IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, April 2025
Impact Factor: 7.67

Android Malware Detection using ML

Prof. Mangala Patil[1], Poornima[2], Pradhipa S[1], Indu A[3], Reena K[4]
Assistant Professor, Department of Computer Science and Engineering[1]
Students, Department of Computer Science and Engineering[2,3,4,5]
Impact College of Engineering and Applied Sciences (ICEAS), Bengaluru, Karnataka, India

Abstract*: Android devices are more prone of malware attacks due to its open-source nature. This makes it easier for installing applications from various sources, which can lead to major security issues. Machine learning learns from examples. It studies data from apps both good and bad and understands its characteristics. Using Machine Learning in this case can help identify harmful malware installed in android devices. Detecting unknown malicious code by applying classification techniques on Opcode patterns, checks the use of Opcode n-gram patterns from disassembled files. Previously studies have proved that utilizing 2-gram of Opcode patterns, term frequency representation, and selecting 200 plus features based on document frequency gives higher performance in detecting unknown malware. In traditional method s this was considered a complex task. This project is focused on addressing the challenges of detecting the Android malware using a machine learning based approach, which improves upon traditional signature-based methods. By focusing on feature-based inputs rather than requiring users to upload APK files, this system enhances both privacy and usability*

Keywords: Data collection, De-compilation, Feature extraction, Classification algorithms, Malware detection model

I. INTRODUCTION

The cybersquatting landscape has evolved significantly, with cybercafe becoming asophisticated and pervasive threat. Despite advancements in security measures, attacks on computer systems have increased in both frequency and complexity. Traditional detection methods, such as manual analysis and signature-based approaches, often struggle to keep pace with the rapid development of new malware variants, especially those employing obfuscation techniques. The proliferation of mobile devices, particularly those running the Android operating system, has introduced new security challenges. Android's open-source nature makes it susceptible to exploitation, leading to a surge in malware targeting these devices. Malicious activities include unauthorized SMS sending, data theft, and remote control, posing significant risks to user privacy and security. To address these challenges, researchers have increasingly turned to machine learning and deep learning techniques for malware detection. By analyzing both static features (code structure) and dynamic behaviors (runtime activities), these models can identify malicious patterns more effectively. For instance, hybrid deep learning models combining Deep Belief Networks (DBN) and Gated Recurrent Units (GRU) have shown promise in detecting Android malware, even those employing advanced obfuscation methods.

II. LITERATURE SURVEY

1.Title: Android malware Detection using Machine Learning: A Review
Authors: Md Naseef-Ur-Rahman Chowdhury, Ahshanul Haque
Publisher: April 2023
Malware for Android is becoming increasingly dangerous to the safety of mobile devices and the data they hold. Although machine learning techniques have been shown to be effective at detecting malware for Android, a comprehensive analysis of the methods used is required.

2. Title: A Systematic Overview of Android Malware Detection.
Authors: Li Meijin, Fang Zhiyang, Wang Junfeng, Cheng Luyu, Zeng Qi.

Publisher: December 2021.
The open-source nature of the Android operating system has led to an increase invulnerabilities and malware attacks. This paper provides a comprehensive review of Android malware detection, emphasizing featureselection algorithms and highlighting areas often overlooked in previous studies, such as limitations and commonly used datasets in machine learning-based models.

3. Title: Malware detection in android mobile platform using machine learning algorithms
Authors: Mariam Al Ali, Suryani Lukman.
Publisher: February 2018
Malware has always been a problem in regard to any technological advances in the software world. Thus, it is to be expected that smart phones and other mobile devices are facing the same issues. In this paper, a practical and effective anomaly-based malware detection framework is proposed with an emphasis on Android mobile computing platform.

4. Title:A Survey on Detecting Android Malware Using Machine Learning Technique
Authors: S Priyadharshini, S Shanthi
Publisher: March 2024
Nowadays, internet connected smart phones devices usage are increasing steadily andthe growth of Android application users are increasing. Mobile devices are used in daily activities such as remote light control, goods and parking payment with the information of saved credit card details. Due to the growth of Android application users, some intruders are creating malicious android applications as a tool to steal sensitive data and identity theft / fraud.

5. Title: Detection of Android Malware Using Machine Learning and Siamese Shot Learning
Technique for Security
Authors: Fahdah A. Almarshad, Mohammed Zakariah
Publisher: November 2023.
Android malware security tools that can swiftly identify and categorize various malware classes to create rapid response strategies have been trendy in recent years. In this paper, a one-shot learning based Siamese neural network is proposed to overcome this issue.

III. PROPOSED SYSTEM

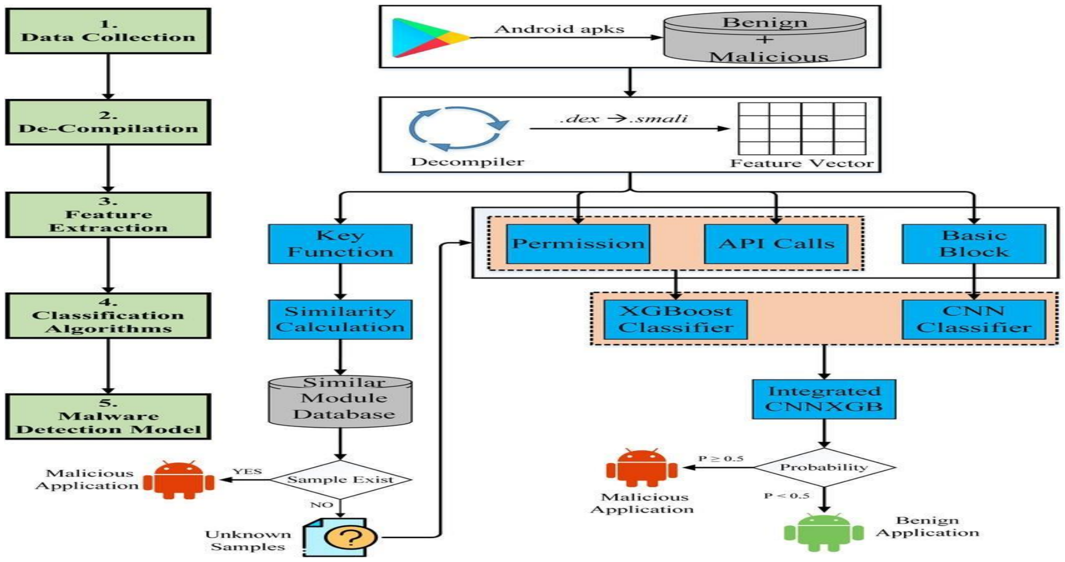


Figure: Proposed system

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, April 2025
Impact Factor: 7.67

A Proposed System for malware detection leverages machine learning to enhance accuracy and adaptability. This system begins by extracting a comprehensive set of features from Android applications, including permissions, API calls, and other relevant metadata. These features are then used to train various machine learning classifiers, such as Support Vector Machines (SVM), Decision Trees (DT), Logistic Regression (LR), Gradient Boosting (GB), and Random Forests (RF), to distinguish between benign and malicious applications. The performance of these classifiers is evaluated using datasets comprising both benign andmalicious Android applications. By employing this approach, the system aims to effectively identify and mitigate threats posed by malicious software on Android devices.

IV. PROBLEM STATEMENT

In the realm of malware detection, two notable studies have explored innovative approaches to identify malicious software. The first study, "Detecting Unknown Malicious Code by Applying Classification Techniques on Opcode Patterns," investigates the use of Opcode n-gram patterns extracted from disassembled files. The researchers conducted extensive experiments with various n-gram settings and feature representations. They discovered thatutilizing 2-gram Opcode patterns, term frequency (TF) representation, and selecting 300 features based on document frequency (DF) yielded superior performance in detecting unknown malware. This approach addresses limitations in traditional machine learning techniques by focusing on the structural aspects of code. The second study, "Detecting Scareware by Mining Variable Length Instruction Sequences," presents a static analysis method that extends general heuristic detection techniques. The researchers applied data mining to variable-length instruction sequences extracted from binary files to detect scareware. While this method shows promise, it does not incorporate specific metrics or unsupervised techniques, which could potentially enhance its robustness and effectiveness.

V. OBJECTIVES

Building the Model: The heart of this project lies in developing a machine learning model that can sniff out malware hiding in Android apps. Instead of relying on traditional methods like signature-based detection, this approach will use specific app features—like permissions and API usageto identify malicious behavior. By training the model with diverse, well-prepped data and testing different algorithms, you'll pinpoint the one that gets the job done most accurately. Think of it as teaching a detective how to spot the culprit using the most crucial clues. **Creating the Interface**: Imagine a sleek web interface where users can simply input app features and instantly get a verdict on whether the app is benign or malicious. The design will be intuitive and easy to use, like having a friendly assistant who guides you through each step. With technologies like Flask or Django on the back end and a polished front-end with HTML and JavaScript, this tool will seamlessly connect users to the power of machine learningreal-time results, no hassle.**P**
roving the System Works: To show off just how smart this model is, you'll put it to the test with some real-world data. By comparing its predictions against known outcomes, you can prove how well it performs using measures like accuracy and recall. Visual aids, like graphs and charts, will make your results pop, showing everyone that machine learning can outshine older methods when it comes to malware detection.

VI. METHODOLOGY

Data Collection: We gather a dataset that includes both benign and malware samples. Each sample has various features like permissions, network requests, and API calls, which are commonly used in determining app behavior. The availability of large volumes of structured and unstructured data allowed practical applications of ML to surge in recent years. Clearly outline the objectives of the data collection process and the specific research questions I want to answer.

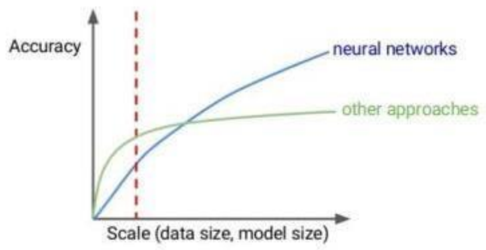


Figure: Data collection

Feature engineering: Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

Types of Feature Creation:

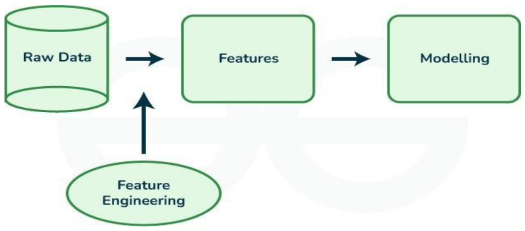


Figure: Feature engineering

Domain-Specific: Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.

Data-Driven: Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.

Synthetic: Generating new features by combining existing features or synthesizing new data points. The most relevant features for detecting malware are selected. These may include:

○ **Permission Usage**: Types of permissions requested by the application, such as location, camera, and storage.

○ **API Call Frequency**: Counts or types of sensitive API calls that could indicate malicious behavior.

○ **Network Activity*:*** Frequency or pattern of network requests, such as frequent access to remote servers.

Model Selection and Training: We use a machine learning algorithm (such as Random Forest or Support Vector Machine) that can classify malware based on the selected features. This model is trained on a labeled dataset with known malware and benign samples.

System Workflow:

User Input: The user inputs key feature values (e.g., permission count, API call indicators) into the web interface.

Model Prediction: The back-end Python model processes these inputs and returns a result indicating whether the app is likely "Malware" or "Benign."

Evaluation and Optimization: The model's accuracy is tested using various metrics such as precision, recall, and F1-score. Based on these metrics, the model is optimized to improve detection rates while reducing false positives.

VII. IMPLEMENTATION

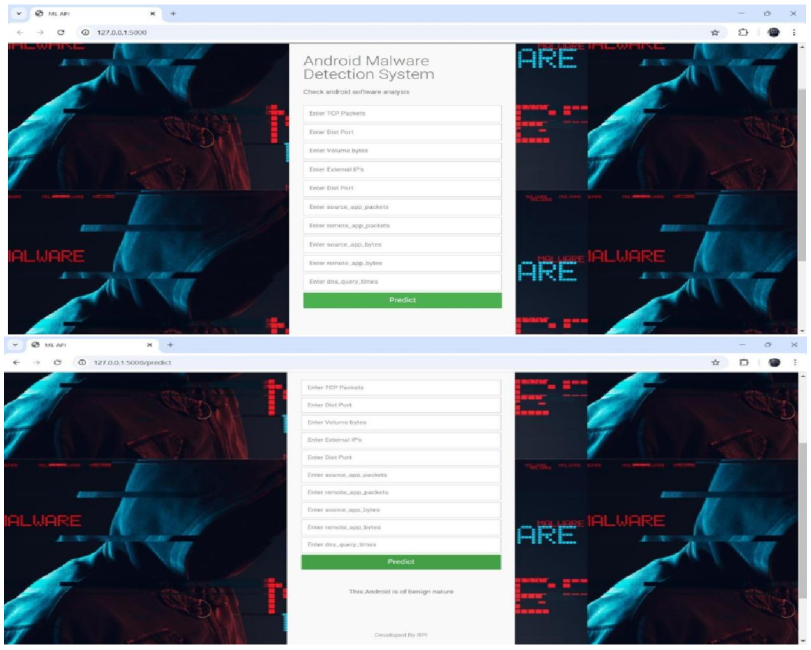
1. FRONTEND:

The front-end of the application is designed with simplicity in mind, utilizing HTML and CSS to create a user-friendly interface. It features a set of input fields where users can easily enter key information related to various features such as permissions and API usage. These fields allow users to specify who has access to certain resources and how different API s are being utilized within the system. The structure is built using HTML, which defines the form elements, including text fields, dropdown menus, and buttons. CSS is then applied to style these elements, ensuring the layout is clean and visually appealing

2. BACKEND:
The back end of the application, built using Python, is responsible for processing the input data received from the front-end. Once the user submits their entries, the back end captures this information and passes it through a trained machine learning model. The model has been previously trained on relevant data to recognize patterns and make predictions based on the input. After processing the data, the model generates a binary classification result, which typically indicates one of two outcomes, such as "approved" or "denied," "true" or "false," or any other applicable classification depending on the specific use case.

TOOLS AND TECHNOLOGIES
Machine Learning Model: For this project, a Random Forest or Decision Tree model was selected due to its efficiency in handling complex and high-dimensional feature data, making it well-suited for classifying Android applications as "Malware" or "Benign." These models are particularly advantageous for malware detection because they excel in recognizing intricate patterns and relationships among input features, such as permissions requested, network activity, and API usage—all key indicators of malicious behavior. The model was trained on a labeled data set thatincluded a diverse collection of Android applications. During training, various features were analyzed to help the model understand patterns typically associated with malware, such as specific combinations of permission requests and network calls.
Decision tree: It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves. In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute.
Random Forest: A random forest is a machine learning technique that's used to solve regression andclassification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms
XG Boost: XG Boot or extreme gradient boosting is one of the well-known gradients boosting techniques(ensemble) having enhanced performance and speed in tree-based (sequential decision trees) machine learning algorithms. XG Boost was created by Tianqi Chen and initially maintained by the Distributed (Deep) Machine Learning Community (DMLC) group. It is the most common algorithm used for applied machine learning in competitions and has gained popularity through winning solutions in structured and tabular data. XG Boost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XG Boost is an implementation of gradient boosted decision trees designed for speed and performance.
SVM: Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms,which is used for Classification as well as Regression problems. However, primarily, it isused for Classification problems in Machine Learning. The goal of the SVM algorithm is tocreate the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence algorithm is termed as Support Vector Machine.

VIII. RESULT ANALYSES



Android Malware Detection System

Check android software analysis

Enter TCP Packets
Enter Dist Port
Enter Volume bytes
Enter External IP's
Enter Dist Port
Enter source_app_packets
Enter remote_app_packets
Enter source_app_bytes
Enter remote_app_bytes
Enter dns_query_times

This Android is of benign nature

Figure: Predicted output.

Detecting Android malware with machine learning is like teaching a guard dog to recognize intrudersit takes precision, practice, and the right tools. The heart of the process lies in ensuring the model accurately identifies malicious apps without falsely flagging safe ones. We measure success using metrics like accuracy, precision, recall, and F1-score, which tell us how well the machine is performing.Features such as app permissions, API calls, or unusual behaviors are like clues that help the model differentiate between harmless and harmful apps. Extracting and analyzing these clues is a crucial part of the process

IX. CONCLUSION

Any smartphone which uses android applications is potentially vulnerable to security breaches, but Android devices are more lucrative for attackers. This is due to its open-source nature and the larger market share compared to other operating systems for mobile devices. We have proposed an android malware detection module based on advanced data mining and machine learning. While such a method may not be suitable for home users, being very processor heavy, this can be implemented at enterprise gateway level to act as a central antivirus engine to supplement antivirus present on end user computers. This will not only easily detect known viruses, but act as a knowledge that will detect newer forms of harmful files. While a costly model requires costly infrastructure, it can help to protect invaluable enterprise data from security threats and prevent immense financial damage

REFERENCES

[1] Varma, P. Ravi Kiran, Kotari Prudvi Raj, and KV Subba Raju. "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms." 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I- SMAC). IEEE, 2017.
[2] Darus, Fauzi Mohd, Salleh Noor Azurati Ahmad, and AswamiFadillah Mohd Ariffin. "Android Malware Detection Using Machine Learning on Image Patterns." 2018 Cyber Resilience Conference (CRC). IEEE, 2018.
[3] Chang, Wei-Ling, Hung-Min Sun, and Wei Wu. "An Android Behavior-Based Malware Detection Method using Machine Learning." 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). IEEE, 2016.

[4] Bhat, P.; Dutta, K. A survey on various threats and current state of security in android platforms. ACM Comput. Surv. (CSUR) 2019, 52, 1–35.

[5] Tam, K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. ACM Comput. Surv. (CSUR) 2017, 49, 1–41.