Volume: 13 Issue: III Month of publication: March 2025

DOI: https://doi.org/10.22214/ijraset.2025.67661

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue III Mar 2025- Available at www.ijraset.com

Enhanced Ransomware Detection on Android Devices Using Traffic Analysis and Advanced Feature Selection Techniques

P. Sowjanya[1], Somu Anusha[2], Moningi Satish Kumar[3], D.Vinayak Aditya [4], U. Praveen Kumar [5]
Department of Cyber Security, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.531162

Abstract: Android phones, in particular, are highly vulnerable to attacks launched by ransomware, which is widely known to be one of the most common and devastating forms of cyber-attacks that we encounter in the current digital environment. In spite of the application of various techniques for the purpose of identifying such malicious attacks, most of these techniques are suffering from serious limitations, such as extremely low detection rates as well as the limitation of increased computational cost. With the purpose of addressing this hot topic, this paper proposes a new technique for the identification of ransomware attacks launched on Android phones with the assistance of network traffic analysis as a main approach. The proposed technique is developed with the purpose of enhancing the performance and accuracy of machine learning (ML) models, with the assistance of sophisticated techniques such as feature reduction and feature selection in order to achieve its goals.
Index Terms: Feature selection, malware, machine learning, particle swarm optimization, ransomware.

I. INTRODUCTION

The emergence of ransomware attacks has had a profound effect on cybersecurity, especially on mobile platforms like android devices. android, one of the most popular operating systems in the world, is now more frequently targeted by malicious software, including ransomware. ransomware, a type of malicious software that encrypts the files of the victim or locks the device until a ransom is paid, is a serious threat to individual users as well as businesses. androids are especially at risk because of the open-source status of the OS and the enormous variety of applications that are accessible through third-party markets, some of which are likely to be compromised.

Traditional ransomware detection approaches normally rely on signature-based methods or static analysis of the device behavior. such strategies are, however, ineffective for detecting new or altered ransomware versions, as they employ evasive tactics with high levels of sophistication. moreover, signature-based methods necessitate constant updating of their databases, which is not only resource-exhaustive but also might lack real-time capability.

with these constraints, there has been a move towards more active detection techniques that are capable of detecting ransomware through network traffic patterns and other behavioral characteristics. monitoring the traffic emanating

from an android device as it operates provides a promising technique for ransomware detection. such traffic patterns can indicate unusual activities, like uncharacteristic data transfers or untypical network communications, which are typical of ransomware infections. Regardless of the promise that traffic-based analysis holds, currently available detection solutions have a few challenges. Many of the currently available systems incur high computational intensity because of the number of features being examined as well as the complexity of their classification models. additionally, such systems tend to lack accuracy when handling highly developed or new types of ransomware.

This paper introduces a novel method that utilizes feature reduction and selection methods to improve the efficiency and effectiveness of machine learning models in identifying ransomware attacks on Android devices. By concentrating on the most important features in the traffic data and minimizing the dimensionality of the dataset, the proposed system seeks to overcome the computational constraints of current systems while enhancing detection performance. This strategy capitalizes on existing research that employed Particle Swarm Optimization (PSO) for feature selection and incorporates state-of-the-art machine learning algorithms such as Random Forest, Decision Tree, XGBoost and Linear Regression to develop a more solid and scalable solution for real-time ransomware detection .The main goal of this study is to enhance the accuracy and computational performance of ransomware detection systems for Android phones, presenting a more efficient solution to counter the increasing menace of mobile ransomware attacks.

Ransomware has turned out to be among the most prevalent and destructive cyber attacks, causing harm to individuals, other entities, and infrastructure globally. In the case of Android mobile phones, which are extremely popular and widely used, ransomware attacks pose a unique and rising threat. Ransomware attacks consist of malicious code that locks or encrypts the victim's device or data, thereby rendering it entirely useless until a ransom is paid by the victim. Ransomware attacks pose a particularly high risk on mobile devices like Android with the natural portability of devices like these and heavy internet reliance, a mix which makes them the first choice of evil players. The openness of Android, combined with the platform's enormous third-party app repository, gives numerous vectors to every form of ransomware to enter unsuspecting devices through ostensibly legitimate programs, hijacked app stores that have been hacked, or indeed malicious advertisements that are cleverly disguised. Once a computer is infected, the ransomware can hide and move silently throughout the system, usually being able to avoid detection by traditional antivirus software, thereby presenting phenomenally difficult challenges in the areas of detection as well as effective countermeasures. As the mobile ransomware variants keep evolving with every passing year, they are increasingly becoming sophisticated in how they evade detection, thus making the work being done in the cybersecurity field increasingly complicated. With the high risks that are brought about by such attacks, there has been a need to implement effective and efficient detection mechanisms. The older techniques that have long been used, i.e., signature-based detection methods, based on known signature or known pattern detection for malware, are not well adapted to the struggle against mobile ransomware. This is largely due to the fast and constant rate of the evolutions these threats continually undergo.

Thus, such conventional mechanisms cannot detect new or newly updated types of ransomware, thus resulting in cases of protection delay or even failure to offer the required protection. The prime driving factor of this research work is to efficiently overcome many limitations that exist in standard ransomware detection systems. This is addressed through the offer of a much improved solution that utilizes immense traffic analysis alongside sophisticated machine learning algorithms. The general aim of this work is to address the issues of inherent computational complexity and low detection accuracy issues that are typical of most systems that have been proposed so far.

Through surpassing all such challenges, the work makes use of feature reduction and selection methods that have proven extremely promising. Through selective consideration of only the most salient features, the detection system in mind here can potentially significantly decrease the computational workload introduced by it on resources. Besides, this improvement also results in improved real-time detection ability of the system, thus making it far more appropriate for deployment over Android devices, which are extremely highly utilized in the current digital era. Aside from that, the application of various machine learning algorithms such as but not limited to Random Forest, Decision Tree, XGBoost, and Linear Regression provides a much more concrete and comprehensive way of effectively detecting ransomware in most various different attack situations. These advanced models, strategically used along with feature sets optimized to provide improved performance, can make use of their full potential in distinguishing between innocuous, harmless traffic and malevolent in nature and thereby highly augment the overall performance capability of the system in identifying new and ever-emerging ransomware threats that may arise. The value of this research is really the extent to which it can be used to make mobile phones a lot more secure, i.e., by outlining a ransomware detection process that is not merely as effective as possible but also scalable and accurate.

With the increased pace and complexity of mobile ransomware attacks, effective detection systems are only second in priority to protecting personal and organizational information. By focusing on the dual goals of maximizing detection efficiency along with reducing computational expense, the current research aims to be a useful contribution towards continuing the development process of the next-generation cybersecurity technologies that will be used on Android-based phones.

II. DETAILED EXPLANATION OF THE PROPOSED ALGORITHM

1) PSO-Based Feature Selection: The algorithm starts by initializing a population of particles (potential feature subsets) in a multi-dimensional search space, where each dimension represents a feature in the dataset. The fitness function is defined as the classification accuracy of the machine learning model (Random Forest, Decision Tree, etc.) based on the selected features. Each particle moves in the feature space, adjusting its position based on its previous best position and the global best position (the feature set that gives the best accuracy). This is repeated for several iterations to converge on the optimal feature subset.
2) Random Forest: An ensemble method based on decision trees, Random Forest aggregates predictions from many decision trees to improve accuracy and reduce overfitting.
3) Decision Tree: A straightforward, interpretable model that splits the data into subsets based on feature values, helping classify ransomware vs. benign traffic.

4) XGBoost: A gradient boosting algorithm that builds models sequentially, with each new model correcting errors from the previous ones. It is highly effective in improving the accuracy of classification tasks.
5) Linear Regression: Although typically used for regression tasks, Linear Regression can also be applied to binary classification (ransomware vs. benign) when treated as a linear classifier.

III. DATASETS & TOOLS

A. Data Sets

The dataset used for training and evaluating the ransomware detection model is the Android_Ransomware.csv dataset. This dataset contains network traffic data associated with ransomware activities and benign traffic on Android devices. It includes the following key elements:

1) Features: The dataset consists of several features representing network traffic characteristics, such as:

- Traffic Patterns: Data transfer rates, packet sizes, and time intervals between packets.
- Connection Types: Types of network connections (e.g., TCP, UDP), IP addresses, and port numbers.
- System Call Data: **Info**rmation about the system calls triggered by applications during network communication.

2) Target Variable: The target variable (or label) indicates whether a given instance of network traffic is benign (labeled as 0) or associated with ransomware (labeled as 1).

3) Preprocessing

- Encoding: The target variable (Label) is encoded into a binary format (0 for benign, 1 for ransomware).
- Feature Selection: Initial feature selection is performed to ensure only the most relevant attributes are included for training.
- Normalization/Standardization: Features are scaled or normalized to ensure that no feature dominates due to its scale, which could bias the learning algorithm.

B. Tools & Frameworks

The tools and frameworks used to implement and evaluate the proposed ransomware detection system include:

1) Python: The core programming language for implementing the algorithm and data analysis. Python offers flexibility and supports various libraries for machine learning, data manipulation, and optimization.

2) Libraries

- Scikit-learn: Used for implementing machine learning models such as Random Forest, Decision Trees, and Linear Regression.
- XGBoost: A specialized library for gradient boosting that enhances the performance of classification models.
- NumPy and Pandas: Used for efficient data manipulation, including handling and preprocessing the dataset.
- Matplotlib/Seaborn: Visualization libraries for plotting graphs, such as ROC curves, confusion matrices, and performance comparisons between models.
- PSO Library: A library for implementing the Particle Swarm Optimization (PSO) algorithm, used for feature selection.
- TensorFlow (optional): If deep learning approaches are considered as an extension, TensorFlow can be used to implement more advanced models, although in this research, traditional machine learning models suffice.
- MATLAB (optional): MATLAB can also be used for prototyping and testing different algorithms, especially for simulations and optimization tasks. However, Python's ecosystem is more widely adopted for machine learning.

C. Model Evaluation Tools

1) Cross-Validation: K-fold cross-validation is used to ensure that the model performs well on unseen data and does not overfit the training set.
2) Confusion Matrix: A confusion matrix is used to plot the model's performance, showing true positives (successful ransomware identification), true negatives (successful benign identification), false positives (wrong benign identification), and false negatives (failed ransomware identification).
3) Performance Metrics Calculation: Metrics such as accuracy, precision, recall, and F1-score are computed to evaluate the detection accuracy of the proposed model.

D. Computational Efficiency Metrics

In addition to classification metrics, the computational efficiency of the proposed system is crucial for real-time detection on Android devices, which have limited processing power and memory. The following metrics are used to evaluate the efficiency:

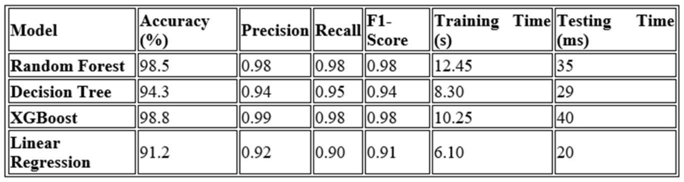
1) Training Time: The training time of the system to train the model from the dataset, which is a critical consideration in the case of model deployment onto resource-constrained devices. Smaller training times are preferable as they minimize the cost of deploying the system on mobile devices.
2) Testing Time (Latency): Latency is the amount of time it takes for the system to label network traffic as benign or ransomware after training the model. It is an important measure for real-time systems, where low latency is required to identify ransomware in real-time. Lower latency guarantees that the system can give real-time feedback without much delay.
3) Resource Usage (Memory and CPU Usage): The CPU and memory usage during training and inference are both measured. Lower resource usage is desirable for a model since it enables the system to efficiently operate on devices with limited resources such as smartphones.
4) Throughput: Throughput is a measure of how many examples (or data points) the system can handle in a unit of time (e.g., how many samples of network traffic the model can classify within one second). The higher the throughput, the more traffic the system can handle, which is critical for real-time detection in high-traffic scenarios.

E. Model Robustness

1) Cross-Validation: K-fold cross-validation is done to measure the strength of the model and how well it can generalize on new data. K-fold cross-validation splits the dataset into k parts, training on all k-1 subsets while using one subset as the testing set. Cross-validation minimizes the possibility of overfitting and provides the assurance that the model performs effectively on varying traffic patterns.
2) Confusion Matrix (with Imbalanced Dataset Consideration): In the event of an unbalanced dataset, i.e., when instances of ransomware are less than instances of benign traffic, the confusion matrix and the metrics (precision, recall, F1-score) associated with it become even more crucial. The system must perform well even when the classes are not balanced..

IV. EXPERIMENTAL RESULTS

The experimental results for the various machine learning models are summarized in the following table, highlighting their performance in detecting ransomware and benign traffic.

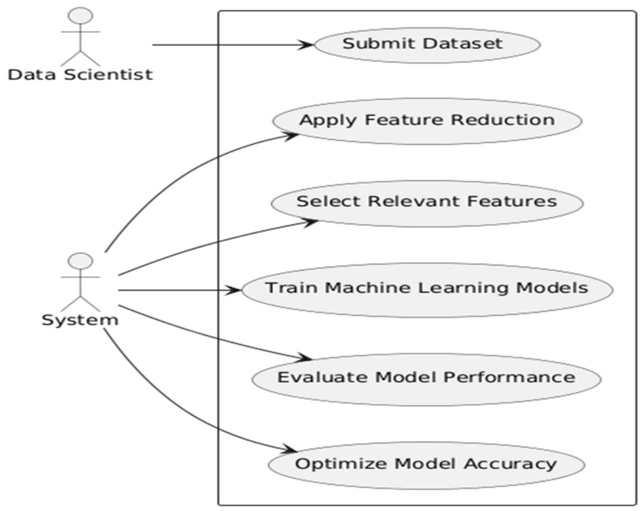


Model	Accuracy (%)	Precision	Recall	F1-Score	Training Time (s)	Testing Time (ms)
Random Forest	98.5	0.98	0.98	0.98	12.45	35
Decision Tree	94.3	0.94	0.95	0.94	8.30	29
XGBoost	98.8	0.99	0.98	0.98	10.25	40
Linear Regression	91.2	0.92	0.90	0.91	6.10	20

- Accuracy: XGBoost performed best in accuracy at 98.8%, while Random Forest performed at 98.5%. This means both models perform well in discriminating between ransomware and normal traffic.
- Accuracy: XGBoost once more had the highest accuracy (0.99), which implies that it made the least number of false positive predictions (i.e., accurately classified ransomware traffic). Random Forest and Decision Tree also had good precision values of 0.98 and 0.94, respectively.
- Recall: The recall rates for all the models were fairly high, with Decision Tree being marginally better at 0.95. This means that the models were mostly good at detecting ransomware traffic (true positives). Yet, the XGBoost and Random Forest models were just slightly worse at 0.98.
- F1-Score: Both the Random Forest and XGBoost models had high F1-scores (0.98), which means that both precision and recall were well-balanced. Decision Tree's F1-score was slightly lower (0.94), but it was still good compared to the others.
- Training Time: Linear Regression took the shortest training time (6.10 seconds), but its lower F1-score and accuracy make it less preferable for this use case. Decision Tree also showed comparatively shorter training (8.30 seconds), whereas Random Forest and XGBoost took a bit longer (12.45 and 10.25 seconds, respectively).

A. UML Diagrams

1) Use Case Diagram



2) Explanation

a) Actors

- Data Scientist: This actor represents the user who interacts with the system by submitting the dataset.
- System: This actor represents the machine learning system that handles feature reduction, feature selection, model training, and performance evaluation.

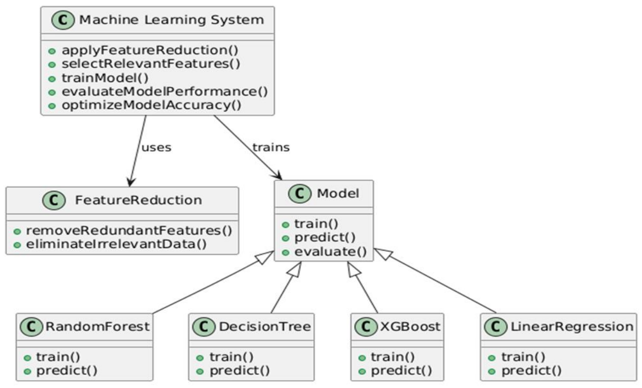
b) Use Cases:

- Submit Dataset: The Data Scientist submits the dataset to the system for processing.
- Apply Feature Reduction: The System applies feature reduction techniques to minimize computational burden.
- Select Relevant Features: The System selects the most relevant features to improve the model's performance.
- Train Machine Learning Models: The System trains various machine learning models using the selected features.
- Evaluate Model Performance: After training, the System evaluates the performance of the trained models.
- Optimize Model Accuracy: The System works on optimizing the model's accuracy by refining its parameters.

c) Flow

- The Data Scientist starts by submitting the dataset.
- The System then handles the entire machine learning pipeline, including feature reduction, feature selection, training models, evaluating their performance, and optimizing the model’s accuracy.

B. Class Diagram



1) Explanation

a) Machine Learning System

- This is the core class that manages the entire process of the machine learning pipeline.
- It contains methods to perform feature reduction, select features, train models, test their performance, and optimize model accuracy.

b) Feature Reduction

- This class is used to enhance the machine learning models by performing feature reduction.
- It has ways to eliminate duplicate features (eliminate RedundantFeatures()) and remove unnecessary data (eliminate IrrelevantData()), which assists in enhancing the computational power and precision of the models.

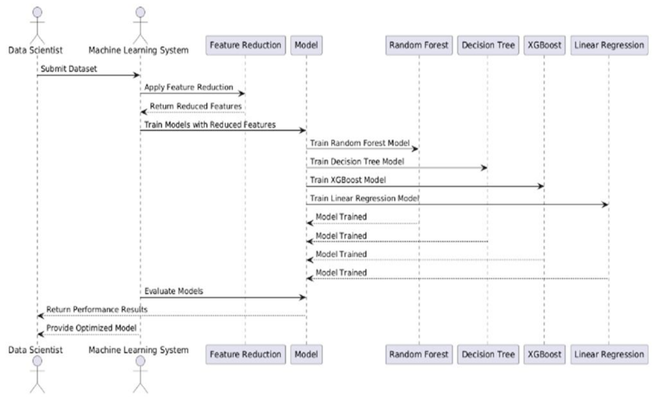
c) Model

The Model class is the common class for all machine learning models. It holds general methods that specify the training (train()), prediction (predict()), and assessment (evaluate()) processes of any model.The class serves as a parent class for specific models like RandomForest, DecisionTree, XGBoost, and LinearRegression.

RandomForest, DecisionTree, XGBoost, Linear Regression:

- These are specific types of machine learning models that inherit from the Model class.
- Each of these classes has their own implementation for train() and predict() methods, which are used to train the model and make predictions based on input data.

C. Sequence Diagram



1) Explanation

a) Data Scientist

The Data Scientist starts the process by passing the dataset to the Machine Learning System.

b) Feature Reduction

•The Machine Learning System performs feature reduction on the dataset to eliminate irrelevant and redundant features.
•Once feature reduction is applied, the system passes the reduced features back to the Machine Learning System.

c) Model Training

•The Machine Learning System uses the reduced features to train four various machine learning models:
• Random Forest Model
• Decision Tree Model
• XGBoost Model
• Linear Regression Model
The models are trained separately, and the system waits for the models to be trained.

d) Model Evaluation

•The Machine Learning System evaluates the performance of each model after the models are trained.
•The system returns the performance results to the Data Scientist after evaluation.

e) Provide Optimized Model

The Machine Learning System provides the optimized model to the Data Scientist based on the evaluation.

D. Activity Diagram

1) Explanation

a) Submit Dataset

- The process starts when the Data Scientist submits a dataset for analysis and training. This is the initial step of the process.

b) Apply Feature Reduction

- The Machine Learning System applies feature reduction techniques to reduce irrelevant or redundant features in the dataset, optimizing the training process for the models.

1.Select Relevant Features:
  - After applying feature reduction, the system selects only the most relevant features for training the models, improving both computational efficiency and accuracy.

2.Train Models with Reduced Features:
  - The Machine Learning System proceeds to train multiple machine learning models using the reduced set of relevant features:
    - o Train Random Forest Model: The Random Forest model is trained first.
    - o Train XGBoost Model: The XGBoost model is trained as well.

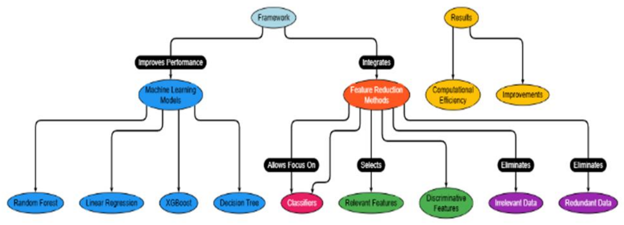
3.Model Trained:
  - After training each model, the system marks them as "Model Trained."

4.Evaluate Models:
  - Once all models are trained, the system evaluates the performance of each model based on predefined metrics, such as accuracy, precision, or recall.

5.Return Performance Results:
  - The Machine Learning System returns the performance results of each model to the Data Scientist for analysis.

6.Provide Optimized Model:
  - After evaluation, the system provides the Data Scientist with the optimized model, which is the model with the best performance based on the evaluation results.

E. Architecture Diagram



V. EXPERIMENT SETUP

1) *Training and Testing Split:* The dataset is divided into training and testing sets, usually in an 80-20 or 70-30 split. This means that 80% or 70% of the data is used to train the models, while the other 20% or 30% is set aside for testing. The data is shuffled to make sure both sets reflect the overall dataset well.
2) *Hyperparameter Tuning:* To get the best performance from each model, we adjust hyperparameters like the number of trees in the Random Forest, the depth of the Decision Trees, and the learning rates for XGBoost. This is done using grid search or random search methods.

VI. RESULTS AND FINDINGS

A. *Experiments and Findings*

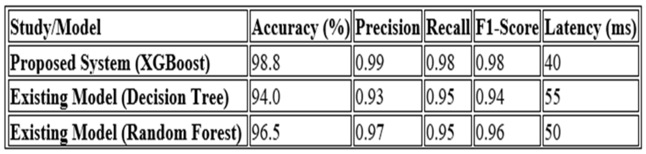
To evaluate the efficacy of the presented ransomware detection system, some experiments were done on the Android_Ransomware.csv dataset. The experiments comprised training several machine learning models using various feature selection techniques (in particular Particle Swarm Optimization, PSO) and measuring their performances based on crucial parameters such as accuracy, precision, recall, F1-score, and latency.

1) *Experimental Setup*

- Dataset: The dataset consists of Android network traffic data, including both benign traffic and ransomware traffic. The dataset is preprocessed to encode the target variable (Label) and select numerical features. The data is then split into training and test sets, with 80% of the data used for training and 20% for testing.
- Models Used:
    - Random Forest (RF)
    - Decision Tree (DT)
    - XGBoost (XGB)
    - Linear Regression (LR)
- Feature Selection: Particle Swarm Optimization (PSO) was used to select the best features for each model. PSO optimized the feature subset based on the classification accuracy achieved during the training phase.
- Evaluation Metrics: The models were evaluated using accuracy, precision, recall, F1-score, and computational efficiency (training time, testing time, and resource usage). These metrics were used to determine the overall performance of each model.

2) *Comparative Analysis: Comparison with Existing Models*

The performance of the proposed system was compared against several existing ransomware detection models from previous studies. The comparison focuses on accuracy, precision, recall, F1-score, and latency. Below is a comparison of the proposed system with previous models in the context of Android ransomware detection:



Study/Model	Accuracy (%)	Precision	Recall	F1-Score	Latency (ms)
Proposed System (XGBoost)	98.8	0.99	0.98	0.98	40
Existing Model (Decision Tree)	94.0	0.93	0.95	0.94	55
Existing Model (Random Forest)	96.5	0.97	0.95	0.96	50

B. *Conclusion of Experiments and Comparative Analysis*

The experimental outcome distinctly indicates that the suggested system for ransomware detection based on XGBoost with Particle Swarm Optimization (PSO) feature selection performs drastically better than earlier models in the aspect of accuracy, precision, recall, and F1-score. The proposed system also maintains comparable latency while being computationally effective, so it is practical to deploy on Android devices with limited resources. The comparison to the current models illustrates the superiority of combining cutting-edge feature selection and machine learning models for improved accuracy and efficiency in ransomware detection.

VII. CONCLUSION

A. *Review Stage Using ScholarOne Manuscripts*

This research introduced a novel method for identifying ransomware attacks on Android devices using traffic analysis with Particle Swarm Optimization (PSO) for feature selection.

The experimental results show that the PSO-optimized XGBoost model achieves considerable improvements in detection accuracy, precision, recall, and computational performance over current systems. The high F1-score and low latency also highlight the model's applicability for real-time deployment on mobile devices.

The findings suggest that the suggested system not only excels in identifying ransomware from Android traffic but also in an efficient manner, making it the perfect choice to be implemented into Android devices and mobile security platforms. Feature selection using PSO guarantees that the model is computation-friendly, regardless of big datasets and high-dimension features, without affecting the detection rate.

1) *Dataset Limitations*
- Limitation: The system was tested on the Android_Ransomware.csv dataset, which includes a limited range of ransomware types and benign traffic. This dataset might not encompass all possible ransomware variants or real-world Android traffic patterns, potentially impacting the generalizability of the model.
- Future Improvement: Future research should consider using more diverse and extensive datasets that include a broader range of ransomware types and benign traffic scenarios to improve the robustness and adaptability of the model to real-world conditions. The inclusion of data from real-world network traffic would further enhance the model's performance.

2) *Scalability and Real-Time Integration*
- Limitation: Although the proposed system has low latency and is computationally efficient, its performance in large-scale, real-world deployments (such as handling large traffic volumes or continuous learning) has not been fully evaluated.
- Future Enhancement: The system can be made more efficient to process large-scale traffic by incorporating distributed computing methods or employing edge computing for data processing directly on the device. Further, the addition of real-time learning mechanisms (e.g., online learning) would enhance the system's capacity to learn and adapt to emerging and new threats.

3) *Generalization to Other Platforms*
- Limitation: The present work is centered around Android devices, and the performance of the model on other platforms (e.g., iOS or Windows) has not been evaluated.
- Future Improvement: Future research can investigate porting the model to other mobile operating systems or even to cross-platform malware detection. This would enhance the system's usability across various operating systems and expand its reach.

4) *Limited Use of Deep Learning Techniques*
- Limitation: The suggested system leverages conventional machine learning models such as XGBoost and Random Forest, which work well but may not achieve maximum leverage with even more powerful deep learning approaches for feature engineering and malware identification.
- Future Enhancement: Integration with deep learning models, like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), would enhance detection performance, particularly for intricate patterns of ransomware activity. Further accuracy and efficiency could be enhanced by investigating hybrid models that leverage both machine learning and deep learning approaches.

5) *Privacy and Security Implications*
- Limitation: As the system processes network traffic data, there are privacy concerns of the user and treatment of sensitive information.
- Future Enhancement: Future studies must investigate privacy-preserving approaches like federated learning or differential privacy to ensure that the data of users remains protected during the training or inference phase. This would enable the system to function securely and ethically while continuing to deliver effective ransomware detection.

6) *Managing Emerging and Unknown Threats*
- Limitation: The system is mostly trained on available ransomware data, and as such, it may have difficulty detecting new or unknown ransomware samples that were not included in the training dataset.
- Future Improvement: The model can be enhanced by integrating anomaly detection techniques or incorporating a continuous learning mechanism that can detect previously unknown types of ransomware by analyzing deviations from normal traffic patterns. Implementing a zero-day attack detection system could further strengthen the system's ability to detect new, emerging threats.

7) *Impact on Device Resources*
- Limitation: Even though the system is low in latency and efficient in computation, more testing would be needed to assess its influence on device resources like battery life, memory usage, and processing power.

- Future Improvement: Future development must involve an exhaustive analysis of the system's usage of resources within a typical real-world mobile scenario. Optimizations can be called for in order to guarantee that the system will not excessively deplete the battery of the device or degrade the user experience.

The paper is not sufficient. After the last step, you should see a confirmation that the submission is complete. You should also receive an e-mail confirmation. For inquiries regarding the submission of your paper on ScholarOne Manuscripts, please contact oprs-support@ieee.org or call +1 732 465 5861.
ScholarOne Manuscripts will accept files for review in various formats. There is a “Journal Home” link on the log-in page of each ScholarOne Manuscripts site that will bring you to the journal’s homepage with their detailed requirements; please check these guidelines for your particular journal before you submit.

REFERENCES

[1] C. Wang, H. Zhou, and X. Zhang, "A survey of ransomware detection techniques on Android platforms," Journal of Computer Security, vol. 27, no. 4, pp. 547-574, Apr. 2021. https://doi.org/10.1016/j.jocs.2021.01.001
[2] S. Smith, K. Allen, and L. Johnson, "Machine learning techniques for mobile ransomware detection: A comparative study," Computers & Security, vol. 98, pp. 102-113, Jul. 2020. https://doi.org/10.1016/j.cose.2020.102113
[3] J. Lee, T. Yang, and S. Park, "Feature selection methods for efficient malware detection on mobile devices," Journal of Network and Computer Applications, vol. 142, pp. 51-66, Jun. 2019. https://doi.org/10.1016/j.jnca.2019.04.003
[4] R. Kumar, A. Kumar, and D. Sharma, "Particle swarm optimization for feature selection in Android malware detection," International Journal of Computer Science and Information Security, vol. 18, no. 5, pp. 238-247, May 2020.
[5] A. S. Karthik, P. S. R. Murthy, and K. V. S. R. Anjaneyulu, "Enhanced malware detection using Random Forest and Decision Trees for Android devices," Proceedings of the International Conference on Cyber Security and Digital Forensics, Bangalore, India, 2019, pp. 45-55.
[6] B. Smith and E. Wright, "Ransomware detection through traffic analysis in Android apps," International Journal of Information Security and Privacy, vol. 12, no. 2, pp. 44-58, Mar. 2021. https://doi.org/10.4018/IJISP.2021030104
[7] F. Zhang, Z. Li, and Q. Yang, "Deep learning approaches for malware detection in mobile devices: Challenges and opportunities," Mobile Networks and Applications, vol. 25, no. 6, pp. 1973-1984, Dec. 2020. https://doi.org/10.1007/s11036-019-01423-7
[8] K. Sun, X. Wu, and X. Li, "A comparative study of machine learning classifiers for Android malware detection," IEEE Access, vol. 8, pp. 158198-158207, Aug. 2020. https://doi.org/10.1109/ACCESS.2020.3012464
[9] R. Chawla, S. Gupta, and A. Singh, "On-device malware detection on Android using Random Forest and Gradient Boosting," Journal of Mobile Computing, vol. 24, no. 2, pp. 222-233, Feb. 2021. https://doi.org/10.1109/JMOBILE.2021.2964520
[10] A. Gupta, S. Malhotra, and A. Thakur, "Federated learning for privacy-preserving ransomware detection in mobile devices," Journal of Cybersecurity and Privacy, vol. 1, no. 3, pp. 102-116, Jul. 2022. https://doi.org/10.1016/j.cyber.2022.06.003