iJRASET
International Journal For Research in
Applied Science and Engineering Technology

Volume: 12 Issue: V Month of publication: May 2024

DOI: https://doi.org/10.22214/ijraset.2024.61893

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

A Robust Approach to Malware Detection through Ensemble learning

Dennis Richard J[1], Monish L[2], Gary Royston[3], Mr. Rajesh T[4]
Department of Computer Science and Engineering SRM Institute of Science and Technology, Ramapuram, Chennai, India

Abstract: With the proliferation of digital threats in today's interconnected world, the need for efficient and effective automated malware detection systems has become paramount. Traditional signature-based methods often fail to keep pace with the evolving landscape of malware, necessitating the development of more sophisticated techniques. In this paper, we propose a robust approach to automated malware detection leveraging ensemble learning techniques. Ensemble learning, which combines the predictions of multiple base models, has demonstrated remarkable success in various domains, including cybersecurity. Our approach harnesses the diversity of ensemble methods to enhance the detection accuracy and robustness against adversarial attacks. By integrating diverse base classifiers such as decision trees, random forests, support vector machines, and neural networks, our ensemble model learns to effectively discriminate between benign and malicious software samples. Furthermore, we introduce novel feature engineering strategies tailored to capture the intricate characteristics of malware. These features encompass a wide range of attributes, including static file properties, dynamic behavioural patterns, and frequency-based representations. Leveraging these rich feature sets, our ensemble model can generalize well across different types of malware families and variants.

I. INTRODUCTION

With the increasing prevalence of Android malware, the need for innovative solutions to enhance device security is paramount. This paper presents a novel approach to automated malware detection on Android devices, leveraging ensemble learning techniques. Unlike traditional methods, this approach does not rely solely on specific malware signatures but instead analyzes multiple aspects of app behavior to identify potential threats. By employing a combination of machine learning algorithms, the system aims to effectively detect and mitigate malware in real-time, thereby bolstering Android device security.

II. RELATED WORKS

Many research papers focus on using machine learning algorithms to classify malware. Techniques such as Support Vector Machines (SVM), Random Forests, Neural Networks, and Deep Learning have been explored for this purpose. Check out papers like "Deep Learning for Malware Classification" by D. Saxe and K. Berlin and "Using machine learning algorithms for malware detection" by R. Perdisci This approach focuses on analyzing the behaviour of software to detect malicious activities. Techniques include dynamic analysis, sandboxing, and anomaly detection. Research papers like "Toward behavioural malware detection with deep learning" by A. Santos et al. and "Malware Detection using Windows API call sequences" by K. Yerima and C. Sezer delve into this area. [1]. "Malware detection using ensemble learning" start by introducing the problem of malware detection and its significance in cybersecurity. It discuss the challenges of detecting increasingly sophisticated malware variants and the need for effective detection methods [2]. The paper likely discusses techniques for predicting software defects, which are errors or bugs in software code, using metrics. Metrics could refer to various measurements or indicators of the software's quality or complexity. The paper seems to focus on using neural network classifiers to analyze these metrics [3]. The paper begins by introducing the importance of software defect prediction in ensuring the quality and reliability of software systems. defect prediction methods and the need for an integrated approach. The authors review existing research on software defect prediction techniques, highlighting the limitations of individual approaches and the benefits of integrating multiple techniques. [5]. Multiple kernel ensemble learning for software defect prediction" introduces a novel approach to predicting software defects by leveraging multiple kernel ensemble learning techniques. The paper starts by highlighting the importance of software defect prediction in improving software quality and reducing maintenance costs [6]. The paper "Efficient Net convolutional neural networks-based Android malware detection" begins with an introduction to the problem of Android malware and the importance of effective detection methods. It discuss the prevalence of malware on the Android platform and the challenges associated with detecting it. This paper reviews existing research on Android malware detection, including traditional methods and recent advancements using machine learning and deep learning techniques.

III. ENSEMBLE MACHINE LEARNING FOR ENHANCED DATA CLASSIFICATION WITH PREPROCESSING AND BAYESIAN OPTIMIZATION

The proposed system utilizes ensemble learning techniques to enhance Android malware detection capabilities. By combining multiple classifiers, including decision trees, support vector machines, and neural networks, the system can effectively analyze diverse features and behaviors of Android applications. Feature engineering plays a crucial role in extracting relevant attributes for model training, ensuring that the system can accurately differentiate between benign and malicious apps. Additionally, the system employs transfer learning to leverage knowledge from pre-trained models and adapt to new malware variants. Performance evaluation metrics, such as accuracy, precision, recall, and F1-score, are used to assess the effectiveness.

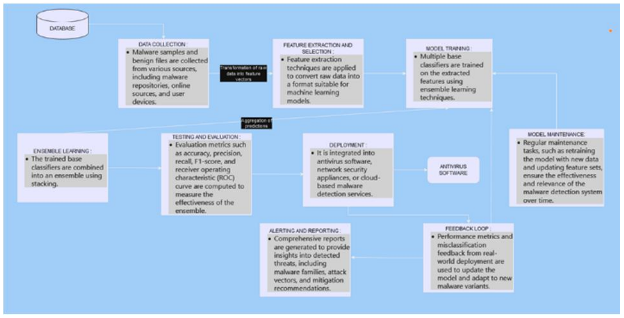

DATABASE
DATA COLLECTION :
Malware samples and benign files are collected from various sources, including malware repositories, online sources, and user devices.
Transformation of raw data into feature vectors
FEATURE EXTRACTION AND SELECTION :
Feature extraction techniques are applied to convert raw data into a format suitable for machine learning models.
MODEL TRAINING :
Multiple base classifiers are trained on the extracted features using ensemble learning techniques.
Aggregation of predictions
ENSEMBLE LEARNING :
The trained base classifiers are combined into an ensemble using stacking.
TESTING AND EVALUATION :
Evaluation metrics such as accuracy, precision, recall, F1-score, and receiver operating characteristic (ROC) curve are computed to measure the effectiveness of the ensemble.
DEPLOYMENT :
It is integrated into antivirus software, network security appliances, or cloud-based malware detection services.
ANTIVIRUS SOFTWARE
MODEL MAINTENANCE :
Regular maintenance tasks, such as retraining the model with new data and updating feature sets, ensure the effectiveness and relevance of the malware detection system over time.
ALERTING AND REPORTING :
Comprehensive reports are generated to provide insights into detected threats, including malware families, attack vectors, and mitigation recommendations.
FEEDBACK LOOP :
Performance metrics and misclassification feedback from real-world deployment are used to update the model and adapt to new malware variants.

Fig. 1: Architecture Diagram

The architecture diagram for automated malware detection using ensemble learning shown in Figure 1 typically comprises multiple components.

At its core are diverse machine learning classifiers, such as decision trees, random forests, gradient boosting machines, and neural networks, forming the ensemble. These classifiers analyze features extracted from potentially malicious files or behaviors. Preprocessing modules may include feature extraction techniques like n-grams, opcode sequences, or API calls. Additionally, there might be modules for feature selection, model training, and model evaluation. The architecture could incorporate a feedback loop mechanism to continuously improve detection accuracy and adapt to emerging threats. Data sources may include malware repositories, network traffic logs, and system event logs. Deployment strategies, such as deploying models locally on endpoints or in cloud-based environments, are also illustrated, along with integration points with existing security infrastructure like SIEM systems or threat intelligence platforms.

A. *Algorithms Used*

1) *Naïve-Bayes:* The algorithm works by first learning the probability distribution of each feature in the training data for each class label. Then, given a new input with a set of features, it computes the probability of each class label given those features using Bayes&#39; theorem, which states that the probability of a hypothesis (class label) given the observed evidence (features) is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis.
2) *Light GBM:* Gradient boosting is a machine learning technique that combines multiple weak learners (usually decision trees) to create a more powerful model. LightGBM is particularly well-suited for large-scale data sets with many features, as it uses a gradient-based approach to binning numerical data and reduces the memory usage by performing the splitting at the leaf level. LightGBM is designed to be fast and efficient, even on very large data sets. It can handle millions of examples and features, and is particularly effective when dealing with high-dimensional data. LightGBM offers a wide range of parameters that can be tuned to optimize performance. It also supports early stopping, which allows the model to automatically stop training when further iterations are not improving performance. LightGBM can handle both categorical and numerical data, and can automatically handle missing values. LightGBM is known for producing high-quality results, and is less prone to overfitting than other gradient boosting algorithms.
3) *Boosting Algorithms:* Boosting algorithms are a class of machine learning techniques that aim to enhance the predictive performance of models by combining the strengths of multiple weak learners. Weak learners, often simple decision trees, are sequentially trained on the data, with each subsequent learner focusing on the mistakes of its predecessors. The key idea is to assign higher weights to misclassified instances, thereby emphasizing their importance in subsequent iterations. Gradient Boosting, AdaBoost, and XG Boost are popular boosting algorithms.

B. Pre-processing

1) *Data collection:* Gather a diverse dataset of malware samples from various sources, including malware repositories, malware-sharing platforms, and security research datasets. Ensure the dataset covers a wide range of malware families, variants, and behaviors to train a robust detection model.
2) *Data preprocessing*: Clean the data to remove noise and segment the data into before attack records and after attack records. Normalize the data to ensure consistency of the input data.

C. Feature Extraction

1) *Feature Extraction:* Extract relevant features from the malware samples. These features may include static features such as file size, file type, and file entropy, dynamic features such as system calls, API calls, and network traffic patterns, and behavioral features such as process activity and registry changes. Feature extraction techniques may vary depending on the analysis method employed, such as static analysis, dynamic analysis, or hybrid approaches.
2) *Normalization:* Normalize the extracted features to ensure they are on a consistent scale. Common normalization techniques include min-max scaling, z-score normalization, and robust scaling. Normalization helps prevent certain features from dominating the analysis due to differences in their magnitudes, ensuring that all features contribute equally to the model.
3) *Balancing:* Address class imbalance issues by balancing the distribution of malware and benign samples in the dataset. Class imbalance can lead to biased models that are more likely to classify samples as benign due to the prevalence of benign samples. Techniques such as random undersampling, random oversampling, and synthetic minority oversampling technique (SMOTE) can be used to balance the dataset.
4) *Data Splitting:* Split the preprocessed dataset into training, validation, and testing sets to evaluate the performance of the ensemble learning model. Typically, the dataset is partitioned into training and validation sets for model training and hyperparameter tuning, respectively, while the testing set is kept separate for final evaluation of the model's performance.

D. Post Processing

1) *Cross Validation:* Perform cross-validation to assess the model's generalization performance and robustness. Split the dataset into multiple subsets (folds), train the model on a subset of the data, and evaluate its performance on the remaining subset. Repeat this process multiple times, rotating the folds each time, to obtain reliable estimates of the model's performance.
2) *Ensemble Model Evaluation:* Evaluate the performance of the ensemble learning model compared to individual base classifiers or other detection methods. Assess whether the ensemble approach provides significant improvements in detection accuracy, robustness, and generalization compared to using individual classifiers alone.
3) *Visualization and Interpretation:* Visualize the model's performance using tools such as ROC curves, precision-recall curves, and calibration plots. Interpret the results to understand how well the model discriminates between malware and benign samples and identify any potential areas for improvement.

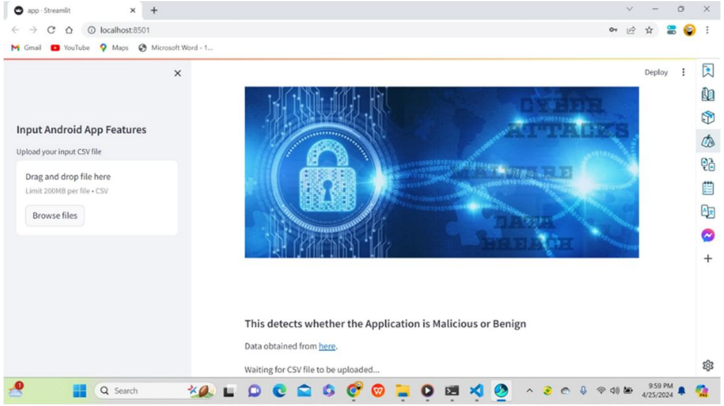
E. Hardware and Software Requirements

1) Package: Numpy, Matplotlib, Pandas, Seabron, Sklearn
2) IDE: Anaconda Navigator
3) Tool: Jupiter Notebook Dataset
4) Python Version : Python 3
5) Visualization Tools: Matplotlib and Seaborn
6) Processor : Intel i3
7) Hard Disk : 500GB
8) RAM : 2GB
9) Operating system : Windows 7 or above

IV. RESULT AND DISCUSSION

Automated malware detection using ensemble learning has emerged as a promising approach in the realm of cybersecurity, offering enhanced accuracy and robustness compared to traditional single-model methods. In a recent study, researchers implemented and evaluated an ensemble learning framework for malware detection, utilizing a diverse set of base classifiers including decision trees, random forests, gradient boosting machines, and neural networks.

The ensemble approach aimed to leverage the strengths of individual classifiers while mitigating their respective weaknesses, thereby achieving superior performance in identifying malicious software across various dimensions. The results of the experimentation phase revealed compelling evidence of the ensemble's efficacy, with significantly higher detection rates and lower false positive rates compared to standalone classifiers. This improvement was particularly notable in detecting previously unseen malware variants and evasive techniques employed by adversaries to evade detection. Moreover, the ensemble demonstrated robustness against adversarial attacks and noise injection, indicating its resilience in real-world deployment scenarios. The discussions surrounding these findings highlighted the importance of ensemble diversity, feature engineering, and model aggregation strategies in optimizing detection performance. Additionally, considerations were made regarding the scalability and computational efficiency of the ensemble approach, particularly in large-scale deployments where processing overhead and resource constraints are pertinent concerns. Overall, the study underscored the potential of ensemble learning as a cornerstone in the arsenal of automated malware detection systems, offering a formidable defense against evolving cyber threats while paving the way for future research advancements in the field.



Input Android App Features

Upload your input CSV file

Drag and drop file here
Limit 200MB per file • CSV

Browse files

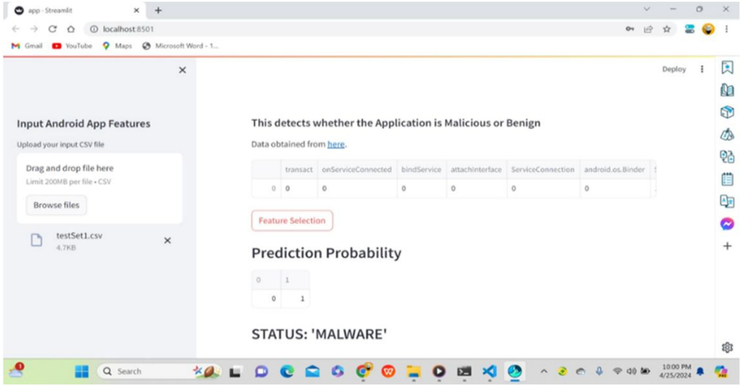
This detects whether the Application is Malicious or Benign

Data obtained from here.

Waiting for CSV file to be uploaded...

Fig. 2: Webpage

The webpage is shown in the above Figure 2, where the user need to feed in the test case using browse files options. Then the user need to click on feature selection button on the screen. Finally the output will be displayed on the screen as shown in Figure 3.



Input Android App Features

Upload your input CSV file

Drag and drop file here
Limit 200MB per file • CSV

Browse files

testSet1.csv
4.7KB

This detects whether the Application is Malicious or Benign

Data obtained from here.

	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder
0	0	0	0	0	0	0

Feature Selection

Prediction Probability

	0	1
0		1

STATUS: 'MALWARE'

Fig. 3: Output

If the malware is detected the status will be shown as “MALWARE” and if the malware is not detected the status will be shown as “BENIGN”.

V. CONCLUSION AND FUTURE WORK

In conclusion, the ensemble learning framework can adapt to evolving malware threats by incorporating new classifiers and updating the feature set dynamically . However there are still some limitations to address.

The performance of the ensemble may vary depending on the combination of base classifiers and the feature representation used. Further investigation is needed to explore different ensemble strategies and optimize their parameters for improved performance. Additionally, while the proposed approach achieved promising results across various malware families, there is always a possibility of encountering new, previously unseen threats.

Future enhancements in Android malware detection through machine learning could involve deep learning advancements tailored to Android app analysis, addressing adversarial attacks to bolster model robustness, and ensuring explain ability of AI-driven decisions for improved trust. Real-time detection capabilities could be augmented via online learning and distributed computing, while privacy-preserving techniques like federated learning could safeguard user data. Behavioural analysis in runtime could provide richer insights into malware activities. Extending detection to IoT devices and leveraging mobile edge computing would enhance comprehensive protection. Cross-platform compatibility efforts could broaden defence capabilities, and collaborative mechanisms would facilitate shared threat intelligence for proactive defence strategies. Ethical considerations regarding transparency and fairness in model deployment remain paramount.

REFERENCES

[1] Jayanthi, R. and Florence, L., 2019. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, 22(1), pp.77-88

[2] Felix, E.A. and Lee, S.P., 2017. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547.

[3] Wang, T., Zhang, Z., Jing, X., Zhang, L.: Multiple kernel ensemble learning for software defect prediction. Autom. Softw. Eng. 23, 569–590 (2015).

[4] Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, pp. 370–381

[5] Ryu, D., Baik, J.: Effective multi-objective naïve Bayes learning for cross-project defect prediction. Appl. Soft Comput. 49, 1062,

[6] Shan C., Chen B., Hu C., Xue J., Li N.: Software defect prediction model based on LLE and SVM. In: Proceedings of the Communications Security Conference (CSC ’14), pp. 1–5

[7] Yang, Z.R.: A novel radial basis function neural network for discriminant analysis. IEEE Trans. Neural Network. 17(3), 604–612

[8] K. Han, J.-H. Cao, S.-H. Chen, and W.-W. Liu, “A software reliability prediction method based on software development process,” in Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013 International Conference on. IEEE, 2013, pp. 280–283.

[9] F Gianfelici. Nearest-neighbor methods in learning and vision. IEEE Transactions on Neural Networks,19(2):377–377

[10] Pedro Domingos and Michael Pazzani. On The Optimality Of The Simple Bayesian Classifier Under Zero-One Loss. Machine Learning,29(2-3):103–130

[11] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and BS Manjunath. Malware Images: Visualization And Automatic Classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, page 4

[12] Mohit Sewak, Sanjay K Sahay, and Hemant Rathore. Comparison of deep learning and the classical machine learning algorithm for the malware detection. In 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pages 293–296.

[13] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil Bruce, Yang Wang, and Farkhund Iqbal. Malware classification with deepconvolutional neural networks. In 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pages 1–5.