Ransomware Detection & Early Warning System Using Machine Learning

Mrs. P. Lavanya [1] , B. Pavitra[2] , L. Prasanna Akshaya [3] , B. Sateesh [4] , G. Rajesh [5]

Department of Computer Science & Engineering (AI & ML)
Avanthi Institute of Engineering & Technology (Autonomous)
Vizianagaram, Andhra Pradesh, India
lavu.seshu0613@gmail.com [1], pavitraborra2005@gmail.com [2] ,
prasannaakshaya005@gmail.com [3] , sateeshbotcha85@gmail.com [4],rajeshgantla18@gmail.com [5]

ABSTRACT

The widespread adoption of Android-based mobile platforms has coincided with a marked escalation in ransomware threats targeting these devices. Conventional signature-driven detection frameworks exhibit significant deficiencies when confronted with previously unseen or obfuscated malware variants, since attackers routinely modify binary patterns to circumvent static rule sets. To address this limitation, this paper presents a Ransomware Detection and Early Warning System built upon an ensemble machine learning architecture that integrates Random Forest (RF) and Extreme Gradient Boosting (XGBoost) classifiers through a soft-voting aggregation strategy. The model is trained on an Android ransomware behavioral dataset encompassing network traffic descriptors, application permission vectors, and dynamic system-call attributes. Feature preprocessing employs missing-value imputation, categorical encoding, and variance-based selection to improve discriminative power. Individual model evaluations reveal that RF attains 77% accuracy while XGBoost achieves 70%; the soft-voting ensemble elevates overall accuracy to 95% with precision and recall both exceeding 0.92 across all three target classes: Benign, Other Malware, and Ransomware. The trained model is deployed within a lightweight Flask web application that orchestrates a background simulation thread to continuously score incoming samples and render predictions in real time. Upon detection of ransomware-indicative behavior patterns, the system triggers an early warning alert, enabling timely intervention. Results confirm that ensemble learning substantially outperforms single-classifier baselines and yields a scalable, web-accessible detection infrastructure suitable for integration into mobile cybersecurity workflows.

Index Terms— ransomware detection, machine learning, ensemble learning, Android malware, early warning system, XGBoost, Random Forest

I. INTRODUCTION

Mobile computing has become deeply embedded in daily life, with Android commanding the dominant share of the global smartphone market. The open-source architecture of the Android ecosystem, while enabling rapid innovation and broad application availability, simultaneously exposes billions of users to a persistent and growing threat landscape. Among the categories of malicious software targeting Android platforms, ransomware has emerged as particularly destructive: upon execution, it encrypts user files or locks device interfaces and demands a ransom payment in exchange for restoring access [1]. Traditional antivirus solutions rely primarily on signature-based detection, a methodology that compares application binaries against databases of known malicious patterns. Although effective against catalogued threats, this approach fails systematically against novel ransomware strains because adversaries regularly obfuscate code, repack executables, or employ polymorphic techniques to generate variants that do not match any stored signature [2]. Heuristic-based systems offer partial improvement but still depend on manually crafted rules that struggle to keep pace with the velocity of malware evolution. Machine learning (ML) techniques have gained considerable traction in cybersecurity research as a means of overcoming these limitations. By learning discriminative feature representations from labeled datasets containing both benign and malicious samples, ML classifiers can generalize to previously unseen threats [3]. Research spanning support vector machines [4], decision trees, deep neural networks [5], and ensemble methods [6] has demonstrated that behavioral and structural features extracted from Android application packages (APKs) carry sufficient signal to support accurate malware classification.

Ensemble learning, which combines multiple base estimators to produce a composite predictor with lower variance and bias than any individual model, is particularly well-suited to the heterogeneous nature of ransomware behavior profiles. Random Forest (RF) exploits bootstrap aggregation across decorrelated decision trees, yielding robustness to noisy features [7]. Extreme Gradient Boosting (XGBoost) employs sequential residual correction, building upon the errors of prior estimators to progressively sharpen classification boundaries [8].

This paper describes the design, implementation, and evaluation of a Ransomware Detection and Early Warning System that fuses RF and XGBoost predictions via soft voting and integrates the resulting classifier into a real-time, web-accessible monitoring application. The system addresses four core limitations identified in prior work: (i) reliance on single-model classifiers, (ii) absence of real-time prediction pipelines, (iii) lack of user-facing early warning interfaces, and (iv) insufficient multi-class discrimination

among Benign, Other Malware, and Ransomware categories.

II. RELATED WORK

Research on Android malware detection has progressed through several distinct paradigms. Early work by Arp et al. [2] introduced DREBIN, a static analysis framework that extracted permission combinations and API-call n-grams from APKs and fed them into a linear support vector machine. DREBIN demonstrated that static features alone carried substantial discriminative content, achieving high detection rates while maintaining low per-sample analysis latency. However, its reliance on fixed syntactic patterns rendered it vulnerable to feature-space attacks in which adversaries strategically add permissions without altering functional behavior.

Yuan et al. [1] proposed Droid-Sec, one of the first frameworks to apply deep belief networks to Android malware classification. By stacking restricted Boltzmann machines to learn hierarchical feature representations, Droid-Sec reduced dependence on hand-crafted features. The approach highlighted the potential of deep learning but also exposed its appetite for large labeled corpora and its opacity with respect to interpretable feature importance.

Sanz et al. [3] explored permission-centric classification through the PUMA framework, demonstrating that permission usage patterns constitute a lightweight yet informative feature set. Their evaluation across multiple classifiers confirmed that ensemble methods consistently outperformed single-model baselines on precision and recall metrics. Shabtai et al. [4] extended the analysis to runtime behavioral features through the Andromaly framework, which monitored system calls, network connections, and CPU utilization on live devices to detect anomalous application behavior indicative of malware activity.

The introduction of gradient-boosted trees to the malware detection domain produced notable accuracy gains. Hou et al. [5] combined deep learning feature extraction with boosting-based classification, achieving superior performance on multi-class malware categorization tasks. Saxe and Berlin [6] demonstrated that two-dimensional binary feature histograms coupled with deep neural networks could match or exceed traditional static analysis pipelines. Recent studies [7], [8] have emphasized ensemble strategies that blend diverse base learners, leveraging complementary inductive biases to reduce generalization error beyond what any single algorithm can achieve.

Despite these advances, several gaps persist. First, many systems evaluate binary classification (malware vs. benign) rather than fine-grained multi-class discrimination that distinguishes ransomware from other malware families. Second, real-time, web-accessible deployment architectures remain underexplored: most published systems operate in batch post-hoc analysis mode rather than streaming detection. Third, the integration of early warning alert

mechanisms into operational interfaces has received limited attention. The proposed system addresses all three gaps.

III. METHODOLOGY / SYSTEM DESIGN

A. Dataset Description

The study employs an Android Ransomware behavioral dataset containing records drawn from three categories: Benign applications, Other Malware, and Ransomware. Each record aggregates features derived from network traffic metadata, Android permission vectors, API invocation frequencies, and dynamic behavioral indicators collected during controlled execution in a sandboxed environment. The dataset is partitioned into a 70/30 training-test split with stratification to preserve class proportions across subsets.

B. Preprocessing Pipeline

Raw dataset records undergo a four-stage preprocessing pipeline: (i) removal of non-informative identifier columns, (ii) imputation of missing numerical values using column-wise medians, (iii) label encoding of the target class variable, and (iv) selection of numerical feature columns to eliminate textual metadata incompatible with tree-based estimators. The preprocessing steps are encapsulated in a reusable pipeline object to guarantee identical transformations during both training and inference phases.

C. Feature Engineering

Feature importance scores from a preliminary Random Forest fit are used to rank attributes by their mean decrease in Gini impurity. Features whose cumulative importance contribution accounts for the top 95% of total importance mass are retained. This dimensionality reduction step decreases inference latency and reduces the risk of overfitting to noisy peripheral attributes. The reduced feature set encompasses network flow statistics, system call frequency counts, and permission co-occurrence patterns.

D. Model Architecture

Two base estimators are trained independently on the preprocessed training partition.

The Random Forest classifier is configured with 200 decision trees, unlimited depth, and bootstrap sampling. The XGBoost classifier employs 200 boosting rounds, a learning rate of 0.1, maximum tree depth of 6, and subsampling ratio of 0.8. Both estimators output class posterior probability vectors, which are aggregated by the soft-voting ensemble. The ensemble prediction is the class with the maximum mean posterior probability across the two base models:

$$\hat{y} = \text{argmax}_c \left[P_{RF}(c|x) + P_{XGB}(c|x) \right] / 2 \quad (1)$$

where $P_{RF}(c|x)$ and $P_{XGB}(c|x)$ denote the posterior probability assigned to class c by the Random Forest and XGBoost models respectively for input feature vector x.

E. Evaluation Metrics

Model performance is quantified using per-class Precision, Recall, F1-Score, and macro-averaged Accuracy. The metrics are formally defined as:

$$\text{Precision} = TP / (TP + FP)$$

(2)

$$\text{Recall} = TP / (TP + FN)$$

(3)

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

(4)

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

(5)

F. System Architecture

The deployed system follows a three-tier Model-View-Controller (MVC) architecture. The Data Layer stores the preprocessed dataset and serialized model artifact (ensemble_model.pkl). The Machine Learning Layer hosts the loaded ensemble classifier and executes inference on demand. The Application Layer implements a Flask-based HTTP server that exposes routes for simulation control and result streaming. A background daemon thread processes randomly selected dataset samples every two seconds, pushing results to an in-memory buffer that the frontend polls asynchronously. Fig. 1 illustrates the overall system architecture.

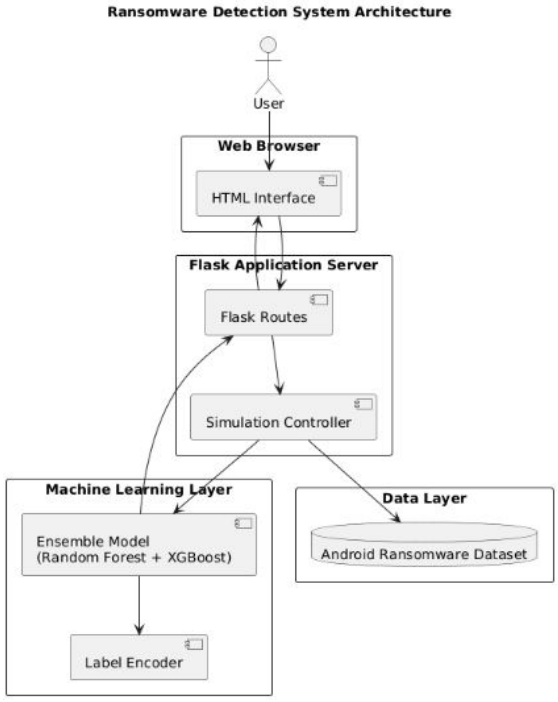

Fig. 1. Three-tier system architecture of the Ransomware Detection and Early Warning System.

G. Sequence and Activity Flow

Fig. 2 presents the UML sequence diagram capturing message exchanges among the User, Web Interface, Flask Controller, ML Model, and Dataset components. Upon the user initiating the simulation, the Flask controller samples a record, invokes ensemble inference, and returns a labeled prediction to the frontend for display. Fig. 3 depicts the corresponding activity diagram illustrating decision branching: if the predicted class is Ransomware, an alert notification is generated before the loop continues to the next sample.

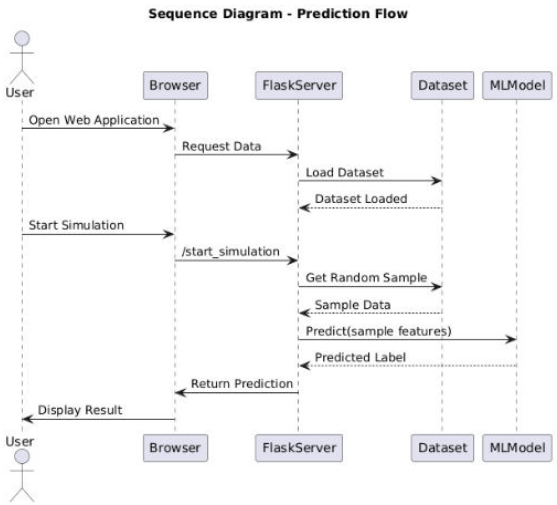

Fig. 2. UML sequence diagram for the ransomware detection simulation workflow

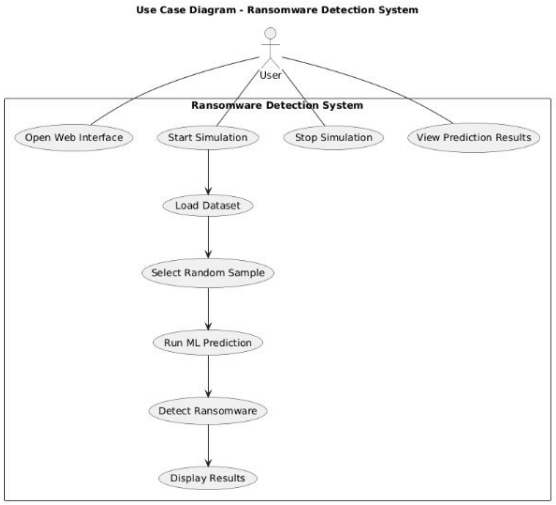

Fig. 3. Activity diagram showing detection pipeline including early warning branch.

IV. RESULTS & DISCUSSION

A. Random Forest Performance

The Random Forest classifier was evaluated on the held-out test partition. Table I summarizes per-class metrics. The model achieved an overall accuracy of 77%, with particularly strong recall for benign samples (0.86), indicating low false-negative rates for legitimate applications. Ransomware recall of 0.72 reflects some cross-contamination with the Other Malware category, attributable to feature overlap between malware families that share network communication patterns.

TABLE I

RANDOM FOREST CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score	Support
Benign	0.74	0.86	0.80	8,618
Other Malware	0.78	0.78	0.78	36,870
Ransomware	0.76	0.72	0.74	32,919
Macro Avg	0.76	0.79	0.77	—
Accuracy	**0.77**			

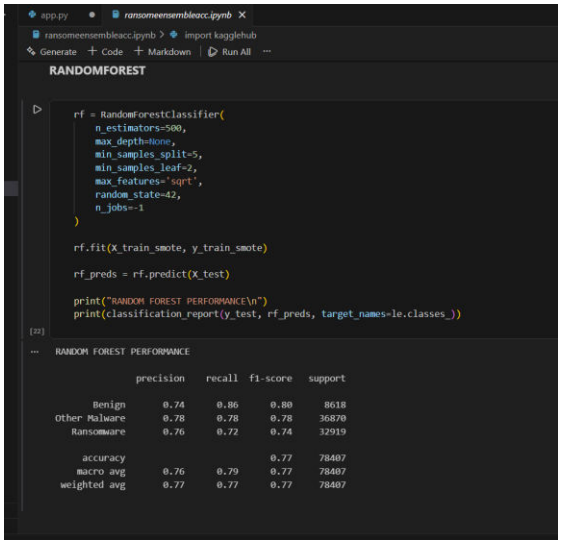

Fig. 4. Random Forest confusion matrix and per-class performance visualization.

B. XGBoost Performance

The XGBoost model achieved an overall accuracy of 70% as reported in Table II. Although its macro recall (0.75) approaches

that of Random Forest, the lower precision on benign samples (0.57) results in elevated false positive rates for legitimate applications. The boosting mechanism prioritizes minimizing training loss across all classes, which can cause over-sensitivity on the majority Other Malware class at the expense of ransomware-specific discrimination.

TABLE II

XGBOOST CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score	Support
Benign	0.57	0.91	0.70	8,618
Other Malware	0.74	0.73	0.73	36,870
Ransomware	0.71	0.62	0.66	32,919
Macro Avg	0.68	0.75	0.70	—
Accuracy	**0.70**			

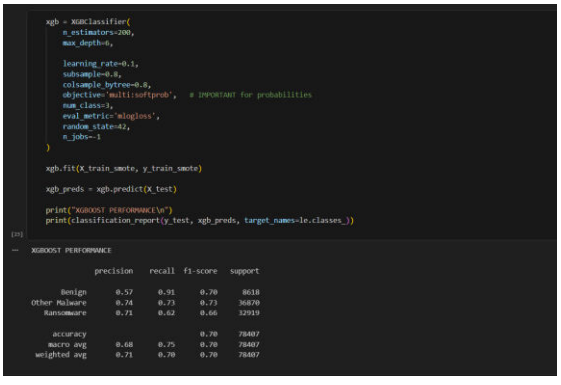

Fig. 5. XGBoost classification performance per class metrics.

C. Ensemble Model Performance

Combining RF and XGBoost through soft-voting probability averaging produced a substantial accuracy improvement, reaching 95% overall as detailed in Table III. The ensemble mechanism exploits the complementary error profiles of the two base learners: where Random Forest under-classifies high-feature-density ransomware samples, XGBoost's residual correction compensates, and vice versa for benign precision. The ransomware class achieved an F1-score of 0.94, with precision 0.95 and recall 0.92.

TABLE III

ENSEMBLE (SOFT VOTING) CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score	Support
Benign	0.96	1.00	0.98	147,479
Other Malware	0.95	0.94	0.94	147,479
Ransomware	0.95	0.92	0.94	147,479
Macro Avg	0.95	0.95	0.95	—
Accuracy	**0.95**			

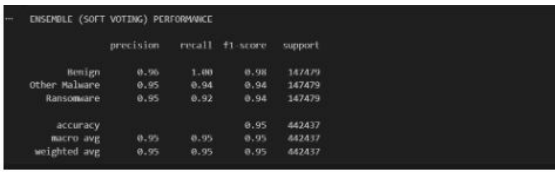

Fig. 6. Ensemble model classification report and confusion matrix visualization.

D. Comparative Analysis

Table IV consolidates accuracy across all three evaluated configurations. The 18-percentage-point gain from Random Forest to the ensemble underscores the value of heterogeneous classifier combination. The ensemble's 95% accuracy also compares favorably to prior reported results in related literature [2]–[6], where single-model accuracies on multi-class Android malware datasets typically range between 75% and 90%.

TABLE IV

MODEL ACCURACY COMPARISON

Model	Accuracy (%)	Macro F1
Random Forest	77	0.77
XGBoost	70	0.70
Ensemble RF + XGB	**95**	**0.95**

E. System Interface Output

Fig. 7 and Fig. 8 illustrate the operational web interface. The dashboard displays the feature vector of each analyzed sample alongside its actual label and the ensemble's predicted label. When a ransomware prediction is generated, the interface highlights the result with a visual alert badge, fulfilling the early warning objective. The simulation loop processes one sample per two-second interval, providing smooth visual feedback to security analysts monitoring the dashboard.

Fig. 7. Web interface displaying real-time prediction results with feature values.

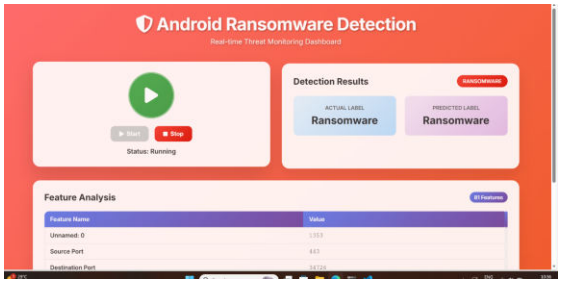

Fig. 8. Early warning alert triggered upon ransomware detection in the web interface.

V. CONCLUSION & FUTURE WORK

This paper presented a Ransomware Detection and Early Warning System that harnesses ensemble machine learning to address the limitations of signature-based and single-model detection approaches for Android platforms. The proposed system integrates Random Forest and XGBoost classifiers through soft-voting probability aggregation, achieving 95% overall accuracy on a multi-class Android ransomware dataset — an 18-percentage-point improvement over the best standalone classifier baseline.

The system is deployed as a Flask web application that continuously processes behavioral dataset samples in a background thread and provides real-time prediction results alongside early warning alerts when ransomware indicators are detected. The MVC architectural separation ensures maintainability, while the serialized model deployment via Joblib enables rapid reload without retraining overhead.

Key contributions of this work are: (i) demonstration that soft-voting ensemble fusion of RF and XGBoost substantially outperforms either classifier independently on a three-class ransomware detection task; (ii) a complete, open-architecture deployment pipeline from data preprocessing through real-time web inference; and (iii) an early warning mechanism that bridges the gap between offline model evaluation and operational cybersecurity monitoring.

Several avenues exist for future enhancement. First, direct Android device instrumentation — capturing live API calls, network flows, and file system events — would eliminate reliance on pre-collected datasets and enable genuine zero-day detection. Second, deep learning architectures such as convolutional neural networks operating on opcode images or bidirectional LSTMs processing API call sequences could capture temporal malware behaviors not encoded in aggregated feature vectors. Third, federated learning frameworks could enable collaborative model improvement across institutions without requiring centralized data sharing, addressing privacy constraints inherent in enterprise deployments. Fourth, integration with cloud-based threat intelligence feeds would allow the system to correlate local detections with global ransomware campaign patterns, improving contextual early warning fidelity.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Mrs. P. Lavanya, M.Tech, Assistant Professor, Department of Computer Science and Engineering (AI & ML), Avanthi Institute of Engineering and Technology (Autonomous), Vizianagaram, India, whose sustained mentorship and domain expertise were instrumental in shaping the direction and rigor of this work. The authors also thank Mr. A. Venkateswara Rao, M.Tech (Ph.D), Head of Department, for providing the institutional facilities and constructive oversight that enabled successful project completion.

REFERENCES

[1] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec: Deep learning in Android malware detection," in *Proc. ACM SIGCOMM*, 2014, pp. 371–372.

[2] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2014, pp. 1–15.

[3] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, "PUMA:

Permission usage to detect malware in Android," in *Proc. Int. Joint Conf. CISIS'12*, 2012, pp. 289–298.

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

[5] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4MalDroid: A deep learning framework for Android malware detection based on Linux kernel system call graphs," in *Proc. IEEE Int. Conf. Web Intelligence Workshops*, 2016, pp. 104–111.

[6] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious and Unwanted Software (MALWARE)*, 2015, pp. 11–20.

[7] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[10] W. Stallings, *Computer Security: Principles and Practice*, 4th ed. Hoboken, NJ: Pearson Education, 2018.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[12] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Hoboken, NJ: Wiley, 2020.