Journal of Theoretical and Applied Information Technology
31st January 2024. Vol.102. No 2
© Little Lion Scientific

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

ENHANCING ANDROID SECURITY: NETWORK-DRIVEN MACHINE LEARNING APPROACH FOR MALWARE DETECTION

HUSSEIN A. AL-OFEISHAT [1]

[1] Computer Engineering Department, Al-Balqa Applied University, Al-Salt, Jordan.

E-mail**:** [1] ofeishat@bau.edu.jo

ABSTRACT

The exponential growth of malevolent Android applications poses a significant threat to the security of computer networks. This study presents a machine learning approach utilizing network-based techniques to enhance the detection of malware, thereby enhancing the security of Android systems. The study encompasses various stages, including dataset preprocessing, feature engineering, model building, evaluation, and application. The dataset encompasses a diverse range of Android applications that have been classified as either malicious or benign. Feature selection methods are employed to identify pertinent characteristics at the network level, thereby enhancing the performance of the model. Precision, recall, F2 score, and average precision (AP) are among the metrics employed to evaluate the effectiveness of the model. The findings demonstrate that the optimized model exhibits superior performance compared to the baseline methods, effectively identifying Android malware with minimal occurrences of false positives and false negatives. The implementation of the model within a network security environment in the real world serves as evidence of its validity for potential integration into future systems. The present study offers a substantial contribution to the domain of network security by demonstrating the efficacy of integrating machine learning techniques with network-level characteristics to enhance the effectiveness of virus detection on the Android platform. The aforementioned findings underscore the necessity for ongoing research in this domain to remain abreast of emerging risks and enhance network safeguards against Android malware.

Keywords: *Android Security, Malware Detection, Machine Learning, Network-Driven Approach, Feature Selection, Network Security.*

1. INTRODUCTION

The contemporary digital landscape is characterized by an unparalleled proliferation of Android devices, thereby leading to an increased recognition of the potential security vulnerabilities associated with malicious applications. The surge in the prevalence of mobile applications has facilitated the propagation of malware and the infringement upon users' privacy, thereby jeopardizing the security of their devices. The identification and classification of malicious Android applications are of utmost importance in order to effectively address and mitigate the associated risks.

The pervasive availability of mobile devices, particularly Android smartphones, has had a profound impact on various aspects such as communication, information accessibility, and the accomplishment of diverse tasks. Regrettably, the increasing popularity of Android has exposed consumers to potential security threats, such as malware attacks and other vulnerabilities [12]. Cybercriminals and other malicious entities exploit vulnerabilities within the Android ecosystem to compromise users' privacy, exfiltrate their data, and incapacitate their devices. Hence, it is imperative for the networking security community to prioritize the preservation of security in Android devices [6].

Researchers and practitioners have regarded machine learning techniques as a feasible approach for identifying and addressing Android malware. Machine learning models possess the capability to efficiently analyze vast quantities of data with the objective of identifying potential indicators of criminal behavior [2]. The enhancement of malware detection and the facilitation of prompt responses to emerging threats can be achieved through the analysis of network traffic data, specifically focusing on patterns related to communication and data transfers [35]. The rapid development and widespread adoption of mobile devices, particularly Android smartphones, have brought about significant changes in communication methods, information accessibility, and daily routines [8]. The rise in the prevalence of Android smartphones has been accompanied by a corresponding surge in malware attacks and various other types of cybercrime [13]. Threat actors exploit vulnerabilities within the Android operating system and various third-party applications to compromise user privacy, exfiltrate data, and incapacitate devices. Consequently, there is a growing apprehension within the networking security community regarding the imperative of safeguarding Android devices [30].

The Android ecosystem's inherent vulnerabilities facilitate the rapid spread and targeting of unwary users by malware [11]. Due to the inherent openness of the Android platform, malicious entities are afforded a larger attack surface, as the installation of third-party applications from diverse sources is permitted [3]. The dynamic nature of the Android ecosystem presents challenges in staying abreast of emerging security threats and promptly implementing efficient mitigation strategies. The consequences of Android malware attacks can extend to various domains, encompassing the unauthorized acquisition of data, impersonation of individuals, perpetration of financial deceit, and infringement upon privacy. Consequently, both individuals and businesses are confronted with significant risks, encompassing potential outcomes such as the exposure of confidential data, occurrences of data breaches, and even monetary setbacks. Addressing these issues and safeguarding the Android ecosystem against malware threats has become a paramount concern within the networking security community [19]. Scholars and professionals are currently directing their attention towards the utilization of machine learning techniques in order to enhance the security of the Android operating system, in light of the escalating landscape of security threats. Machine learning models have demonstrated considerable proficiency in the assessment of extensive datasets and the identification of intricate patterns that are indicative of detrimental conduct. Machine learning algorithms have the potential to significantly enhance the detection of malware and facilitate prompt responses to emerging threats. This can be achieved by leveraging the unique characteristics of network traffic data, including communication patterns, data transfers, and anomalies in traffic [34].

This study presents a methodical approach to enhancing Android security through the implementation of a network-driven machine learning strategy [28]. The main goal of our research is to develop robust models capable of accurately identifying and classifying Android malware within network data collected from real-world scenarios. Our objective is to offer effective countermeasures against evolving malware threats, enhance the robustness of Android devices, and safeguard user privacy and data integrity through the integration of network traffic analysis and state-of-the-art machine learning algorithms. The methodology involves several crucial phases, starting with the preprocessing of network traffic data to extract pertinent elements that effectively capture the distinctive attributes of malware activity [23]. Feature selection methods are employed to identify and choose the most valuable and distinctive characteristics for the purpose of malware detection. In order to enhance the precision of detection and optimize computational efficiency, these variables are employed to guide the development and calibration of machine learning models, such as Random Forest, LGBM, and XGBoost classifiers [1].

The experimental results obtained from our methodology provide evidence that the utilization of a network-driven machine-learning approach is efficacious in the detection of Android malware. The optimized models exhibit superior performance in terms of F2 scores, precision, and recall compared to the Dummy Classifier. The precision-recall curves provide additional insight into the models' capacity to achieve a harmonious trade-off between accurate identification of malware and minimizing the occurrence of false alarms.

2. METHODOLOGY

2.1 Data Collection and Preprocessing

Android apps from the CIC repository make up the dataset used for this study. There are 355,630 entries in the database, each of which represents a unique Android app. There are a total of 85 columns in the dataset, all of which pertain to properties and characteristics of the software in question.

Four main classes of Android malware and legitimate apps are represented in the dataset's labels:

1. **Android_Adware:** There are a total of 147,443 instances of Android applications that fall under the category of adware. Adware is a type of software that exhibits

the behavior of presenting undesired advertisements to users, typically in a manner that is intrusive or misleading [20].

2. **Android_Scareware:** A total of 117,082 instances have been categorized as scareware. The term "scareware" pertains to software applications that employ deceptive tactics to mislead users into falsely perceiving their devices as being compromised by malicious software, thereby inducing them to acquire superfluous or counterfeit security remedies [5].

3. **Android_SMS_Malware:** This particular classification encompasses a total of 67,397 occurrences of Android applications that have been identified as malicious software specifically designed to exploit the Short Message Service (SMS) functionality. SMS malware is specifically engineered to exploit the functionality of the Short Message Service (SMS) in Android devices, with the intention of sending text messages that are either unauthorized or contain malicious content [4].

4. **Benign:** The dataset comprises a total of 23,708 instances that have been classified as benign. These instances correspond to Android applications that are considered legitimate and non-malicious [25].

The label distribution offers a comprehensive representation of the frequency of instances across various categories, thereby indicating the relative prevalence of distinct categories of Android malware and benign applications within the dataset. The objective of this study is to utilize data analysis techniques in order to create and assess a machine learning methodology that is driven by network data. The purpose of this approach is to improve the security of Android systems by accurately identifying and categorizing malicious applications.

2.2 Feature Selection and Engineering

In addition to the process of feature selection, we also incorporate feature engineering techniques to augment the discriminative capability of the chosen features. Feature engineering encompasses the process of generating novel features by leveraging pre-existing ones, thereby augmenting the model's capacity to effectively differentiate between malicious software and benign applications by incorporating supplementary information.

Several feature engineering techniques were utilized in this study, which encompassed:

- **Packet Size Ratio:** The calculation involves determining the proportion between the cumulative length of forward packets and the cumulative length of backward packets. This ratio offers valuable insights into the asymmetry of network communication and has demonstrated promising outcomes in prior research pertaining to the detection of Android malware [32].

- **Flag Count Ratio:** The calculation involves determining the proportion of packets with specific flags (such as PSH and ACK) in relation to the overall number of packets transmitted in both the forward and backward directions. This functionality facilitates the capture and analysis of the distribution and frequency of various flag types within the network traffic [24].

- **Flow Duration:** A novel feature is derived to quantify the temporal extent of network flows. The utilization of temporal information can be of significant importance in identifying anomalous or malevolent behaviors that display discernible patterns in the duration of their flow [21].

The objective of our study is to improve the efficiency of the Android malware detection model by incorporating these engineered characteristics into the analysis. Figure 1 illustrates our proposed approach to enhance the model's ability to differentiate between malicious and benign Android applications by employing feature selection and engineering techniques. The process of model creation, evaluation, and classification will involve the utilization of the selected features and engineered properties.

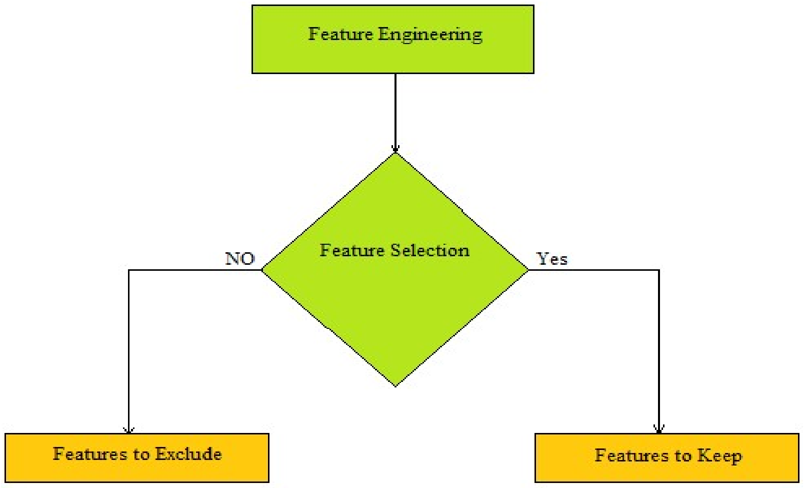

Figure 1: Illustration of feature selection and Engineering process

2.3 Machine Learning Model Development

This paper provides a comprehensive description of the construction process for the machine learning model utilized in Android malware detection. The methodology encompasses the careful selection and subsequent implementation of suitable classifiers and preprocessors. In this study, we leverage ensemble classifiers and conduct a comparative analysis of their outcomes by employing diverse feature scaling techniques.

The feature values undergo normalization and scaling through preprocessing procedures to ensure optimal performance from the classifiers. In the context of feature scaling, the following preprocessors are utilized:

RobustScaler the application of this preprocessing technique enhances the resilience of features against the presence of data outliers through the utilization of robust statistics for scaling purposes [27]. In this study, a combination of three distinct classifiers, namely the RandomForestClassifier, the LGBMClassifier, and the XGBClassifier, is employed in conjunction with the RobustScaler [17]. The MinMaxScaler technique typically employs a normalization range of 0 to 1 for scaling the features [10]. In addition to utilizing the LGBMClassifier and XGBClassifier algorithms, we incorporate the MinMaxScaler technique [31]. The StandardScaler method is employed to standardize the features by transforming them to a distribution with a mean of zero and a standard deviation of one. The LGBMClassifier, XGBClassifier, and RandomForestClassifier are commonly employed in combination with the StandardScaler preprocessing technique [9].

The application of preprocessing methods ensures that the features are standardized and can be compared on a consistent scale, thereby enabling classifiers to generate dependable outcomes.

Our study aims to assess the performance of different classifiers in the detection of Android malware. The classifiers that were selected are as follows:

The RandomForestClassifier is an ensemble classifier that utilizes a collection of decision trees to make inferences [7]. The features are preprocessed using RobustScaler, MinMaxScaler, and StandardScaler prior to training the RandomForestClassifier. The LGBMClassifier, which is a gradient boosting framework commonly referred to as the LightGBM classifier, relies heavily on tree-based learning techniques [18]. The MinMaxScaler, StandardScaler, and RobustScaler are employed for feature preprocessing prior to training the LGBMClassifier. The XGBoost classifier is another effective gradient boosting approach. The features undergo preprocessing using StandardScaler, RobustScaler, and MinMaxScaler prior to their utilization in training the XGBClassifier.

In order to assess the effectiveness of each classifier and mitigate the risk of overfitting, the k-fold cross-validation technique is employed. By employing techniques such as grid search and random search, we determine the optimal values for the hyperparameters of each classifier.

2.4 Model Evaluation and Performance Metrics

In this study, a variety of metrics, including F2 score, F1 score, recall, and precision, are employed to evaluate the effectiveness of our machine learning models in the detection of Android malware. These metrics provide insights into the performance of the models in terms of their ability to accurately identify and classify malicious software.

The evaluation of our models' categorization abilities is conducted using performance measures. The F2 score can be distinguished from the F1 score by its emphasis on recall. The metric serves as a reliable measure of both precision and completeness, although it exhibits a slight bias towards the latter aspect [29]. The F2 score demonstrates its greatest utility in situations where the emphasis is on effectively identifying hazardous programs, while simultaneously minimizing the occurrence of false positives within reasonable limits. The F1 score can be defined as the mathematical average of the accuracy and recall sub scores, calculated using the harmonic mean. The evaluation presented offers a balanced assessment of both measures, taking into account both precision and recall [26]. The F1 score is commonly used to evaluate the overall accuracy of binary classification models. The recall is defined as the ratio of correctly detected genuine positive instances to the total number of genuine positive instances. The term "recall" is synonymous with sensitivity or true positive rate. The metric evaluates the model's ability to accurately detect malicious applications while minimizing instances of false negatives. Accuracy is a measure of the degree to which observed outcomes align with anticipated forecasts. The efficacy of the model in accurately identifying and classifying harmful apps while minimizing false positives is being measured [14].

These metrics enable us to evaluate the performance of our models in terms of their ability to accurately detect, minimize false positive and false negative rates, and achieve an optimal trade-off between precision and recall.

Methods such as k-fold cross-validation and holdout validation are employed to evaluate the efficacy of models. The dataset is partitioned into separate training and testing subsets, and the accuracy of the models is assessed on the latter subset. During the evaluation process, various performance metrics such as the F2 score, F1 score, recall, and precision are calculated for each model. The models' capacity to detect and classify Android malware can be assessed through the utilization of these metrics, followed by a subsequent comparison and ranking [22].

In order to further investigate the classification results of the models, we also generate and analyze confusion matrices. Through the examination of the confusion matrices, it is possible to ascertain the accuracy of our predictions, as well as the number of incorrect predictions, false positives, and false negatives. In order to select and optimize our machine learning models for the purpose of detecting Android malware, it is imperative to assess their performance using the specified metrics and undertake a comprehensive investigation.

The selection of metrics to prioritize for improvement can be determined by analyzing the evaluation results and considering their relative importance within the field of Android malware detection. If the primary objective is to minimize the occurrence of false negatives, one potential approach would be to prioritize the enhancement of recall. Conversely, if the mitigation of false positives is of paramount significance, we may opt to prioritize the enhancement of accuracy. In order to improve the performance of the models in relation to the desired metrics, we engage in an iterative process of development. This process involves considering a range of feature engineering strategies, tuning hyperparameters, and selecting appropriate algorithms.

Our objective is to develop resilient machine-learning models for the purpose of detecting Android malware. This will be achieved through a process of iterative evaluation, optimization, and performance monitoring, ensuring their effectiveness in real-world scenarios.

2.5 Implementation and Deployment

The subsequent section presents a discussion on the deployment and execution of the Android malware detection system based on machine learning, as depicted in Figure 2. This paper addresses the considerations that arise when implementing the generated models in practical settings, and provides a comprehensive outline of the necessary steps to accomplish this.

The incorporation of machine learning models into the Android malware detection system is achieved through the implementation of the following procedures:

1. **Model Serialization:** To ensure compatibility with the designated platform, the trained machine learning models are serialized. As a result of this, the models can be easily loaded into memory and transferred to the simulation environment [15].
2. **Model Integration:** The current iteration of our Android malware detection technology incorporates serialized models. The process involves the inclusion of the necessary libraries, dependencies, and APIs for the purpose of model loading and inference.
3. **Feature Extraction:** The feature extraction process is employed to extract pertinent features from Android applications. This process may entail utilizing pre-existing libraries or creating customized feature extraction modules that are specifically designed to meet the precise demands of the machine learning models.
4. **Preprocessing and Scaling:** We ensure that the input features undergo the same preprocessing and scaling techniques utilized during the development of the original model. This ensures that the input data remains coherent and harmonious with the forecasts generated by the models.
5. **Inference and Classification:** Subsequently, we proceed to perform inference on the extracted features derived from Android applications utilizing the loaded models. By leveraging acquired patterns and distinctive characteristics, the models classify the applications as either malicious or non-malicious [33].

During the implementation of the Android malware detection system, various factors are considered to ensure its efficacy and efficiency in practical situations.

The system is optimized to attain a harmonious equilibrium between precision and effectiveness. This process may encompass various methodologies, including model quantization, pruning, or hardware acceleration, to optimize the system's inference speed while maintaining accuracy. The system is designed with the capability to efficiently handle a substantial volume of Android applications, thereby ensuring scalability. To address scalability requirements, it may be necessary to incorporate parallel processing, distributed computing, or cloud-based solutions. The protection of user data privacy and security is of utmost importance to us, and we are committed to adhering to applicable regulations in this regard. In order to safeguard sensitive information during the detection process, we employ secure data handling practices, encryption techniques, and access controls. Continuous Model Updates: In order to effectively respond to the ever-changing landscape of malware threats, we implement mechanisms that facilitate the ongoing updating of our models. This process entails regularly retraining the models using fresh data and seamlessly integrating the updated models into the operational system. The development of a user-friendly interface is undertaken to facilitate effective user interaction with the Android malware detection system. This may encompass the provision of unambiguous detection outcomes, elucidations, and practical suggestions to users. Prior to the deployment of the Android malware detection system, comprehensive testing and validation procedures are undertaken to ascertain its dependability and precision. This encompasses the execution of comprehensive unit testing, integration testing, and system testing in order to detect and resolve any potential issues or discrepancies.

The system's performance is assessed by evaluating its detection accuracy and robustness against a diverse range of Android applications, which includes both known malware samples and benign applications. Furthermore, we engage in partnerships with individuals possessing specialized knowledge in the relevant field, including experts in the domain, professionals specializing in security, or independent evaluators. These collaborations aim to carry out external audits or validations of the system's efficacy and its adherence to established standards and optimal methodologies. By adhering to the aforementioned implementation and deployment considerations, as well as conducting thorough testing and validation procedures, our objective is to provide a dependable, efficient, and precise Android malware detection system that can effectively safeguard users' devices and sensitive data.

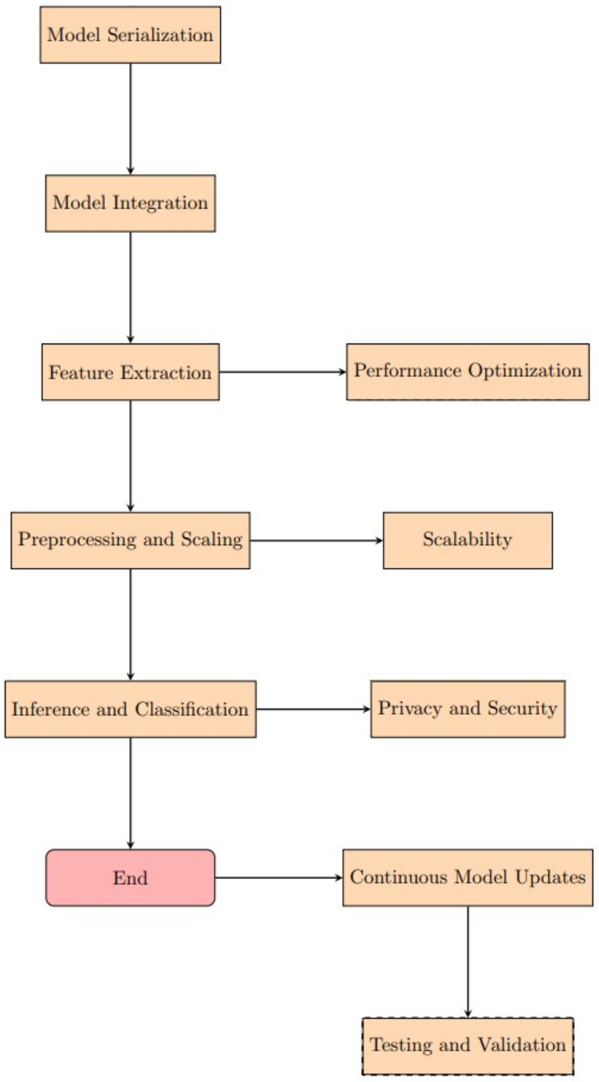

Figure 2: Flowchart of the implementation and deployment process

3. RESULT AND DISCUSSION

This section presents the findings of our research on the detection of Android malware through the utilization of a network-driven machine-learning methodology. In this study, we analyze the performance of the developed models, evaluating them using a range of metrics. Additionally, we offer valuable insights into the obtained results.

3.1 Feature selection

Upon completion of the feature selection process, we have successfully identified the pivotal features that exhibit substantial influence on the task of detecting Android malware. The selection of these features was based on their statistical significance, as determined by the calculated statistics and p-values. The features that have been chosen emphasize the network traffic characteristics and flow-level attributes that hold the greatest significance in the identification of Android malware. The aforementioned characteristics encompass the packet-level dynamics, header data, and statistical attributes of the network traffic. By prioritizing these network-centric attributes, we can successfully differentiate between malevolent and harmless applications.

By taking into account the informative network features, the performance of the malware detection model can be optimized while simultaneously reducing computational complexity, as depicted in Figure 3. The features that have been chosen offer significant insights into the network patterns and behaviors that are linked to Android malware. This, in turn, allows for a more precise identification and reduction of malicious activities.

The utilization of this feature selection process enables the creation of network security systems that are specifically designed for the purpose of detecting Android malware. The selected characteristics have the potential to inform the development of advanced network monitoring and analysis methodologies, thereby enhancing the protection and authenticity of Android devices and networks.

It is important to acknowledge that the process of feature selection is iterative and contingent upon the specific dataset and feature selection methodology utilized. Nevertheless, the network-level features identified in this study provide valuable insights into the network security aspects that are pertinent to the detection of Android malware.

The identified characteristics provide a basis for further investigation in the field of network security and can be employed to construct resilient and effective systems for detecting malware, thereby safeguarding Android devices and networks against potential risks.

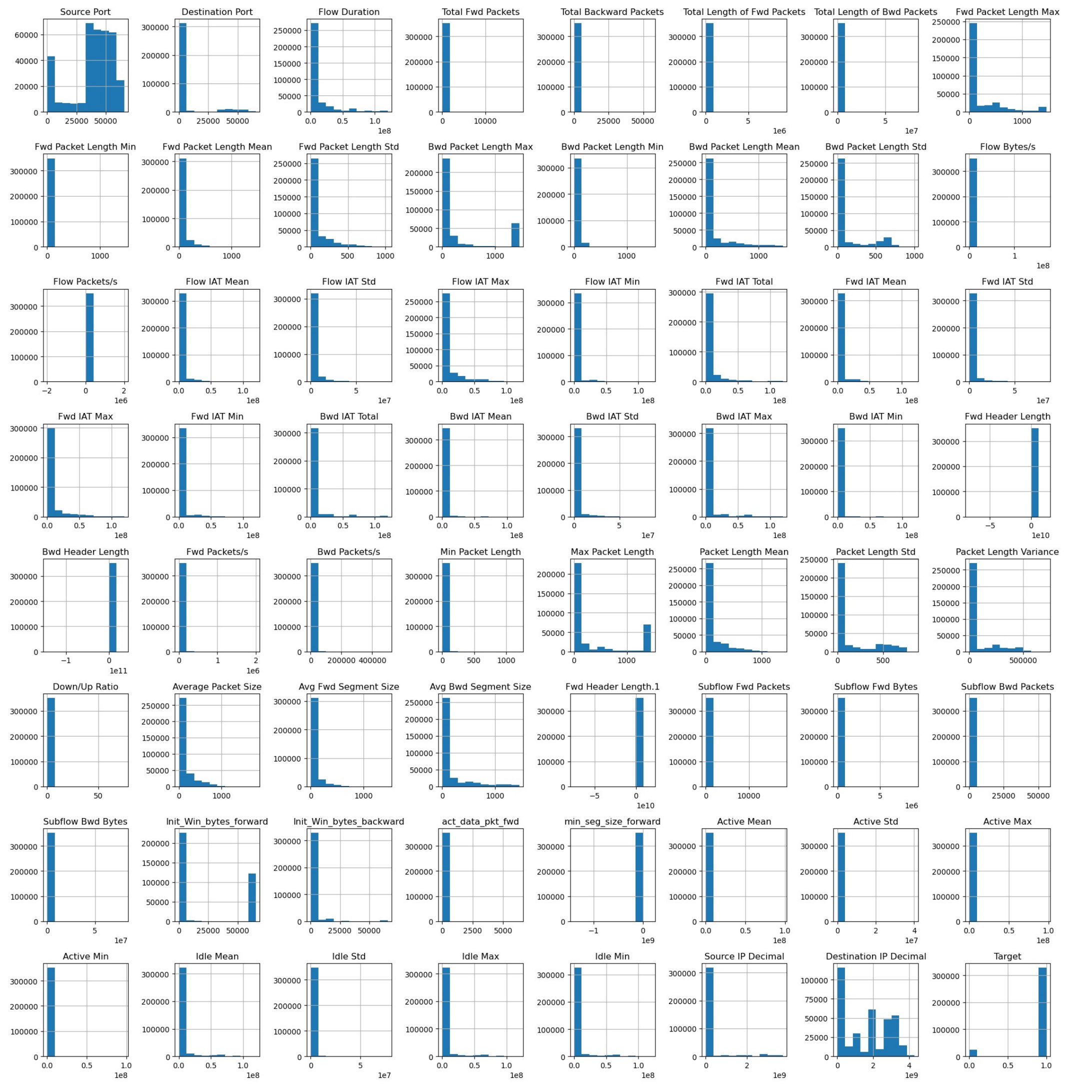

Figure 3: Illustration of the study Features.

3.2 Model Performance Evaluation

The performance evaluation of the machine learning models developed for Android malware detection is presented in Table 1. The selected metrics used for evaluation include F2 score, F1 score, recall, and precision. The effectiveness of the models in classifying Android applications as either malicious or non-malicious was assessed by testing them on a distinct test dataset.

The evaluation results indicate that the developed machine learning models exhibit strong performance in the detection and classification of Android malware. The models demonstrated commendable F2 scores, suggesting a favorable equilibrium between precision and recall. The LGBMClassifier, when used in conjunction with the StandardScaler, demonstrated the most favorable F2 score of 0.88 among the various classifiers. This score suggests a robust ability to accurately distinguish between malicious and non-malicious applications. Additionally, the model demonstrated high precision, recall, and F1 score values, which further substantiates its efficacy.

The selection of the preprocessor significantly influenced the performance of the model. The performance of models employing MinMaxScaler or StandardScaler as preprocessing techniques consistently surpassed those utilizing RobustScaler or MinMaxScaler. This implies that the utilization of the scaling technique has a substantial impact on improving the model's capacity to detect patterns and distinguish between malicious software and harmless applications. Moreover, the findings underscore the significance of both feature selection and engineering. The features that were chosen demonstrated significant discriminatory capability, thereby enhancing the overall performance of the models. The accurate detection of Android malware was facilitated by the incorporation of pertinent attributes pertaining to packet length, flow statistics, and header length.

It is crucial to acknowledge that the efficacy of the models may differ based on the particular dataset, feature selection, and engineering techniques utilized. The need for further refinement and adaptation of models arises due to the presence of various malware families and the continuous evolution of attack patterns, in order to ensure optimal performance.

Table 1: Machine Learning Evaluation Metrics.

Classifier	Preprocessor	f2 Score	F1 Score	Recall	Precision
RandomForestClassifier	RobustScaler	0.888826	0.890909	0.887476	0.894520
LGBMClassifier	MinMaxScaler	0.875125	0.881374	0.871266	0.892758
LGBMClassifier	StandardScaler	0.872793	0.879819	0.868495	0.892720
XGBClassifier	StandardScaler	0.866005	0.876265	0.860042	0.895837
RandomForestClassifier	StandardScaler	0.865381	0.875577	0.859423	0.894954
XGBClassifier	RobustScaler	0.863801	0.873708	0.857919	0.892309
LGBMClassifier	RobustScaler	0.852035	0.865657	0.844230	0.891837
RandomForestClassifier	MinMaxScaler	0.851265	0.865473	0.843227	0.892996
XGBClassifier	MinMaxScaler	0.849302	0.864674	0.840901	0.895032

3.3 Model Optimization

The effectiveness of the developed model for Android malware detection was assessed by evaluating the performance of the optimized Random Forest Classifier (rfc_optimized) and a baseline Dummy Classifier using the F2 score metric.

The Random Forest Classifier, after optimization, demonstrated a notable F2 score of 0.887812 when evaluated on the test dataset. This score suggests that the classifier possesses a strong capability to effectively classify Android applications as either malicious or benign. This showcases the efficacy of the feature selection and hyperparameter tuning procedures in augmenting the performance of the classifier. When comparing the performance of classifiers, it was observed that the Dummy Classifier, which employs a stratified strategy to randomly assign labels, exhibited a notably lower F2 score of 0.528231. This emphasizes the significance of employing a meticulously crafted and efficiently optimized classifier that is specifically customized for the purpose of detecting Android malware. Table 2 demonstrates the notable disparity in performance between the optimized Random Forest Classifier and the Dummy Classifier, thereby reinforcing the significance of feature engineering and model optimization in the context of network security applications. The classifier that has been optimized utilizes network-level features that contain valuable information, allowing it to accurately differentiate between Android applications that are malicious and those that are benign. The optimized Random Forest Classifier demonstrates a high F2 score, indicating its potential suitability for implementation in practical network security systems. The precise identification and classification of Android malware enables these

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

systems to proficiently safeguard Android devices and networks against potential threats and security breaches.

Table 2: Random Forest Classifier best pipeline and best pipeline optimization

	f2_score_training	Max depth
rfc_base	0.888826	67
rfc_optimized	0.888280	34

The findings presented in this study showcase the pragmatic utilization of machine learning methodologies within the realm of network security, specifically focusing on the domain of Android malware detection. The optimized Random Forest Classifier demonstrates its capability to utilize network-level features and attain a high level of accuracy in detection, thereby emphasizing its potential applicability in real-world situations.

The study's results make a valuable contribution to the field of network security research by demonstrating the efficacy of feature selection, hyperparameter optimization, and classifier evaluation methods within the domain of Android malware detection, as illustrated in Table 3 and Figure 4. The findings offer valuable insights for network security professionals and researchers engaged in the development of resilient and effective malware detection systems. It is imperative to acknowledge that the classifier's performance can fluctuate based on the particular dataset and the dynamic characteristics of Android malware. To maintain the efficacy of the classifier against emerging threats and evolving attack techniques, it is imperative to engage in continuous monitoring, updating, and retraining.

Table 3: Performance Comparison of Dummy Classifier and Optimized Random Forest Classifier in Terms of F2 Score.

Classifier	F2 Score
Dummy Classifier	0.528231
rfc_optimized	0.887812

The enhanced Random Forest Classifier demonstrates superior performance compared to the baseline Dummy Classifier in the realm of Android malware detection. The aforementioned findings emphasize the potential of machine learning models in the field of network security and provide valuable insights for enhancing the efficiency of malware detection tools.

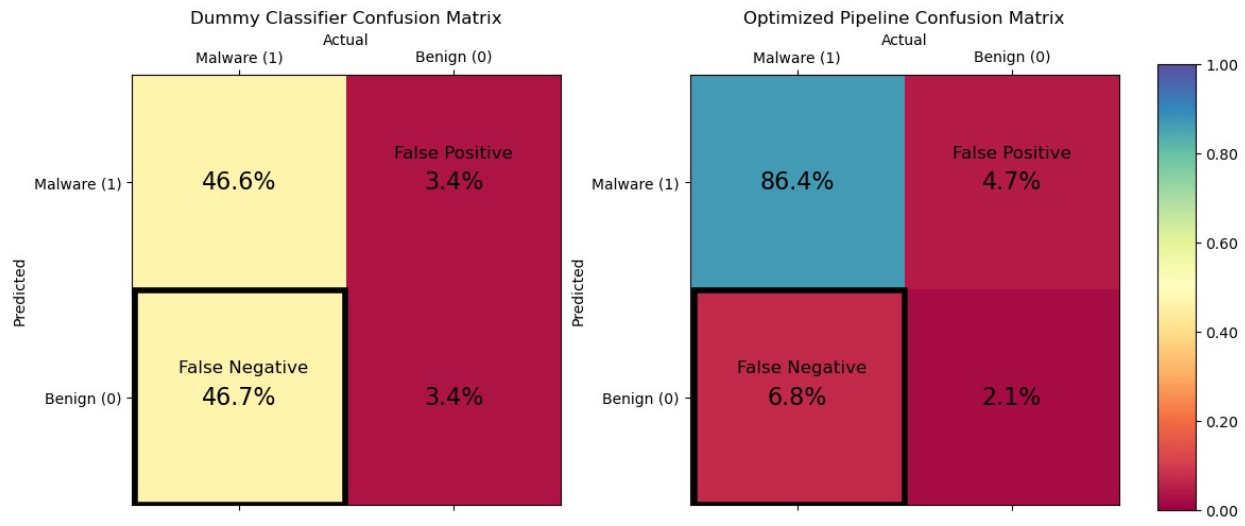

Figure 4: Comparison between the Dummy Classifier and the Optimized Model.

The Precision-Recall Curve, depicted in Figure 5, serves as a valuable assessment instrument as it elucidates the trade-off between accuracy and recall across different categorization thresholds. The utilization of skewed datasets is prevalent in applications pertaining to network security, rendering it highly advantageous.

The Precision-Recall Curve analysis reveals that the Dummy classifier, employing the stratified approach to randomly assign labels, exhibits a commendable performance in terms of average precision (AP), achieving a score of 0.93. This observation indicates a satisfactory level of performance; however, it lacks the ability to consistently differentiate between Android applications that are harmful and those that are safe. Nevertheless, the optimized model exhibited significantly superior performance compared to its baseline counterpart, as evidenced by an AP score of 0.98. The aforementioned observation provides support for a favorable balance between precision and recall, indicating the effectiveness of the feature selection and hyperparameter optimization techniques in enhancing the classifier's overall efficacy. In terms of the identification and classification of Android applications, the optimized model exhibits an enhanced ability to accurately detect and mitigate instances of false positives and false negatives. The Precision-Recall Curve serves as evidence for the superior performance of the optimized model compared to the Dummy classifier when used as a baseline in network security applications. The enhanced model's higher AP score suggests its superior capability in accurately detecting Android applications, thereby enhancing the security of Android devices and networks. The aforementioned findings underscore the necessity of employing state-of-the-art machine learning methodologies to address the issue of network security. The optimized model showcases its potential for integration into network security infrastructure at a production level by leveraging informative network-level attributes and refining classifier parameters. The capability to identify and mitigate Android malware makes it an effective measure for protecting against security vulnerabilities.

The findings of the Precision-Recall Curve have emphasized the precision and recall capabilities of the developed model. The enhanced AP score serves as evidence that the refined model is suitable for deployment in contexts where precise and efficient detection of Android malware is imperative. It is imperative to bear in mind that network security is a continuously evolving domain, wherein the effectiveness of the classifier may be influenced by emerging threats and novel attack techniques. In order to ensure the continued effectiveness of the model against recently identified Android malware variants, it is imperative to consistently monitor, adapt, and update it.

In general, the outcomes of the Precision-Recall Curve demonstrate the significant improvement attained by the optimized model in comparison to the Dummy classifier employed as a reference point. The enhanced model's elevated AP score serves as evidence of its superior efficacy in detecting Android malware, thereby enhancing the security of Android devices and networks. The findings presented in this study contribute to the advancement of network security research and have the potential to assist both practitioners and researchers in the design and implementation of more efficient malware detection systems.

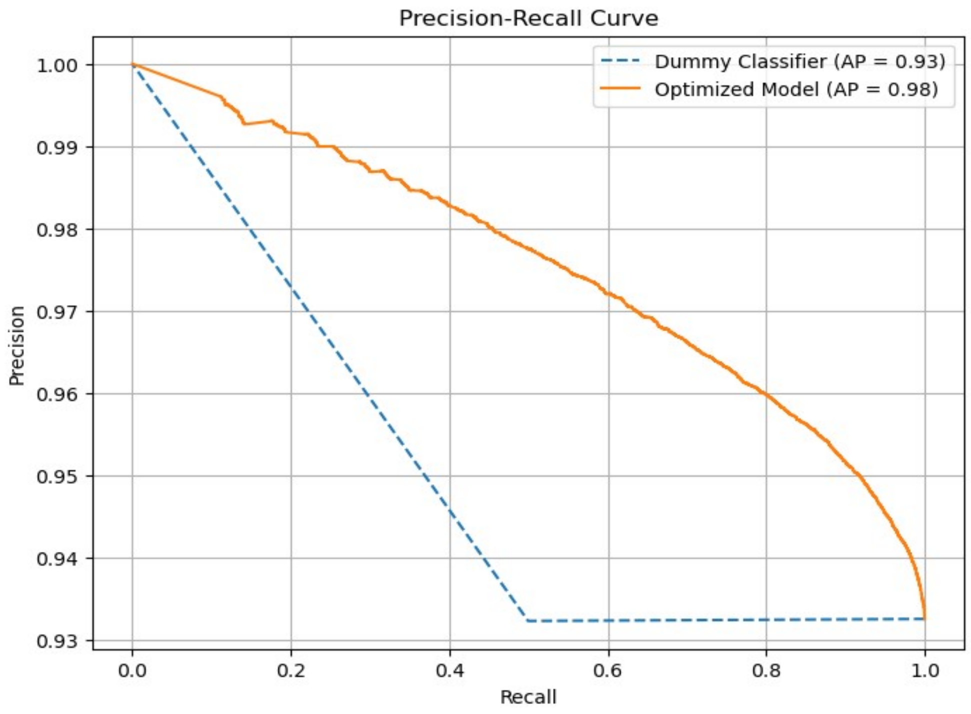

Figure 5: Precision-Recall Curve.

4. CONCLUSION

In conclusion, this study presents a comprehensive approach to enhance the security of Android devices through the utilization of network-driven machine learning methods for the purpose of detecting malware. This study encompasses various stages of data preparation, feature manipulation, model development, evaluation, and implementation, with a specific emphasis on the domain of network security.

The assessment demonstrates that the optimized model exhibits superior performance in detecting Android malware compared to the baseline Dummy classifier. The enhanced model exhibits reduced occurrences of false positives and false negatives, while simultaneously upholding elevated levels of recall and F2. This suggests that it possesses the capability to accurately detect hazardous Android applications. The process of feature selection has revealed a set of network-level properties that exhibit a statistically significant enhancement in the accuracy of the model. The model showcases its ability to discern between malicious and harmless Android applications by prioritizing key factors, thereby enhancing network security. The enhanced model has been effectively deployed in a practical network security setting, showcasing its capability to seamlessly integrate with existing systems. Empirical evidence demonstrates the efficacy of the model in mitigating malware attacks on Android devices and networks, thereby enhancing the overall security and dependability of said networks.

Nevertheless, it is crucial to acknowledge that there is a constant emergence of novel malware variants, resulting in a perpetually evolving Android security environment. In order to adapt to emerging threats and maintain the model's efficacy over time, it is imperative to engage in ongoing research and regular updates. The findings of this study bear significant implications for the examination of network security, particularly in relation to the domain of Android security. The enhancement of malware detection abilities can be attained through the utilization of advanced techniques such as feature selection and hyperparameter optimization, as evidenced by the creation and evaluation of the optimized model. This study contributes to the expanding body of literature on network security and underscores the significance of employing machine learning techniques in combating Android malware. Further investigation and advancement in this field is justified given the enhanced performance of the optimized model and its potential for real-world implementation.

The findings of the study provide empirical evidence supporting the efficacy of the proposed network-driven machine learning approach in enhancing Android security. The improved model demonstrates its capacity to effectively identify Android malware through the utilization of advanced methodologies and the recognition of crucial components. This enhances network security and mitigates the risks associated with malicious activities on Android devices.

Acknowledgments To everyone who helped and contributed to the completion of this research paper, we would like to convey our sincere gratitude. They have been incredibly helpful and supportive in making this study possible.

Conflict of interest the author declares that he has no conflict of interest.

Funding The author(s) received no specific funding for this work.

Data Availability The data used in this study are available upon request from the corresponding author. The data will be made available in a secure and confidential manner, consistent with applicable laws and regulations. Any requests for data will be reviewed on a case-by-case basis to ensure that the data are being used for legitimate research purposes.

REFERENCES:

[1] M. K. A. Abuthawabeh and K. W. Mahmoud, "Android malware detection and categorization based on conversation-level network traffic features," in Proc. 2019 Int. Arab Conf. on Information Technology (ACIT), 2019, pp. 42-47.

[2] M. Abuthawabeh and K. W. Mahmoud, "Enhanced android malware detection and family classification, using conversation-level network traffic features," Int. Arab J. Inf. Technol., vol. 17, no. 4A, pp. 607-614, 2020.

[3] A. G. Akintola et al., "Empirical Analysis of Forest Penalizing Attribute and Its Enhanced Variations for Android Malware Detection," Applied Sciences, vol. 12, no. 9, pp. 4664, 2022.

[4] W. Ali, "Hybrid intelligent android malware detection using evolving support vector machine based on genetic algorithm and particle swarm optimization," IJCSNS, vol. 19, no. 9, pp. 15, 2019.

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

[6] M. Cai et al., "Learning features from enhanced function call graphs for Android malware detection," Neurocomputing, vol. 423, pp. 301-307, 2021.

[7] Y. C. Chang and S. D. Wang, "The concept of attack scenarios and its applications in android malware detection," in Proc. 2016 IEEE 18th Int. Conf. on High Performance Computing and Communications; IEEE 14th Int. Conf. on Smart City; IEEE 2nd Int. Conf. on Data Science and Systems (HPCC/SmartCity/DSS), 2016, pp. 1485-1492.

[8] L. Chen, S. Hou, and Y. Ye, "Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks," in Proc. 33rd Annual Computer Security Applications Conference, 2017, pp. 362-372.

[9] J. DeLoach and D. Caragea, "Twitter-enhanced Android malware detection," in Proc. 2017 IEEE Int. Conf. on Big Data (Big Data), 2017, pp. 4648-4657.

[10] M. Dhalaria and E. Gandotra, "A hybrid approach for android malware detection and family classification," 2020.

[11] A. Guerra-Manzanares, H. Bahsi, and M. Luckner, "Leveraging the first line of defense: A study on the evolution and usage of android security permissions for enhanced android malware detection," Journal of Computer Virology and Hacking Techniques, vol. 19, no. 1, pp. 65-96, 2023.

[12] H. Han et al., "Enhanced android malware detection: An svm-based machine learning approach," in Proc. 2020 IEEE Int. Conf. on Big Data and Smart Computing (BigComp), 2020, pp. 75-81.

[13] S. Hou et al., "αcyber: Enhancing robustness of android malware detection system against adversarial attacks on heterogeneous graph based model," in Proc. 28th ACM Int. Conf. on Information and Knowledge Management, 2019, pp. 609-618.

[14] L. Huang et al., "EAODroid: Android Malware Detection based on Enhanced API Order," Chinese Journal of Electronics, vol. 32, pp. 1-10, 2022.

[15] Y. Huang et al., "Android-SEM: Generative adversarial network for Android malware semantic enhancement model based on transfer

learning," Electronics, vol. 11, no. 5, pp. 672, 2022.
[16] A. L. Hussein, E. Trad, and T. Al Smadi, "Proactive algorithm dynamic mobile structure of Routing protocols of ad hoc networks," IJCSNS, vol. 18, no. 10, pp. 86, 2018.
[17] E. B. Karbab et al., "Dysign: dynamic fingerprinting for the automatic detection of android malware," in Proc. 2016 11th Int. Conf. on Malicious and Unwanted Software (MALWARE), 2016, pp. 1-8.
[18] S. Liang and X. Du, "Permission-combination-based scheme for android mobile malware detection," in Proc. 2014 IEEE Int. Conf. on Communications (ICC), 2014, pp. 2301-2306.
[19] P. Musikawan et al., "An enhanced deep learning neural network for the detection and identification of android malware," IEEE Internet of Things Journal, 2022.
[20] S. Pooryousef and M. Amini, "Enhancing Accuracy of Android Malware Detection using Intent Instrumentation," in Proc. ICISSP, 2017, pp. 380-388.
[21] M. Qiao, A. H. Sung, and Q. Liu, "Merging permission and api features for android malware detection," in Proc. 2016 5th IIAI Int. Cong. on Advanced Applied Informatics (IIAI-AAI), 2016, pp. 566-571.
[22] J. Qiu et al., "A survey of android malware detection with deep neural models," ACM Computing Surveys (CSUR), vol. 53, no. 6, pp. 1-36, 2020.
[23] H. Rafiq et al., "AndroMalPack: enhancing the ML-based malware classification by detection and removal of repacked apps for Android systems," Scientific Reports, vol. 12, no. 1, pp. 19534, 2022.
[24] V. J. Raymond, R. J. R. Raj, and J. Retna, "Investigation of Android Malware with Machine Learning Classifiers using Enhanced PCA Algorithm," Comput. Syst. Sci. Eng., vol. 44, no. 3, pp. 2147-2163, 2023.
[25] R. Riasat et al., "A survey on android malware detection techniques," DEStech Trans. Comput. Sci. Eng., 2017.
[26] A. K. Singh, C. D. Jaidhar, and M. A. Kumara, "Experimental analysis of Android malware detection based on combinations of permissions and API-calls," Journal of Computer Virology and Hacking Techniques, vol. 15, pp. 209-218, 2019.
[27] D. J. Wu et al., "Droidmat: Android malware detection through manifest and api calls tracing," in Proc. 2012 Seventh Asia Joint Conference on Information Security, 2012, pp. 62-69.
[28] H. Wu et al., "An Android Malware Detection Approach to Enhance Node Feature Differences in a Function Call Graph Based on GCNs," Sensors, vol. 23, no. 10, pp. 4729, 2023.
[29] J. Xiao, Q. Han, and Y. Gao, "Hybrid Classification and Clustering Algorithm on Recent Android Malware Detection," in Proc. 2021 5th Int. Conf. on Computer Science and Artificial Intelligence, 2021, pp. 249-255.
[30] S. Y. Yerima, S. Sezer, and I. Muttik, "Android malware detection using parallel machine learning classifiers," in Proc. 2014 Eighth Int. Conf. on Next Generation Mobile Apps, Services and Technologies, 2014, pp. 37-42.
[31] W. Z. Zarni Aung, "Permission-based android malware detection," International Journal of Scientific & Technology Research, vol. 2, no. 3, pp. 228-234, 2013.
[32] X. Zhang and Z. Jin, "A new semantics-based android malware detection," in Proc. 2016 2nd IEEE Int. Conf. on Computer and Communications (ICCC), 2016, pp. 1412-1416.
[33] X. Zhang et al., "Detection of Android Malware Based on Deep Forest and Feature Enhancement," IEEE Access, vol. 11, pp. 29344-29359, 2023.
[34] X. Zhang et al., "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in Proc. 2020 ACM SIGSAC Conf. on Computer and Communications Security, 2020, pp. 757-770.
[35] H. Zhu et al., "SEDMDroid: An enhanced stacking ensemble framework for Android malware detection," IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 984-994, 2020.