCTION

Mobile devices have become indispensable tools in modern life, often surpassing personal computers in daily Internet usage. They support a wide range of activities, such as email communication, social networking, online banking, navigation, and multimedia applications [1]. The increasing storage of sensitive information, including passwords, contacts, and financial data, has made mobile platforms attractive targets for cyberattacks, leading to heightened security concerns, particularly for Android devices[2]. Android is the dominant mobile operating system globally, holding approximately 88% of the market share and powering devices across more than 190 countries [3]. Its popularity is driven by its open-source architecture, extensive hardware support, and large application ecosystem. However, this openness also exposes the platform to significant security threats, as attackers exploit vulnerabilities to distribute malicious applications. To address these risks, multiple protection mechanisms have been proposed, including malware detection, application reinforcement, developer reviews, and vulnerability analysis [4]. Android malware detection remains a critical security measure for preventing malicious applications from being installed or distributed through application marketplaces[5]. Existing detection approaches are commonly classified into static, dynamic, and hybrid techniques [6]. Static detection analyses application code without execution, and is computationally efficient but vulnerable to obfuscation and dynamic code-loading techniques. Dynamic detection observes application behaviour at runtime, enabling the detection of sophisticated threats, but it is resource-intensive. Hybrid detection combines static and dynamic techniques to balance accuracy and efficiency.

Machine-learning-based anomaly detection has been widely adopted across these approaches, enabling the identification of previously unseen malware. In contrast, traditional signature-based methods, commonly used in network intrusion detection systems, rely on known attack patterns and are less effective against evolving threats [7, 8]. Due to Android's open marketplace model, malicious applications can be easily distributed by third-party developers, increasing the risk of data leakage and unauthorised access. Millions of applications are downloaded daily from official and third-party app stores, and despite security mechanisms such as Google Play Bouncer, malware continues to bypass existing defences [9, 10]. These challenges highlight the need for more robust and scalable malware detection solutions.

To overcome the limitations of traditional detection techniques, this study proposes a static-analysis-based Android malware detection framework that leverages anomaly-based machine learning. Unlike approaches that rely on single classifiers or shallow feature representations, this work integrates dimensionality reduction with a hybrid ensemble learning strategy. Specifically, t-SNE is employed to uncover latent feature patterns, followed by a hybrid ensemble model combining Random Forest and XGBoost classifiers. By focusing exclusively on the CIC-AndMal2017 dataset, the proposed approach ensures consistent, reproducible, and interpretable results.

The main contributions of this research are as follows:

- A hybrid ensemble model that combines Random Forest and XGBoost classifiers, achieving superior accuracy, precision, and recall compared to traditional single-classifier approaches.
- Implement a dimensionality reduction approach based on t-SNE to visualise and retain high-value features,

- enhancing classification while reducing computational overhead.
- Perform a thorough evaluation of the hybrid model using key metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, demonstrating its robustness and superior performance in distinguishing between malicious and benign applications.
- This work highlights the importance of permissions and API calls as critical features for static Android malware detection, and shows how combining these features with advanced ensemble learning techniques enhances detection capabilities.

The remainder of this paper is organised as follows: Section 2 reviews related work, Section 3 describes the proposed methodology, Section 4 presents the experimental results and discussion, and Section 5 concludes the paper.

1. RELATED WORK

Android malware detection has been extensively studied as malicious applications continue to evolve, often employing advanced evasion and obfuscation techniques to bypass traditional defences. Early approaches relied primarily on signature-based detection, which is effective against known malware but performs poorly against zero-day and obfuscated threats. Consequently, recent research has shifted toward anomaly-based and hybrid detection techniques supported by machine learning, ensemble learning, feature engineering, and deep learning. TABLE 1 provides an overview of key methodologies, outlining their detection type, datasets, metrics, and core feature sets.

Machine-learning-based static detection has demonstrated strong potential by enabling automated behavioural analysis. Li et al. [11] proposed a lightweight Support Vector Machine (SVM) model using a reduced permission set, achieving 93.62% accuracy, though reliance on limited permissions reduces robustness against novel malware. Anderson et al. [12] proposed a distinctive transformation of Dalvik op-codes into grayscale images, enabling visual pattern recognition using classifiers such as KNN, Decision Trees, and Random Forest. Dynamic analysis approaches focus on runtime behaviour to detect malicious activity missed by static methods. Massarelli et al. [13] analysed resource-consumption patterns, including CPU, memory, and network usage, achieving 94.2% accuracy using SVM. While effective, such methods require sandboxed execution and incur high computational overhead, limiting suitability for real-time or large-scale deployment. These studies highlight trade-offs between detection accuracy, feature richness, and scalability.

To further improve detection accuracy and minimise false positives, ensemble and hybrid approaches have been proposed. Sharma et al. [14] introduced OEL-AMD, an optimised ensemble learning framework incorporating Binary Grey Wolf Optimiser (BGWO)-based feature selection, achieving 92.4% accuracy. Similarly, Sharma and Agrawal [15] combined metaheuristic optimisation with deep learning to perform dimensionality reduction and classification, achieving 94.5% accuracy. Although ensemble models consistently outperform single classifiers, they introduce additional complexity and computational cost, which can hinder deployment on resource-constrained devices. Feature engineering and selection remain critical to effective malware detection. Li et al.[16] proposed the Significant Permission Identification (SigPID) technique to eliminate irrelevant permissions, improving classification performance. Bhagwat and Gupta [17] applied Principal Component Analysis (PCA)

and mutual-information-based feature selection, achieving 92.8% accuracy. Metaheuristic optimisation methods such as Genetic Algorithms and Harris Hawks Optimisation have further enhanced feature selection [18]. However, aggressive feature reduction may remove subtle behavioural indicators, limiting generalisation to emerging malware[19, 20].

TABLE 1: SUMMARY OF ANDROID MALWARE DETECTION APPROACHES

References	Approach	Detection Type	Category	Dataset Used	Performance Metrics	Key Features
[11, 13, 16]	Machine Learning	Static/Hybrid	Anomaly-based	Drebin, CIC-AndMal 2017	Accuracy, Precision, Recall	API calls, Permissions
[16-18]	Feature Engineering	Static/Dynamic	Signature-based	Andro Zoo, Custom	F1-score, Recall	Permissions, Intents
[14, 15, 17]	Ensemble Methods	Static/Hybrid	Anomaly-based	CIC-AndMal 2017, Drebin	Accuracy, Precision, Recall	Combination of classifiers
[6-8]	Signature-based Detection	Static	Signature-based	Custom, Google Play	Detection Rate	Malware signatures
[13, 19, 20]	Dynamic Detection	Dynamic	Anomaly-based	Custom Sandboxing Datasets	Behavioural Metrics	API monitoring, System calls

Overall, prior research reveals a trade-off between detection accuracy, computational overhead, and scalability. These limitations motivate the proposed hybrid framework, which combines Random Forest and XGBoost with efficient feature selection and t-SNE-based dimensionality reduction to achieve strong detection performance while maintaining practicality.

2. METHODOLOGY

The proposed methodology consists of five main stages: dataset collection and preprocessing, feature extraction, dimensionality reduction, model training, and evaluation. A hybrid ensemble model combining Random Forest and XGBoost classifiers is employed, with t-Distributed Stochastic Neighbour Embedding (t-SNE) used to enhance feature representation and reduce computational complexity.

2.1 Dataset Collection and Preprocessing

The CIC-AndMal2017 dataset is used in this study, containing labelled Android APK files categorised as benign, adware, scareware, and SMS malware. During preprocessing, missing values were handled by removing incomplete records to maintain data integrity. Non-numeric and identifier-based attributes were excluded, as they do not contribute to classification performance. The dataset provides a representative and diverse set of benign and malicious applications. Benign applications serve as a baseline, while adware, scareware, and SMS malware reflect common real-world attack behaviours, enabling robust model training and evaluation [21]. Together, these categories provide the foundation for an effective and comprehensive malware detection model, as summarised in the

TABLE 2 below.

TABLE 2: DESCRIPTION OF THE CATEGORIES IN THE CIC-ANDMAL2017 ANDROID MALWARE DATASET

Category	Description
Benign Applications	Non-malicious apps that serve as a baseline for comparison.
Adware	Apps designed to display intrusive advertisements often affect user experience and consume resources.
Scareware	Apps that use deceptive tactics to manipulate users, such as creating false alerts or urging unnecessary purchases.
SMS Malware	Apps that exploit SMS functionality for malicious purposes, like sending unauthorised messages or stealing data.

2.2 Feature Extraction

Based on the categorised dataset described in Section 3.1, static analysis was performed to extract discriminative features from each APK that capture application behaviour and potential security risks. The extracted features are designed to reflect how applications interact with system resources, operate in the background, and communicate with users and the Android operating system. The selected feature categories are as follows:

- I. **API Calls:** Methods invoked within the application that reflect its behaviour and interactions with the system. These are crucial for understanding how an application communicates with the operating system and external services.
- II. **Permissions:** The access rights requested by the application, which can be indicative of potential security risks, such as reading messages or accessing location data. These permissions often highlight critical application functionality and potential threats.
- III. **Services:** Background processes that persistently run in an application, which can be exploited for malicious purposes, such as maintaining a presence on the device without the user's knowledge.
- IV. **Receivers:** Broadcast receivers that handle system-wide events, often used by malware to exploit system or application-level events stealthily.
- V. **Activities:** The user interface components of the application that facilitate interaction with the user, helping to infer the overall intent and functionality of the application.

All extracted features were stored in a structured JSON format for each APK, enabling efficient processing and subsequent analysis. To examine the contribution of different feature combinations, the dataset was organised into multiple configurations:

- I. **API + Permissions + Services:** A comprehensive dataset integrating all three feature types.
- II. **API + Permissions:** Focused on API behaviour and requested permissions.
- III. **API + Services:** Examining API interactions alongside background services.

- IV. **Permissions + Services:** Exploring the link between permissions and background behaviour.
- V. **API Data Alone:** Isolated API calls to assess their standalone predictive power.

These configurations ensured that the data was structured in a way that allowed for efficient processing and loading, optimising the training and testing phases. FIGURE 1 presents the Top 10 Permissions and Top 10 APIs extracted from the dataset, providing valuable insights into the most influential attributes for the model's performance.

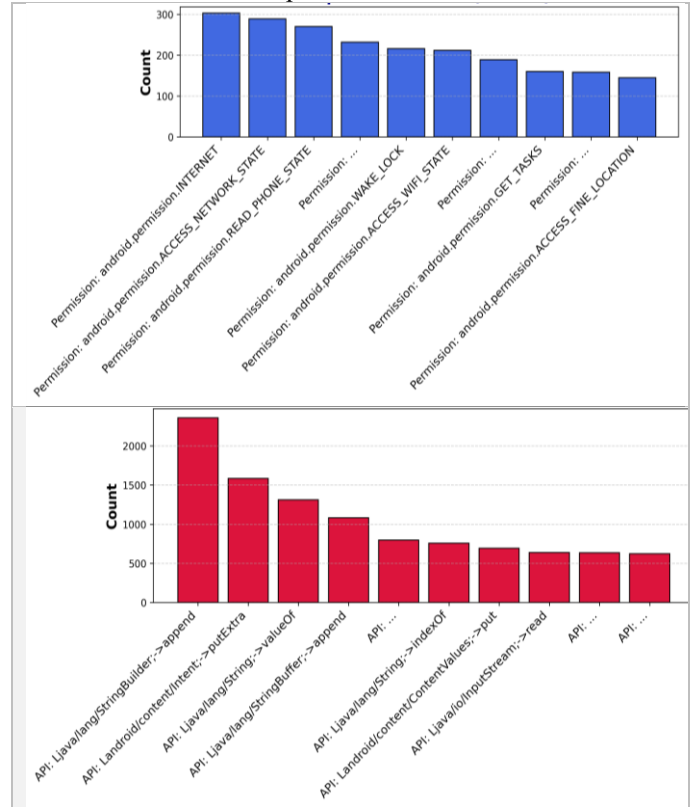


FIGURE 1: TOP 10 PERMISSIONS AND TOP 10 APIS

2.3 Dimensionality Reduction using t-SNE

To address the high dimensionality of the extracted feature space, t-Distributed Stochastic Neighbour Embedding (t-SNE) was applied for dimensionality reduction. After feature scaling, t-SNE projected the data into a two-dimensional space while preserving local relationships between samples. This approach improves visualisation and reduces computational overhead while retaining critical feature patterns.

The mathematical formulation of t-SNE is based on the probability that a data point x_i would pick a neighbouring point x_j as its neighbour, with probabilities defined by a Gaussian distribution:

$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma^2\right)} \quad 1$$

Where p_{ij} represents the probability that the data point x_i and x_j as its neighbour, and σ controls the width of the neighbourhood around each point. t-SNE minimises the divergence between probability distributions in high-dimensional and low-dimensional spaces, enabling meaningful clustering of benign and malicious samples. Below, FIGURE 2 illustrates the two-dimensional representation obtained after dimensionality reduction.

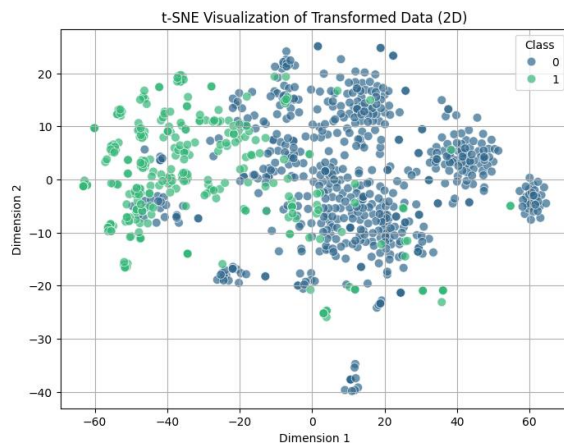


FIGURE 2: 2D VISUALIZATION OF DATA AFTER T-SNE DIMENSIONALITY REDUCTION

2.4 Model Training and Evaluation

A hybrid malware detection model was implemented by combining Random Forest and XGBoost classifiers using a voting-based ensemble strategy. Random Forest constructs multiple decision trees using random feature subsets, while XGBoost applies gradient boosting to iteratively reduce prediction errors. Combining both models improves robustness and generalisation.

The dataset was divided into training (70%) and testing (30%) subsets to ensure reliable evaluation on unseen data. Both the Random Forest and XGBoost classifiers were trained using API calls and permissions, as these features showed strong correlations with malicious behaviour. To optimise model performance, hyperparameter tuning was conducted using GridSearchCV. For the Random Forest classifier, key parameters such as the number of trees, maximum tree depth, and minimum sample thresholds were optimised, while for XGBoost, the learning rate, tree depth, and number of estimators were tuned. The trained models were evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score, which collectively assess overall classification accuracy, false-positive reduction, malware detection capability, and robustness under class imbalance. The model performance was evaluated using the measures of accuracy, precision, recall (sensitivity), and F1-score.

This methodology demonstrates the use of Random Forest for Android malware detection, with feature selection using correlation analysis and PCA for dimensionality reduction. The approach effectively handles the high-dimensional data, improving the model's performance and reducing the computational burden. The Random Forest model, trained on the most relevant features, achieves significant accuracy in detecting Android malware.

3. RESULTS AND DISCUSSION

This section presents the experimental evaluation of the proposed Hybrid Model (RF + XGBoost) and compares its performance with individual Random Forest and XGBoost classifiers. All experiments were conducted using the CIC-AndMal2017 dataset, which was pre-processed and consolidated into a single CSV file. The dataset was divided into training (70%) and testing (30%) subsets, and all models were trained and evaluated using the same data split to ensure a fair comparison. Performance was assessed using standard classification metrics, including accuracy, precision, recall, and F1-score, while confusion matrices were used to analyse classification behaviour. TABLE 3 below presents a summary of the evaluation metrics for each model, including accuracy,

precision, recall, and F1-score.

TABLE 3: EVALUATION METRICS FOR ALL MODELS

Model	Accuracy	Precision	Recall	F1-Score
Proposed Hybrid Model (RF + XGBoost)	0.9120	0.9113	0.8409	0.9190
Random Forest	0.8901	0.8706	0.7957	0.8315
XGBoost	0.8938	0.8810	0.7957	0.8362

As shown in Table 5, the Hybrid Model achieved the highest overall accuracy of 91.20%, outperforming both Random Forest (89.01%) and XGBoost (89.38%). It also obtained the highest precision (0.9113) and F1-score (0.9190), indicating superior capability in distinguishing between benign and malicious applications. While XGBoost demonstrated competitive performance, particularly in precision, the Hybrid Model consistently delivered stronger results across all evaluation metrics, especially recall (0.8409), highlighting its improved ability to detect malware samples.

3.1 Model Comparison and Discussion

In our experiment, the Hybrid Model (RF + XGBoost) significantly outperformed individual classifiers such as Random Forest and XGBoost. The Hybrid Model achieved the highest accuracy of 91.20%, with a precision of 91.13%, a recall of 84.09%, and an F1-score of 91.90%. This demonstrates that combining the strengths of both the Random Forest and XGBoost classifiers results in improved overall performance, with the model being particularly effective in identifying malicious applications with high precision and recall.

Random Forest achieved a slightly lower accuracy of 89.01%, with a precision at 87.06%, a recall at 79.57%, and an F1-score of 83.15%. Although Random Forest performed well, its performance was outperformed by the Hybrid Model, particularly in the precision and F1-score metrics, which highlights the advantage of combining multiple models. Similarly, XGBoost showed a performance close to Random Forest, with an accuracy of 89.38%, precision of 88.10%, recall of 79.57%, and F1-score of 83.62%. However, it still lagged behind the Hybrid Model in all key performance metrics. Despite this, XGBoost performed better than Random Forest in terms of precision for malware detection, indicating that gradient boosting can be effective in reducing false positives.

This comparative analysis underscores the effectiveness of the Hybrid Model (RF + XGBoost) in Android malware detection, where it consistently outperformed the other models in terms of overall accuracy, precision, recall, and F1-score. The Hybrid Model benefits from the strengths of both classifiers, offering a more robust solution for malware classification tasks. Below FIGURE 3 presents a visual comparison of model accuracies, clearly illustrating the superior performance of the Hybrid Model relative to Random Forest and XGBoost.

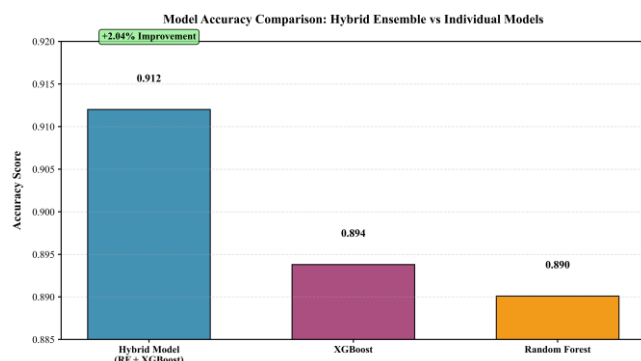


FIGURE 3: MODEL ACCURACY COMPARISON

3.2 Confusion Matrix

To provide deeper insights into the Hybrid Model (RF + XGBoost)'s classification performance, this study presents its confusion matrix, which details the model's effectiveness in distinguishing between benign and malicious applications. The matrix reveals the following key components:

- I. True Positives (TP): 72 instances of malware correctly classified as malware.
- II. True Negatives (TN): 177 instances of benign applications correctly classified as benign.
- III. False Positives (FP): 7 benign applications incorrectly classified as malware.
- IV. False Negatives (FN): 17 malware instances incorrectly classified as benign.

These values are illustrated in the confusion matrix below, which provides a clearer picture of how the model handles false positives and false negatives. The model's relatively low false positive and false negative rates suggest its robustness in identifying both benign and malicious applications.

As shown in

FIGURE 4, the Hybrid Model (RF + XGBoost) effectively classified the majority of both benign and malware instances, with only a small number of false positives and false negatives. The model demonstrated strong calibration, correctly identifying benign applications with high precision and effectively detecting malware with minimal misclassification. These results indicate that the model is proficient in distinguishing between benign and malicious apps, with a balanced performance that minimises both false positives and false negatives.

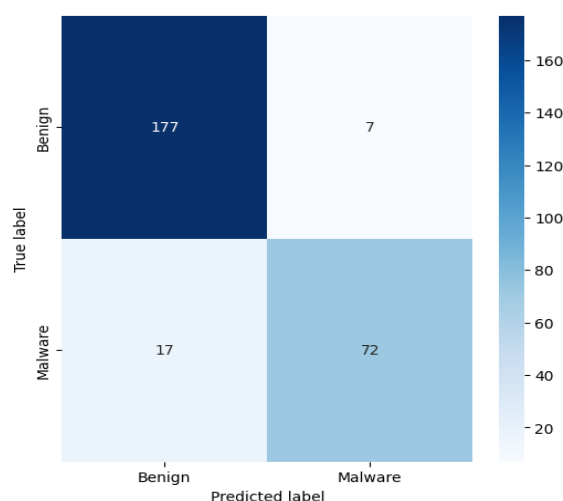


FIGURE 4: CONFUSION MATRIX OF PROPOSED HYBRID MODEL.

4. CONCLUSION

This study presents a robust and scalable Android malware detection framework based on a hybrid ensemble model that combines Random Forest and XGBoost classifiers. By leveraging API calls and permissions as discriminative static features, the proposed approach effectively identifies malicious applications using the CIC-AndMal2017 dataset, which provides a diverse representation of benign and malicious Android apps. A correlation-based feature selection strategy was applied to reduce the original high-dimensional feature space, improving computational efficiency while preserving classification performance. In addition, t-Distributed Stochastic Neighbour Embedding (t-SNE) was employed to enhance feature representation and visualisation by projecting high-dimensional data into a lower-dimensional space. Experimental results demonstrate that the Hybrid Model achieves an accuracy of 91.20%, with strong precision (91.13%), recall (84.09%), and F1-score (91.90%), outperforming individual Random Forest and XGBoost classifiers. Although the framework relies on static analysis and may be limited in detecting malware that employs advanced dynamic evasion techniques, future work will focus on integrating dynamic behavioural features, network activity analysis, and sequential API patterns, as well as exploring advanced ensemble and deep learning strategies to further improve detection robustness and scalability on resource-constrained mobile devices.

Acknowledgement

The authors would like to express their sincere gratitude to Universiti Teknologi PETRONAS (UTP) for the financial support provided under the research grant (YUTP-Cost Centre: 015LC0-533).

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest.

Funding

The authors received no financial support for this book review.

Human Participants

This paper does not involve human participants.

Originality Note

The authors confirm that the manuscript is their original work, and if others' works are used, they are properly cited/quoted.

II. REFERENCES

1. B. D. Deebak and S. O. Hwang, "Healthcare applications using blockchain with a cloud-assisted decentralized privacy-preserving framework," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5897-5916, 2023.
2. M. Chaieb, M. A. Ghorab, and M. A. Saied, "Detecting Android Malware: From Neural Embeddings to Hands-On Validation with BERTroid," *arXiv preprint arXiv:2405.03620*, 2024.
3. M. Gohari, S. Hashemi, and L. Abdi, "Android malware detection and classification based on network traffic using deep learning," in *2021 7th International Conference on Web Research (ICWR)*, 2021: IEEE, pp. 71-77.
4. K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE access*, vol. 8, pp. 124579-124607, 2020.
5. P. Faruki et al., "Android security: a survey of issues, malware penetration, and defenses," *IEEE communications surveys & tutorials*, vol. 17, no. 2, pp. 998-1022, 2014.
6. J. Garcia, M. Hammad, and S. Malek, "Lightweight,

- obfuscation-resilient detection and family identification of android malware," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 26, no. 3, pp. 1-29, 2018.
7. P. Panagiotou, N. Mengidis, T. Tsikrika, S. Vrochidis, and I. Kompatsiaris, "Host-based intrusion detection using signature-based and ai-driven anomaly detection methods," *Information & Security: An International Journal*, vol. 50, no. 1, pp. 37-48, 2021.
 8. L. Li et al., "Static analysis of android apps: A systematic literature review," *Information and Software Technology*, vol. 88, pp. 67-95, 2017.
 9. I. F. Darwin, *Android Cookbook: Problems and Solutions for Android Developers*. "O'Reilly Media, Inc.", 2017.
 10. R. Mayrhofer, J. V. Stoep, C. Brubaker, and N. Kralevich, "The android platform security model," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 3, pp. 1-35, 2021.
 11. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1869-1882, 2014.
 12. H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *arXiv preprint arXiv:1804.04637*, 2018.
 13. L. Massarelli, L. Aniello, C. Ciccotelli, L. Querzoni, D. Ucci, and R. Baldoni, "Android malware family classification based on resource consumption over time," in *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*, 2017: IEEE, pp. 31-38.
 14. S. K. Smmarwar, G. P. Gupta, S. Kumar, and P. Kumar, "An optimized and efficient android malware detection framework for future sustainable computing," *Sustainable Energy Technologies and Assessments*, vol. 54, p. 102852, 2022.
 15. R. M. Sharma and C. P. Agrawal, "MH-DLdroid: A Meta-Heuristic and Deep Learning-Based Hybrid Approach for Android Malware Detection," *Int. J. Intell. Eng. Syst*, vol. 15, pp. 425-435, 2022.
 16. 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 Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216-3225, 2018.
 17. S. Bhagwat and G. P. Gupta, "Android malware detection using hybrid meta-heuristic feature selection and ensemble learning techniques," in *International conference on advances in computing and data sciences*, 2022: Springer, pp. 145-156.
 18. O. A. Alzubi, J. A. Alzubi, A. M. Al-Zoubi, M. A. Hassonah, and U. Kose, "An efficient malware detection approach with feature weighting based on Harris Hawks optimization," *Cluster Computing*, pp. 1-19, 2022.
 19. J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, no. 4, pp. 725-738, 2022.
 20. S. Fallah and A. J. Bidgoly, "Android malware detection using network traffic based on sequential deep learning models," *Software: Practice and Experience*, vol. 52, no. 9, pp. 1987-2004, 2022.
 21. A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan conference on security technology (ICCST)*, 2018: IEEE, pp. 1-7.

About the authors

Abdul Museeb received his B.Sc. degree in Computer Science from the University of Engineering and Technology (UET), Peshawar, Pakistan, in 2022. He is currently pursuing his M.Sc. degree at Universiti Teknologi PETRONAS (UTP), Malaysia, under joint supervision with the University of Limoges, France, and is working as a Research Assistant at UTP. His master's research focuses on developing an intelligent intrusion detection system that integrates advanced feature selection with an efficient gradient boosting approach to detect diverse network attacks and enhance real-time cybersecurity systems. Previously, he worked at the National Database & Registration Authority (NADRA), Pakistan, and has experience as a Web Penetration Tester and Cybersecurity Researcher. His research interests include artificial intelligence, machine learning, cybersecurity, network intrusion detection systems, cryptography, and web application security.

YAMAN HAMED is a Lecturer at the Department of Applied Science, Universiti Teknologi PETRONAS (UTP), Malaysia. He holds a PhD in Applied Sciences from the Statistics Cluster within the Fundamental and Applied Sciences Department at UTP. His research focuses on statistics, applied machine learning, and data analytics, with interdisciplinary applications in cybersecurity, hydrogen production, flow assurance, and financial market clustering. Dr Hamed has been actively involved in research collaborations across various sectors, including oil and gas, predictive maintenance, public health, finance, and cybersecurity. He works closely with industry partners such as PETRONAS, BURSA Malaysia, PRSB, and IBM, among others. He has also secured multiple research grants from government agencies, industry stakeholders, and international collaborators.

Tan Eng Hao Louis received his B.Sc. (Hons) degree in Engineering Physics from The Hong Kong Polytechnic University in 2021. He later completed his Master's degree in Data Science at the University of Texas at Austin in 2023. He is currently pursuing his PhD in Applied Science at Universiti Teknologi PETRONAS (UTP), Malaysia. His research centres on data science and machine learning, with a particular focus on statistical modelling, dimensionality reduction, and interpretable learning systems. His current work explores novel training strategies for Parameterised Long Short-Term Memory (LSTM) networks, addressing temporal gradient imbalances in sequence generation and improving generalisation across sparsely sampled parameter spaces. His research spans techniques such as CART-based models, principal component analysis (PCA), clustering algorithms, and artificial neural networks (ANN), with an emphasis on developing clear, robust, and pedagogically sound analytical frameworks in statistical learning theory and model interpretability.



This is an open-access article distributed under the terms of the [Creative Commons Attribution-NonCommercial 4.0 International](https://creativecommons.org/licenses/by-nc/4.0/) (CC BY-NC 4.0), which allows reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator.