ISSN 2277-2685
IJESR/April-June. 2025/ Vol-15/Issue-2s/753-759
B Keerthi *et. al.,* / International Journal of Engineering & Science Research

Andriod Malware Detection

Ms AVS Radhika[1], B Keerthi[2], N Keerthi[3]
[1]Assistant Professor, Department of CSE, Bhoj Reddy Engineering College for Women, India.
[2,3]B.Tech Students, Department of CSE, Bhoj Reddy Engineering College for Women, India.

Abstract
Malware is one of the major issues regarding the operating system or in the software world. The android system is also going through the same problems. We have seen other Signaturebased malware detection techniques were used to detect malware. But the techniques were not able to detect unknown malware. Despite numerous detection and analysis techniques are there, the detection accuracy of new malware is still a crucial issue. We propose Machine learning algorithms that will be used to analyse such malware and also we will be doing semantic analysis. We will be having a data set of permissions for malicious applications. This will be compared with the permissions extracted from the application which we want to analyse. In the end, the user will be able to see how much malicious permission is there in the application and also we analyse the application through comments.

Introduction

In today's fast-paced world, ensuring the security of Android devices has become increasingly important due to the growing number of malicious applications. Android's open-source nature and popularity make it a prime target for malware writers, posing significant threats to users by exploiting sensitive data or gaining unauthorized control over devices. To tackle this issue, the development of an Android Malware Detection System using genetic algorithm-based feature selection and machine learning has been undertaken as a mini project. This system aims to efficiently detect malware by combining advanced feature selection with optimized machine learning models, providing a reliable solution for zero-day malware detection.

Proposed System

The methodology involves reverse engineering two sets of Android Apps (Malware and Goodware) to extract features such as permissions and counts of app components (Activity, Services, Content Providers, etc.) from their AndroidManifest.xml files. These features, represented as a feature vector with class labels (0 for Malware and 1 for Goodware), are stored in CSV format. To reduce the dimensionality, a Genetic Algorithm is used to select the most optimized set of features. This optimized feature set is then used to train two machine learning classifiers: Support Vector Machine (SVM) and Neural Network (NN).

Literature Survey

[1] Drebin: A Lightweight and Explainable Malware Detection System (D. Arp et al., 2014) This paper introduces Drebin, a static analysis-based system for Android malware detection that does not require running the app. The system extracts key features such as app permissions and API calls from the APK file and uses a Support Vector Machine (SVM) for classification. Drebin operates directly on mobile devices, making it efficient and suitable for smartphones with limited computational resources. It also provides transparency by explaining the reasoning behind the detection of malicious apps. The study highlights Drebin's ability

to offer high accuracy while maintaining speed and clarity in detection, especially on mobile platforms.

[2] Comparative Study of Machine Learning Methods for Malware Classification (N. Milosevicetal.,2017).This research compares several machine learning algorithms, including Decision Trees, Naive Bayes, and SVM, to determine the most effective method for Android malware detection. The study focuses on using app features such as permissions and services. The results reveal that machine learning techniques outperform traditional signature-based approaches in terms of both accuracy and detection rates. The authors emphasize the importance of feature selection in improving model performance and the need to avoid overfitting to achieve better generalization. This work demonstrates the potential of machine learning to enhance Android app security.

[3] Permission-Based Feature Selection for Efficient Malware Detection (J. Li et al., 2018) This paper introduces a method for selecting the most significant permissions in Android apps to improve malware detection. Rather than using all available permissions, the study proposes focusing on those with the highest impact on classification performance. The results show that this selective approach not only enhances detection accuracy but also reduces computational complexity. The authors validate their method through experiments with various machine learning models, demonstrating that permission-based feature selection leads to faster and more efficient malware detection without sacrificing accuracy.

[4] MADAM: Real-Time Behavior-Based Malware Detection (A. Saracino et al., 2018) MADAM is a hybrid malware detection system that monitors both user and app behavior in real-time to identify malicious activity. Unlike cloud-based systems, MADAM operates directly on Android devices, providing an efficient and resource-conserving solution. The system analyzes interactions between the user and the app, as well as internal app operations, to detect potential threats. MADAM is designed to minimize false positives, making it highly effective for mobile deployment. This approach demonstrates a lightweight, real-time detection system that is ideal for resource-constrained mobile environments.

[5] SAMADroid: A 3-Level Hybrid Detection Framework (S. Arshad et al., 2018) SAMADroid is a hybrid detection framework that integrates static analysis, dynamic analysis, and signature-based methods to provide comprehensive Android malware detection. The three-layered approach leverages the strengths of each technique, improving detection accuracy and reducing the number of false alarms. The system is tested on a wide range of Android applications and consistently outperforms standalone methods. SAMADroid's integrated framework enhances its ability to detect a broader spectrum of malware types, making it a robust solution for Android security. The study highlights SAMADroid's superior performance compared to traditional detection methods.

Methodology

The Android malware detection system is developed using a data-driven and modular approach that ensures high detection accuracy through feature-rich datasets and machine learning techniques. The methodology is divided into the following key components:

System Architecture

The system is designed using a layered architecture:

- **DataLayer**:
  Utilizes a labeled dataset containing both benign and malicious Android

applications. Each application includes static features like permissions, API calls, intent filters, and components extracted from the APK files.

- **Processing Layer**:
  - **Data Preprocessing**: Involves removing redundant entries, handling missing values, encoding categorical features, and scaling numerical data.
  - **Feature Selection**: Identifies the most relevant features (e.g., suspicious permissions or API calls) using correlation analysis or chi-square tests to improve model performance and reduce overfitting.
- **Modeling Layer**:
  - Implemented using Python libraries such as Pandas, NumPy, Scikit-learn, and Matplotlib.
  - Multiple classifiers are applied, including Support Vector Machine (SVM), Decision Tree, Random Forest, and Neural Networks, for comparative analysis.
- **Evaluation & Output Layer**:
  - Models are evaluated based on metrics like accuracy, precision, recall, F1-score, and confusion matrix.
  - Visualizations such as ROC curves, bar graphs of feature importance, and confusion matrices are generated for interpretability.

Workflow

1. **Data Acquisition**:
   - Collect APK files from reliable datasets such as Drebin or VirusShare.
   - Extract static features using reverse engineering tools (e.g., Androguard or Apktool).
2. **Data Preprocessing**:
   - Remove noise and duplicates.
   - Encode permissions and features as binary vectors. Normalize numerical values if any.
3. **Feature Selection**:
   - Use correlation or chi-square analysis to retain only important features.
   - Drop irrelevant or redundant attributes to reduce computation time and improve model clarity.
4. **Model Training**:
   - Split the dataset into training and test sets (commonly 80:20).
   - Train the selected machine learning algorithms on the training data.
5. **Model Testing & Evaluation**:
   - Test the trained models on the unseen data from the test set.
   - Evaluate using metrics like accuracy, recall (important in malware detection), precision, and F1-score.
6. **Detection Output**:
   - Classify new applications as either ‘Malware’ or ‘Benign’ based on extracted features.
   - Display results along with the model’s prediction confidence and performance scores.

7. **Result Interpretation**:
    - Use visual tools like bar graphs (for feature importance), confusion matrix heatmaps, and ROC curves.
    - Provide insights into which features most influence the classification decision.

Results

Fig 1 Run the python code (app.py) from the terminal

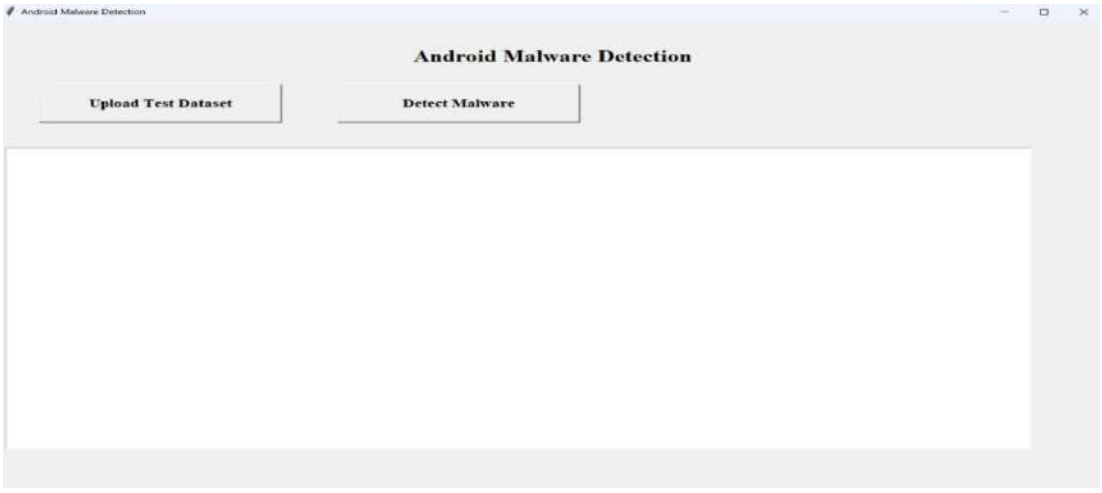

Fig 2 Home Page

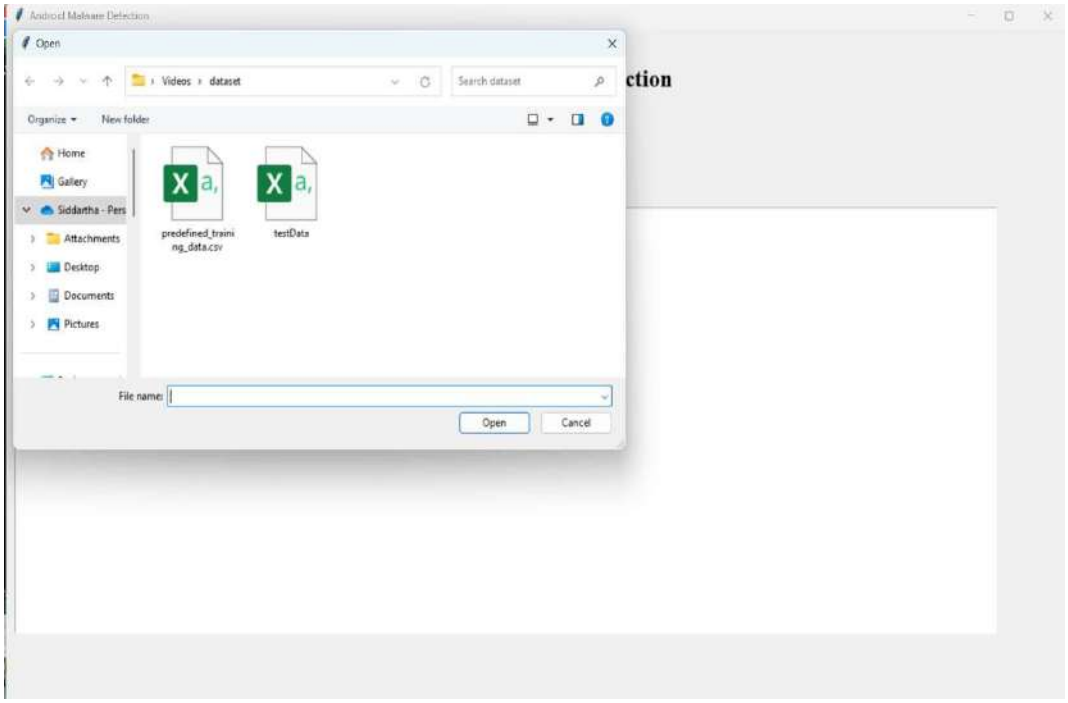

Fig 3 pload the Dataset

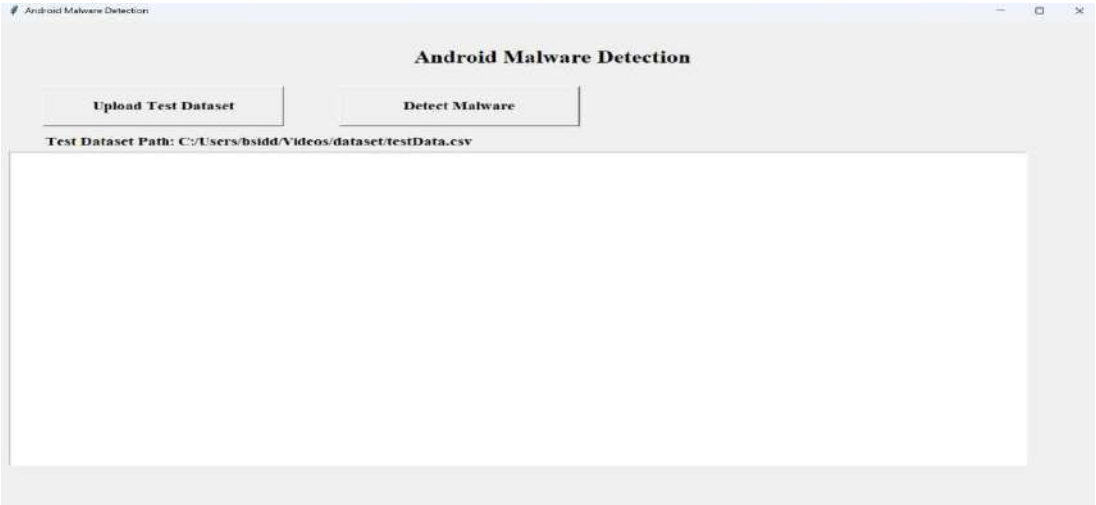

Fig 4 Dataset is loaded

Fig 5 Click on Detect Malware & Predict the result Result Page showing Model Accuracy Results

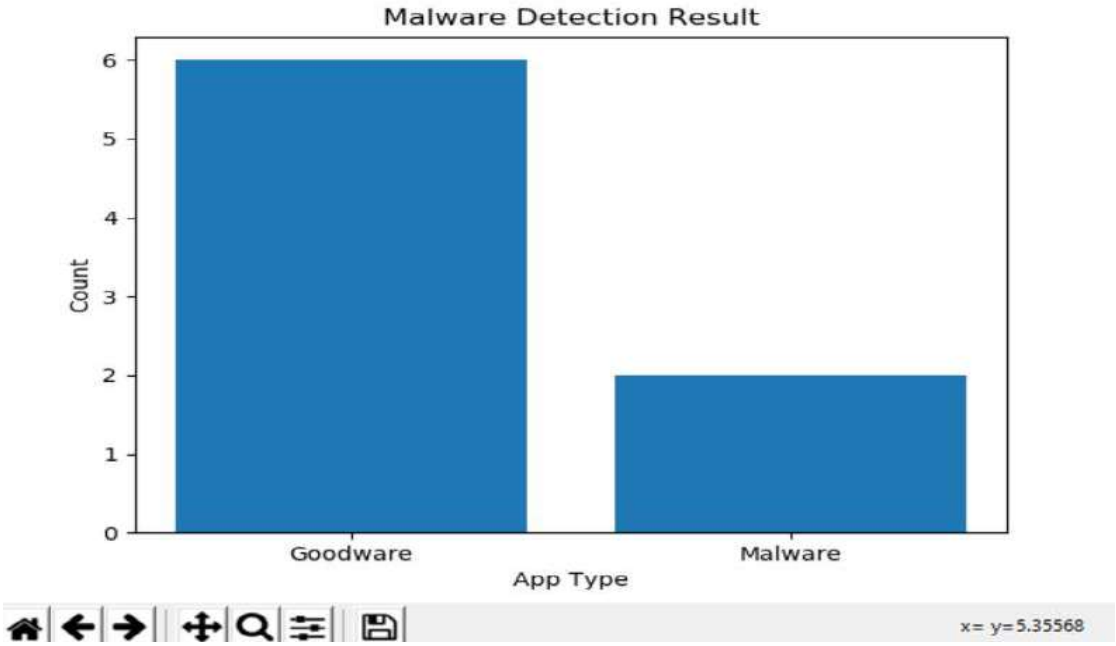

Fig 6 Result Page showing Graphical Representation

Conclusion And Future Scope

Conclusion

The project titled "Android Malware Detection Using Genetic Algorithm-Based Optimized Feature Selection and Machine Learning" provides a robust approach to addressing malware threats in the Android ecosystem. By combining static analysis with machine learning classifiers, such as

Support Vector Machines (SVM) and Neural Networks (NN), the project improves detection accuracy. The use of Genetic Algorithms (GA) for feature selection enhances the efficiency of these classifiers, reducing dimensionality and enabling better performance. With its user-friendly GUI and integration of advanced algorithms, the system demonstrates significant promise in detecting zero-day threats and protecting users against malicious applications.

Future Scope

Adding interactive visualizations, real-time logs, and customizable settings. Expanding to iOS malware detection, using advanced AI techniques like federated learning and explainable AI, and enabling real-time detection will enhance functionality. Features like malware removal suggestions, notifications, and integration with threat intelligence improve usability. Blockchain, zero-day detection, edge computing, and industry collaboration ensure scalability, security, and innovation.

REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hübner, . Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Proceedings 2014 Network and Distributed System Security Symposium, 2014

[2] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," Comput. Electr. Eng., vol. 61, pp. 266–274, 2017.

[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," IEEE Trans. Ind. Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] A.Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Trans. Dependable Secur . Comput., vol. 15, no. 1, pp. 83–97, 2018.

[5] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," IEEE Access, vol. 6, pp. 4321–4339, 2018.