Permission Based Detection Of Android Malware

Yasaswini Muthineni
Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
Bapatla, India
yasaswinitechmail@gmail.com

Anupama Nadendla
Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
Bapatla, India
anupamanadendla123@gmail.com

Sreevani Muntha
Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
Bapatla, India
munthasreevani1@gmail.com

Poojitha Kuchipudi
Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
Bapatla, India
kuchipudipoojitha2003@gmail.com

***Abstract*— Android malware poses a significant threat to user privacy and security. This paper presents a permission-based detection approach aimed at identifying malware by analyzing the permissions requested by Android applications. We conducted experiments using a dataset of Android apps to evaluate the effectiveness of our approach. Results show promising accuracy rates, highlighting the potential of permission-based detection in combating Android Malware. This paper proposed to use different techniques and classification algorithm like Support Vector Classifier (SVC) and Artificial Neural Network (ANN) in which feature set is optimized using Genetic Algorithm (GA)whether a application is Malicious or Benign, by uploading a APK file. We have experimented on Dataset containing 1800 samples performs great for this classification task**

Keywords—Malware, Benign, Permissions, Genetic Algorithm, Neural Network, Support Vector Classifier.

I. INTRODUCTION

With the widespread adoption of mobile devices and the increasing reliance on mobile applications for various tasks, the security of the Android ecosystem has become a paramount concern. Malicious actors continually devise sophisticated techniques to exploit vulnerabilities and compromise users' devices and data. In response to this growing threat landscape, there is a pressing need for advanced and proactive approaches to detect and mitigate Android malware.

This project presents a novel solution for permission-based detection of Android malware by leveraging Artificial Neural Networks (ANN) and Support Vector Classification (SVC) algorithms, optimized using Genetic Algorithms (GA). Unlike traditional signature-based methods, which rely on known patterns of malware, our approach focuses on analyzing the permissions requested by Android applications (APK files) to determine their potential maliciousness.

The primary objective of this project is to develop a user-friendly system where users can upload APK files, and our trained models predict the likelihood of the uploaded app being malicious based on its requested permissions. By extracting and analyzing the permission set of each APK file, our system provides a proactive and insightful assessment of app security, empowering users to make informed decisions about the apps they install.

Figure I.1 our model is designed to predict the presence of malware.

Key features of our project include:

Advanced Machine Learning Techniques: We employ ANN and SVC algorithms, renowned for their ability to learn complex patterns from data, to classify Android apps based on their permission sets. These models are trained and optimized using Genetic Algorithms to achieve high accuracy in malware detection.

Permission-Based Analysis: By focusing on app permissions, our system provides a granular understanding of the behavior and intentions of Android applications. This permission-centric approach allows us to identify potential security risks and malicious activities more effectively.

User-Centric Interface: Our platform is designed with user convenience and transparency in mind. Users can easily upload APK files through a user-friendly interface and receive clear and comprehensible results indicating the likelihood of the uploaded app being malicious.

Continuous Learning and Adaptation: The system is designed to adapt and evolve alongside emerging threats. As new malware variants emerge, our models can be updated and retrained to ensure robust protection against evolving security risks.

Our project aims to provide a proactive and user-friendly solution for detecting Android malware based on app

permissions. By harnessing the power of machine learning and genetic algorithms, we strive to enhance the security of the Android ecosystem and empower users to safeguard their devices and data effectively.

II. Related Work

This section contains the literature review of the previous research. Here, we provide the relevant information of the previous analysis, research reports and relevant articles.

We examine the existing malicious App detection schemes. Nisha et al. [1] proposed the detection of repackaged Android malware using mutual information and chi-square techniques for the selection of features. Random forest classifier was able to achieve the highest accuracy of 91.76% among employed classifiers. However, Nisha et al.'s technique is focused on the 88 uniquely identified permissions for the analysis, which can be further optimized to include only the harmful ones.

Similarly, Sandeep [2] extracted information from the applications and performed Exploratory Data Analysis (EDA). The EDA approach focuses on the detection of malware using deep learning techniques during the installation process. Sandeep's detection framework employed several options like permissions to mirror the behaviors of the applications. It achieved 94.6% accuracy when Random Forest was used as the classifier. The approach uses 331 features for the classification which can further be optimized.

Li et al. [3] suggested a permission-based detection system called SIGPID on the permission usage analysis.
Li et al. employed a multi-level data pruning technique for the selection of features. Using three levels of pruning—Permission Ranking with Negative Rate (PRNR) Support Based Permission Ranking (SPR), and Permission Mining with Association Rules (PMAR)—they could identify 22 significant permissions.

Similarly, Wang et al. [4] performed a Multilevel Permission Extraction (MPE) approach where they focused on automatically identifying the permission interaction that helps to distinguish between the benign and the malicious applications. Their dataset included 9736 applications from each category set—benign and malicious—and experimental results show that a detection rate of 97.88% was achieved.

Similarly, Sun et al. [5] used static analysis in the study and proposed a collection of program characteristics consisting of sensitive API calls and permissions and performed evaluations using the Extreme Learning Machine (ELM) technique. Sun et al.'s aim consists of detection which involves minimal human intervention. Sun et al. implemented an automated tool called WaffleDetector. The proposed tool showed ~97.14% accuracy using the ELM technique.

However, the results are based on a small dataset of only 1049 applications from third-party or unofficial stores. Zhu et al. [6] proposed a low-cost system for malware detection by extracting a set of system events, permissions, and sensitive APIs and calculating the permission rate according to the key features. Zhu et al.'s approach employed the ensemble Rotation Forest (RF) to construct a model for the classification of malicious and benign APKs.

The approach Symmetry 2022, 14, 718 5 of 19 yielded over 88% detection accuracy which is almost 3.3% higher when compared with SVM classifier. However, Zhu et al.'s technique is evaluated on the limited APKs dataset. Further, the proposed feature set can be optimized and evaluated with other ML classifiers to improve detection accuracy.

III. PROPOSED METHODOLOGY

Permission-based analysis focuses on scrutinizing the permissions requested by Android applications as a primary indicator of potential malicious behaviour. By analysing the permissions requested by an app, such as access to sensitive data or device resources, security systems can identify suspicious or potentially harmful applications.

We proposed a system which will *predict the presence of malware using Artificial Neural Network and Support V ector Classifier which are optimised using Genetic Algorithm.*

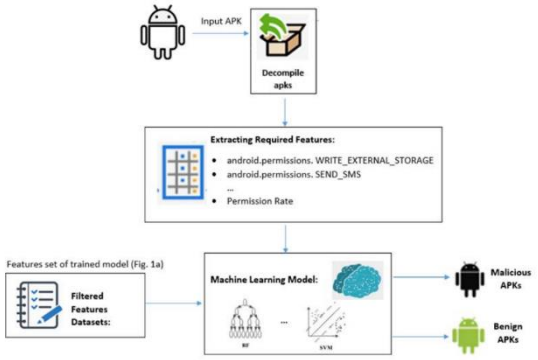

Figure III.1 Architecture of Proposed system.

A. *Process for implementation*

1. Model is trained with new dataset, which include more features for malware detection.

2.The dataset includes permissions as features. All the records have class labels as either malware or benign.

3.After, The Feature set optimized using Genetic Algorithm.

4.Then, selected models are trained using optimized feature set and evaluated.

5.Finally, stakeholders can upload their APK files.

6.The model extracts the permissions from APK file uploaded by user, compare those permissions with the permissions it gets trained and detect whether it is malware or safe application.

B. *Interface*

To create a user interactive web page, we used a framework called flask.

Flask is a lightweight web framework for Python that makes it easy to build web applications. It simplifies tasks like routing URLs to functions, handling HTTP requests and responses, and rendering HTML templates. Flask is known for its simplicity, flexibility, and ease of use, making it a popular choice for developing web applications, APIs, and microservices. It provides you with all the tools you need to create different parts of your website, like pages, forms, and buttons.

C. Algorithms

The model designed for the detection of Android malware based on the permissions requested by applications. Leveraging the power of Artificial Neural Networks (ANN), Support Vector Machines (SVC), and Genetic Algorithm optimization, the model aims to provide accurate and efficient malware detection capabilities, thereby enhancing the security of Android devices.

1) Genetic Algorithm:

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. They simulate "survival of the fittest" among individuals of consecutive generations to solve a problem. Each generation consists of a population of individuals and each individual represents a point in search space and possible solution.

Following is the foundation of GAs based on this analogy –

1.Genetic algorithm starts with a random population of candidate solutions.

2.The solutions are evaluated based on their fitness, and the best ones are selected for reproduction.

3.The genes of the selected solutions are combined through crossover to create new solutions.

4.The genes of the new solutions are randomly mutated to maintain diversity.

5.The process is repeated until a stopping criterion is met, such as a maximum number of generations or a satisfactory fitness level.

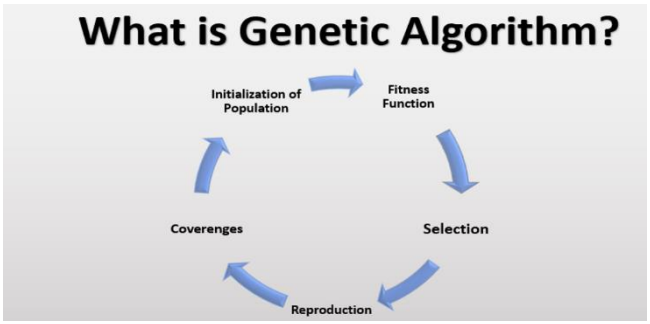

Figure III.2 genetic algorithm cycle process.

2) Support Vector Classifier

Support Vector Classifier (SVC) is a supervised machine learning algorithm used primarily for classification tasks. The key objective of SVC is to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. The margin is defined as the distance between the hyperplane and the support vectors.

SVC aims to maximize this margin. By maximizing the margin, SVC not only separates the classes but also improves the model's ability to classify unseen data accurately. SVC can efficiently handle non-linearly separable data by using the kernel trick. The kernel trick implicitly maps the input data into a higher-dimensional space where it becomes linearly separable. SVC introduces a regularization parameter (C) to balance between maximizing the margin and minimizing the classification error.

Overall, SVC is a versatile and powerful algorithm for classification tasks, capable of handling both linear and non-linear data with appropriate kernel selection and parameter tuning.

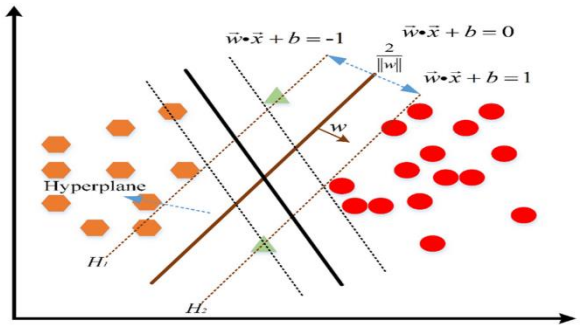

Figure III.3 support vector classifier.

3) Artificial Neural Network

An Artificial Neural Network (ANN) Sequential Model is a type of neural network architecture used in machine learning and deep learning for various tasks such as classification, regression, and sequence prediction. The sequential model is a linear stack of layers where each layer is densely connected to the next one. Let's break down the components and workings of an ANN sequential model.

Each layer in the network feeds its output to the next layer, forming a sequential flow of information. each layer in the network feeds its output to the next layer, forming a sequential flow of information. Activation functions are applied to the outputs of each layer to introduce non-linearity into the model, allowing it to learn complex patterns in the data. It needs to be compiled before training. Once compiled, the sequential model can be trained on a dataset. Once trained and evaluated, the model can be used to make predictions on new, unseen data.

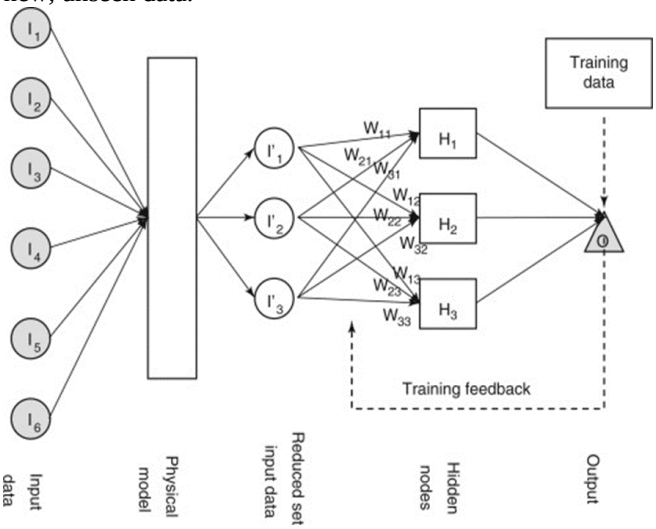

Figure III.4 Illustrates ANN working.

D. Dataset

The model is trained with labelled binary dataset where permissions are considered as features. If a particular permission is allowed, then the value is to 1 or else the value is set to 0. The dataset is downloaded from GitHub repository of a developer in which there are up to 400 features with 1700

distinguish values to those permissions. The target value or the class label of the dataset is either malware or benign.

Final > app > static > permissions.txt

```
android.permission.ACCESS_ALL_DOWNLOADS
android.permission.ACCESS_BLUETOOTH_SHARE
android.permission.ACCESS_CACHE_FILESYSTEM
android.permission.ACCESS_CHECKIN_PROPERTIES
android.permission.ACCESS_CONTENT_PROVIDERS_EXTERNALLY
android.permission.ACCESS_DOWNLOAD_MANAGER
android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED
android.permission.ACCESS_DRM_CERTIFICATES
android.permission.ACCESS_EPHEMERAL_APPS
android.permission.ACCESS_FM_RADIO
android.permission.ACCESS_INPUT_FLINGER
android.permission.ACCESS_KEYGUARD_SECURE_STORAGE
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
android.permission.ACCESS_MOCK_LOCATION
android.permission.ACCESS_MTP
android.permission.ACCESS_NETWORK_CONDITIONS
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_NOTIFICATIONS
android.permission.ACCESS_NOTIFICATION_POLICY
android.permission.ACCESS_PDB_STATE
android.permission.ACCESS_SURFACE_FLINGER
android.permission.ACCESS_VOICE_INTERACTION_SERVICE
android.permission.ACCESS_VR_MANAGER
```

Figure III.5 Some Dataset features**.**

E. WorkFlow

- Data Collection**:** Gather data from various sources which includes various permissions that can predict presence of malware.
- Preprocessing Data: Handle missing values-Impute missing data or remove incomplete records. Also splitting the data.
- Feature Selection: The considered dataset has upto 400 features, so it is necessary to optimised the feature set.in this project we considered Genetic Algorithm to optimise.
- Model Selection: Choosing the most suitable algorithm for predicting presence of malware. In this project we have selected ANN and SVC.
- Training the Model: Feed the model with optimised dataset for training.
- Testing the model: After training with dataset, model is ready to make predictions. Model is input with many APK files for testing.
- Evaluation: Various evaluation metrics are calculated like accuracy, recall, precession.
- Deployment: Deploy the selected model as a web application using frameworks such as Flask.

IV. EVALUATION

You can evaluate the model using various performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score. These metrics help quantify the accuracy and precision of the model's predictions.

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted class label and the actual class label. A lower MAE indicates better accuracy.

Mean Squared Error (MSE): MSE measures the average squared difference between the predicted chances and the actual chances. It penalizes large errors more than MAE.

Root Mean Squared Error (RMSE): RMSE is the square root of the MSE and provides an interpretable measure of the average prediction error. It is in the same units as the target variable and is easier to interpret than MSE.

R-squared (R^2) Score: R-squared measures the proportion of the variance in the target variable that is explained by the independent variables in the model. A higher R-squared value indicates a better fit of the model to the data.

Cross-Validation: It's essential to perform cross-validation to assess the model's generalization performance. This involves splitting the dataset into training and testing subsets multiple times and evaluating the model on each split to ensure that it performs consistently across different subsets of the data.

V. RESULT

The starting page of the application displays when application is executed on Editor Terminal, the application is hosted on a web and the below page is opened on the browser.
It is designed in such a way that the user can upload his APK file and choose either neural network or SVC to predict the result.

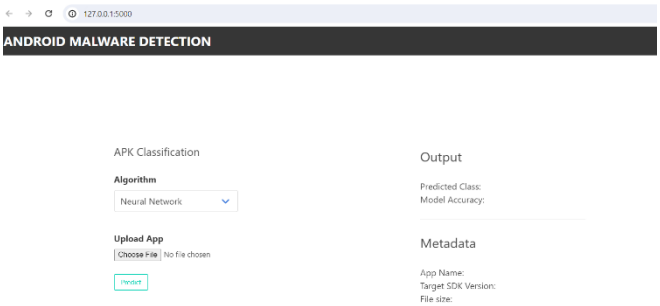

Figure V.1 Home page.

When user uploaded APK file and choose Neural Network to predict, if the input APK file is malicious, then result is displayed as below.

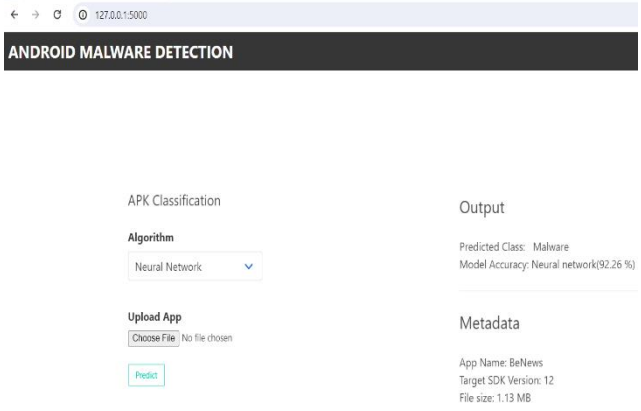

Figure V.2 output when malicious APK file is input.

VI. CONCLUSION

In conclusion, the project successfully demonstrates the efficacy of employing permission-based detection of Android malware, leveraging advanced algorithms such as Neural Networks and Support Vector Classification (SVC). By optimizing the feature set through Genetic Algorithms, the system achieves enhanced accuracy and robustness in identifying malicious apps.

ACKNOWLEDGMENT

The authors would like to express our sincere gratitude to Dr. Nagalla Sudhakar for guidance, support throughout the duration of this project. Thank you to all the research paper authors whose work contributed to the success of our project.

REFERENCES

[1] Jannath, N.O.S.; Bhanu, S.M.S. Detection of repackaged Android applications based on Apps Permissions. In Proceedings of the 2018 4th International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, India, 15–17 March 2018; pp. 1–8.

[2] Sandeep, H.R. Static analysis of android malware detection using deep learning. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Secunderabad, India, 15–17 May 2019; pp. 841–845.

[3] Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-An, W.; Ye, H. Significant Permission Identification for Machine-Learning-Based Android Malware Detection. IEEE Trans. Ind. Inform. 2018, 14, 3216–3225.

[4] Wang, Z.; Li, K.; Hu, Y.; Fukuda, A.; Kong, W. Multilevel permission extraction in android applications for malware detection. In Proceedings of the 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 28–31 August 2019; pp. 1–5.

[5] Sun, Y.; Xie, Y.; Qiu, Z.; Pan, Y.; Weng, J.; Guo, S. Detecting android malware based on extreme learning machine. In Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress Orlando, FL, USA, 6–10 November 2017; pp. 47–53.

[6] Zhu, H.-J.; You, Z.-H.; Zhu, Z.-X.; Shi, W.-L.; Chen, X.; Cheng, L. DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing 2018, 272, 638–646.

[7] Rawat, Nishant & Singh, Avjeet & Amrita. (2023). Permission-Based Malware Detection in Android Using Machine Learning. Ymer. 22(2023). 1505-1517. 10.37896/YMER22.03/C4.

[8] Ehsan A, Catal C, Mishra A. Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review. Sensors (Basel). 2022 Oct 18;22(20):7928. doi: 10.3390/s22207928. PMID: 36298282; PMCID: PMC9609682.

[9] Kshirsagar, D., & Agrawal, P. (2022). A study of feature selection methods for android malware detection. Journal of Information and Optimization Sciences, 43(8), 2111–2120. https://doi.org/10.1080/02522667.2022.2133218.