IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research Paper © 2012 IJFANS. All Rights Reserved, Journal Volume 14, Iss 04, 2025

THE USE OF EQUILIBRIUM OPTIMIZER WITH DEEP LEARNING MODEL FOR THE PURPOSE OF DETECTING MALWARE ON ANDROID DEVICES USING INTELLIGENT PATTERN RECOGNITION

Ashish Halder[1], Jaidi Mahathi Reddy[2], Vasamsetty Mahima Kumar[3], Kandi Teja Reddy[4], D. Sunitha[5], Dr G Veeraswamy[6],

[1,2,3&4]Ug Schoalr, Department Of Cse(Ai&Ml), Narsimha Reddy Engineering College (Ugc- Autonomous), Maisammaguda (V), Kompally, Secunderabad, Telangana-500100

[5]Assistant Professor, Department Of Cse(Ai&Ml), Narsimha Reddy Engineering College (Ugc- Autonomous), Maisammaguda (V), Kompally, Secunderabad, Telangana-500100

[6]Associate Professor, Department Of Civil Engineering, Narsimha Reddy Engineering College (Ugc- Autonomous), Maisammaguda (V), Kompally, Secunderabad, Telangana-500100

ABSTRACT

The objective of the project titled "Malware Analysis and Detection Using Machine Learning Algorithm" is to improve the effectiveness of cyber-security measures by reliably detecting malicious software via the use of sophisticated machine learning protocols. In addition to being built using Python, the project makes use of the Flask web framework for backend operations. Additionally, HTML, CSS, and JavaScript are used in order to provide a frontend experience that is both responsive and interactive. The Extra Tree Classifier and Logistic Regression are the two machine learning models that are at the core of this research. A training accuracy of 97.42% and a testing accuracy of 97.23% are both achieved by the Extra Tree Classifier model, which displays exceptional performance. As a point of reference, the Logistic Regression model achieves an accuracy of 94.84% during training and 93.67% during testing. Both models are trained and verified with the use of the TUNADROMD dataset, which consists of 4465 cases and 242 characteristics. The target classification attribute is responsible for discriminating between malware and goodware.For the purpose of the study, a subset of 23 qualities was chosen and selected on the basis of their significance and influence on the categorization problem. With this deliberate choice, we hope to improve the performance of the model while simultaneously lowering the complexity of the computations. The findings of the experiment reveal that the Extra Tree Classifier is very efficient in discriminating between dangerous and benign software. As a consequence, it provides a trustworthy instrument for the detection of malware in applications that are used in the real world. In general, this study illustrates the effectiveness of machine learning algorithms in the field of cyber security. It offers a strong solution for the identification of malware that can be included into a variety of digital security infrastructures.

1.INTRODUCTION

Of the many types of malware, the Android virus has recently garnered a lot of attention, and it has been discovered that it occupies a significant amount of attention all over the world. Android is a widely used operating system (OS) in comparison to other operating systems that dominate the market for OS technology. The assaults that are carried out by malware are well-known scenarios that may be identified as an assault on Android [3]. Different researchers have come up with a wide variety of definitions for malware, and these definitions are dependent on the harm that is performed. Malicious applications that use a variety of encryption methods with the intention of getting illegal access are the most important aspect of the infection. In the event of a breach in security, it is intended to carry out acts that are neither allowed nor suitable. Privacy, availability, and integrity are the three fundamental rules that play a significant role in the field of security [4]. It is possible to identify malicious software that is related with smart systems in three different settings as risks. These contexts include aims and attitude, sharing and infection pathways, and privilege acquisition techniques. Theft of information, spam e-mails, fraudulent activities, and inappropriate usage of websites are all examples of potentially harmful goals and behavioral perspectives. It is possible to identify software, browsers, marketplaces, systems, and networks as the directions that are responsible for distributing and infecting [5]. In the privilege and acquisition methods, procedure exploitation and employer management, such as social engineering, are included on the list of possible approaches. Malware that is specifically related with an Android operating system has been identified as Android malware, which is malware that causes harm to or steals information from a mobile device that is based on Android [6]. Ransomware, adware, botnets, spyware, worms, backdoors, and Trojans are all examples of these types of malicious software. Malware is defined by Google as characteristics that have the potential to do harm [7]. They classed malware as commercial and non-commercial privilege escalation, spyware, phishing, and other types of frauds such as Trojans, backdoors, toll fraud, and SMS fraud. Malware was separated into several categories. As of right now, a number of researchers have approached and created it in order to lessen the danger posed by malware on Android. The approaches for identifying malware may be broken down into two categories: dynamic analysis-based classification and static information analysis-based classification [8]. It is possible for dynamic analysis to regulate dangerous characteristics that use difficult approaches such as code packing or encryption. This is one of the advantages of using dynamic analysis. In the process of detecting malware, both machine learning (ML) and deep learning (DL) techniques have been shown to be useful, particularly in the context of Android feature analysis and in larger domains of cybersecurity [9]. When it comes to Android malware identification, it has been discovered that DL approaches are far more effective than typical ML methods. Static analysis, which is based on algorithms, uses linguistic aspects that may be deleted without the installation of a feature, while dynamic analysis-based approaches make use of semantic elements that are detected when an application is operated in certain situations [10]. Both machine learning and non-machine learning techniques struggle with issues such as limited effectiveness in the development of malware patterns and a higher overall rate of false positives. After that, it was concluded that DL techniques were effective, with the capacity to automatically extract detailed characteristics and detect tough patterns being highlighted. This made them more capable of dealing with the dynamic nature of Android malware. Putting an emphasis on this particular aspect, the study endeavors to enhance the performance of Android malware detection, with the goal of contributing to the development of solutions that are more accurate and efficient in the constantly shifting field of cybersecurity assaults. The purpose of this research is to offer an Intelligent Pattern Recognition approach for Android Malware Recognition that makes use of an Equilibrium Optimizer with Deep Learning (IPREODL) methodology. The primary objective of the IPR-EODL system is to correctly detect and classify malicious software for Android devices in a manner that allows for effective security measures to be taken. As part of the IPR-EODL method, the data preparation stage was carried out in order to transform the input data into a configuration that was compatible. Furthermore, the IPR-EODL technique utilizes the channel attention long short-term memory (CALSTM) algorithm in order to identify malicious software that is designed to infect Android devices. With the IPR-EODL technique, the EO algorithm is used for the hyperparameter tuning operation. This is done in order to improve the solution that the CA-LSTM algorithm provides. An Android malware database that serves as a benchmark may be used to assess the experimental evaluation of the IPR-EODL technique. The following is a summary of the most important contributions that the study made. Through the implementation of an IPR-EODL algorithm for Android malware detection, a novel approach will be presented in order to overcome the issues that are prevalent in this area. The primary purpose of the IPR-EODL algorithm is to reliably identify and categorize malicious software for Android devices, hence emphasizing the need of implementing stringent security measures.Employs the CA-LSTM methodology for the purpose of conducting an effective analysis and detection of Android malware. It illustrates standard methods by identifying long-term dependencies and focusing on significant characteristics via the use of channel attention.Combines the EO technique with the CA-LSTM algorithm in order to optimize the hyperparameters of the algorithm. Additionally, this optimizer models efficiency and has the ability to improve accuracy in relation to parameters that are either fixed or manually modified.

2.LITERATURE REVIEW

An technique known as intelligent hyper parameteraltered DL-based malware detection (IHPT-DLMD) was suggested by the authors in the previous reference [11]. Additionally, the Bi-directional Long Short-Term Memory (BiLSTM) approach has been used for the purpose of identifying and classifying malicious software that is designed to infect Android devices. In conclusion, the glowworm swarm optimization (GSO) strategy has been introduced as a means of improving the hyperparameters of the BiLSTM method. A standard dataset is used to evaluate the model that has been developed. In [12], the authors described a machine learning-based approach for recognizing malicious software on Android devices. In addition to that, the method that was developed takes advantage of the Information Gain design. Several different integrations of API Calls, contextual features, and authorizations are used by them. The integrations were then loaded into a variety of machine learning models. [13] introduced a new method for identifying malicious software in Android applications using GRU, which is a form of RNN methodology. This method was provided by the authors. In [10], a machine learning-based recognition approach is suggested. This technique takes use of hybrid analysis-based particle swarm optimization (PSO) and an adaptive genetic algorithm (AGA). First and foremost, the process of feature selection (FS) was carried out by applying PSO to the features. Following that, the AGA works to improve the presentation of XGBoost and random forest (RF) by making use of machine learning IDs. Image-based malware visualizations of Android Dalvik Executable (DEX) files were used by the author in [14] to create a structure that was projected. For the purpose of feature abstraction, the structure makes use of the EffectiveNetB0 convolutional neural network (CNN). The resulting feature abstraction was then transmitted via a global average pooling layer and presented in the form of a stacking identifier. Through the use of a DL-based Android malware detection (RHSODL-AMD) technique, the author of [7] proposes a Rock Hyrax Swarm Optimizer. Additionally, the Adamax enhancer with attention recurrent autoencoder (ARAE) approach has been used for the purpose of recognizing malicious software on Android devices. In reference number 15, a CNN approach that has been pre-trained for Android malware identification is proposed. A technique that is based on EfficientNet-B4 CNN has been created in order to identify malicious software for Android. These attributes are delivered as a softmax identifier after being transmitted by a layer that functions as a global average pooling layer. A unique DL-based approach was provided by the authors with the number 16. Visualization of a portable executable (PE) file as a colorful graphic is the primary function of this program. On the other hand, it recognizes malicious software that is based on deep features and uses support vector machine (SVM) technology. The proposed model incorporates deep learning and machine learning techniques, and it is evaluated with the use of benchmark datasets.

The authors, A. Gómez and A. Muñoz, Utilizing deep learning for the purpose of attack detection and categorization in Android devices Because of the ever-increasing prevalence of Android-based smartphones, which presently hold an astounding 72% percent of the worldwide market, these devices have become an ideal target for cybercriminals. Because of this, the identification of malicious software for Android devices has become an important topic of study. Despite the fact that both academic institutions and private businesses have investigated a variety of methods in order to produce solid and effective solutions for Android malware detection and classification, this problem continues to be a persistent obstacle. Within the scope of this investigation, we describe a supervised learning approach that exhibits encouraging outcomes in the identification of malware on Android devices. The construction of a complete labeled dataset, which includes over 18,000 samples that have been categorized into five unique categories—namely, Adware, Banking, SMS, Riskware, and Benign applications—is the most important aspect of our technique. Validation of the efficacy of our proposed model is accomplished via the use of well-established datasets such as CICMalDroid2020, CICMalDroid2017, and CICAndMal2017. In terms of accuracy, recall, efficiency, and other essential criteria, our approach surpasses previous semi-supervised approaches in particular parameters. This occurs when we compare our findings with techniques that are considered to be state-of-the-art. On the other hand, we fully accept that our model does not display any substantial variances when compared to other techniques in terms of certain elements. In general, the findings of our study provide a contribution to the continuing efforts that are being made to build more sophisticated methods for the identification and classification of malware on Android devices. With the help of our discoveries, we anticipate that more investigations will be prompted, which will ultimately result in improved security measures and protection for Android devices in the face of constantly increasing threats.

J. Grundy, Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, and J. Klein are the authors of this study. An examination of the influence that sample duplication has on the detection of malware using machine learning for Android
In the world of Android, the identification of malware on a large scale is often accomplished by the use of machine learning algorithms. It has been claimed that innovative methods like DREBIN and MaMaDroid, which are considered to be state-of-the-art, provide high identification rates when evaluated against well-known datasets. The unfortunate reality is that such datasets could include a significant amount of duplicated samples, which might potentially introduce bias into the recorded experimental findings and insights. This paper presents the results of comprehensive tests that we conducted in order to quantify the performance gap that arises as a result of the de-duplication of datasets. Based on the findings of our experiments, we have determined that the influence of duplication in published datasets on supervised malware classification models is quite minor. When compared to the findings that Allamanis discovered about the general issue of machine learning bias for huge code, this discovery stands in stark contrast. However, the results of our tests indicate that sample duplication has a more significant impact on unsupervised learning models (for example, malware family grouping). Nevertheless, we contend that our fellow researchers and practitioners should always take sample duplication into mind when undertaking machine-learning-based Android malware detections (by either supervised or unsupervised learning), regardless of how large the effect may be. This is because sample duplication may have a considerable influence.
As well as H. Wang and W. Zhang, H. He The permits have informed me that you are there! Android virus detection using a combination of several strategies

A major rise in the use of Android devices in a variety of facets of our lives has been seen over the course of the last few years. On the other hand, users have the ability to download Android applications via third-party channels, which opens up a multitude of options for malicious software. When an attacker wants to acquire access to the sensitive private intelligence of users, they will employ rights that they have not requested. In light of the fact that signature-based antivirus solutions are no longer able to satisfy practical requirements, there is an urgent need for solutions that are both efficient and adaptive, particularly in novel versions. We offer a hybrid Android malware detection strategy that combines dynamic and static strategies as a solution to the problem presented here. In the initial step of our process, we use a technology that is based on machine learning to do static analysis, which involves inferring different permission usage patterns between malicious and benign applications. After extracting the object reference associations from the RAM heap, we develop a dynamic feature base in order to further identify the applications that are deemed to be suspicious. Following that, we provide an enhanced version of the DAMBA-based state-based algorithm. The results of our experiments on a real-world dataset consisting of 21,708 applications demonstrate that our method exhibits superior performance than the well-known detector with a 97.5% F1-measure. Moreover, it has been proved that our system is resistant to obfuscation methods and behaviors that include the misuse of permissions. Three of the authors, H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak Detection of Android malware with adversarial superiority: Lessons learned from reinforcement learning-based evasion attacks and defenses Three of the authors, H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak Android devices are now being used by billions of people, and as a result, they have become a financially profitable target for those who build malicious software. Consequently, in this zero-sum game of malware detection between the anti-malware community and malware producers, it is more of a need than a desire to be one step ahead of the competition. The purpose of this study is to construct adversarially better android malware detection models by focusing on a proactive adversary-aware framework designed for that purpose. The first thing that we do is study the adversarial resilience of thirty-six different malware detection models. These models were developed using two static characteristics (permission and intent) and eighteen different classification techniques. Within the framework of reinforcement learning, we developed two Targeted Type-II Evasion Attacks, namely TRPO-MalEAttack and PPO-MalEAttack, with the purpose of exploiting flaws in the malware detection models referenced above. In order to subvert any malware program into an adversarial application that is capable of fooling malware detection algorithms, the assaults are designed to introduce the fewest possible perturbations into each application. In thirty-six different malware detection models, the TRPO-MalEAttack obtains an average fooling rate of 95.75% (with 2.02 mean perturbations), which results in a reduction in the average accuracy from 86.01% to 49.11%. On the other hand, The

PPO-MalEAttack is able to obtain a higher average fooling rate of 96.87% (with 2.08 mean perturbations), which results in a decrease in the average accuracy from 86.01% to 48.65% in the same thirty-six detection models. In addition to this, we compile a list of the 10 most susceptible permissions and intents for Android, which an adversary might utilize to create further malicious apps. Subsequently, we offer a defensive mechanism known as MalVPatch in order to combat the adversarial assaults that are directed against malware detection algorithms. The MalVPatch protection is able to produce a greater detection accuracy in addition to a significant increase in the adversarial resilience of malware detection models. As a conclusion, we have reached the conclusion that it is essential to investigate the adversarial robustness of models prior to their deployment in the real world. This investigation also contributes to the achievement of adversarial superiority in android malware detection.

3.EXISTING SYSTEM:

❖ The existing system for Android malware recognition was developed using an Equilibrium Optimizer with Deep Learning (IPR-EODL) approach. This innovative system integrates the Equilibrium Optimizer algorithm with deep learning techniques to enhance the detection and classification of Android malware.

❖ The Equilibrium Optimizer is a nature-inspired optimization algorithm, which mimics the dynamic and equilibrium states observed in physical systems. By integrating this optimizer with deep learning models, the system effectively enhances the training process, ensuring more accurate and efficient malware detection.

❖ In the IPR-EODL approach, the deep learning component typically involves advanced neural networks that are adept at processing large datasets and identifying complex patterns associated with malware behavior. This synergy between the Equilibrium Optimizer and deep learning models allows for robust feature selection and classification, resulting in a highly reliable malware recognition system.

❖ This existing system leverages the extensive data processing capabilities of deep learning and the optimization efficiency of the Equilibrium Optimizer to achieve high performance in identifying malicious software. It represents a significant step forward in the field of cybersecurity, particularly in protecting Android devices from various malware threats.

3.1. DISADVANTAGES OF EXISTING SYSTEM:

❖ While the Equilibrium Optimizer with Deep Learning (IPR-EODL) approach for Android malware recognition presents significant advancements, it also has several disadvantages:

❖ High Computational Complexity: The integration of the Equilibrium Optimizer with deep learning models demands substantial computational resources. The process of training deep neural networks, combined with the optimization algorithm, can be computationally intensive and time-consuming, requiring powerful hardware and extended training periods.

❖ Scalability Issues: Due to the high computational requirements, scaling the system to handle larger datasets or more complex models can be challenging. This limitation can hinder its application in environments with rapidly growing data or diverse malware types.

❖ Energy Consumption: The deep learning models and optimization processes consume significant amounts of energy. This is particularly problematic for mobile and embedded systems, such as Android devices, which have limited battery life and processing power.

❖ Overfitting Risk: The complexity of deep learning models increases the risk of overfitting, especially when dealing with insufficient or imbalanced training data. Overfitting can lead to reduced generalization capability, making the model less effective in detecting new, unseen malware variants.

❖ Maintenance and Updates: Maintaining and updating the system can be labor-intensive. As new malware emerges, the models need to be frequently retrained and updated to remain effective. This continuous maintenance requires ongoing expertise and resources.

❖ Interpretability: Deep learning models, especially when combined with optimization algorithms, often act as black boxes, making it difficult to interpret their decision-making processes. This lack of transparency can be a drawback for security professionals who need to understand and trust the detection mechanisms.

❖ Latency Issues: Real-time malware detection can suffer from latency due to the complex computations involved in the IPR-EODL approach. This delay can be detrimental in scenarios where immediate detection and response are critical.

❖ Overall, while the IPR-EODL approach represents a significant technological advancement in malware detection, these disadvantages highlight the need for more efficient, scalable, and interpretable solutions in the field of cybersecurity.

3.2. PROPOSEDSYSTEM:

❖ The proposed system for malware analysis and detection leverages machine learning algorithms to enhance the accuracy and efficiency of identifying malicious software. This system is developed using Python as the primary coding language, with the Flask web framework for backend operations, and a frontend built with HTML, CSS, and JavaScript to ensure a user-friendly and interactive interface.

❖ The proposed system utilizes two primary machine learning models: the Extra Tree Classifier and Logistic Regression. These models are selected for their proven effectiveness in classification tasks, particularly in distinguishing between malware and benign software.

❖ In the proposed system, TUNADROMD dataset is employed for training and testing the models. This dataset contains 4465 instances and 242 attributes, with a specific target attribute for classification—categorizing the data into malware or goodware. For the purpose of this project, 23 attributes are carefully selected to optimize the classification process.

❖ In the proposed system, the Extra Tree Classifier and Logistic Regression models are trained and evaluated to assess their performance. The Extra Tree Classifier achieves a training accuracy of 97.42% and a testing accuracy of 97.23%.

The Logistic Regression model attains a training accuracy of 94.84% and a testing accuracy of 93.67%.

❖ In the proposed system, a subset of 23 attributes is chosen from the original 242 attributes in the dataset. This selection process is aimed at reducing the dimensionality of the data, improving model performance, and focusing on the most relevant features for malware detection. The backend, developed using Flask, handles data processing and model inference. The frontend, created with HTML, CSS, and JavaScript, provides a seamless and intuitive interface for users to interact with the system, view results, and manage the detection processes.

❖ Overall, the proposed system focuses on applying machine learning techniques to malware detection, ensuring high accuracy and efficient performance through careful model selection and feature engineering.

3.3. ADVANTAGES OF PROPOSED SYSTEM:

❖ High Accuracy: The proposed system achieved high accuracy in detecting malware, with the Extra Tree Classifier achieving a training accuracy of 97.42% and a testing accuracy of 97.23%. This high level of accuracy ensures reliable detection of malicious software, reducing false positives and false negatives.

❖ Efficiency in Detection: In the proposed system, by selecting only 23 relevant attributes from the TUNADROMD dataset, the system reduces computational complexity and processing time, leading to faster detection and classification of malware.

❖ User-Friendly Interface: The integration of HTML, CSS, and JavaScript for the frontend provides a responsive and intuitive user interface. This design makes it easier for users to interact with the system, view results, and manage malware detection processes without requiring extensive technical knowledge.

❖ Scalability: Developed using Python and the Flask web framework, the system is highly scalable. It can be easily adapted and expanded to accommodate larger datasets, more features, or additional machine learning models, making it suitable for evolving cybersecurity needs.

❖ Flexibility in Model Use: The system employs both the Extra Tree Classifier and Logistic Regression models, offering flexibility in approach. Users can choose the model that best fits their specific requirements or context, whether it be a preference for tree-based methods or linear models.

❖ Reduced Overfitting: The careful selection of 23 attributes helps mitigate the risk of overfitting, enhancing the generalization capabilities of the models. This ensures that the system performs well not only on the training data but also on new, unseen data.

❖ Integration Capabilities: The Flask framework facilitates easy integration with other systems and technologies, allowing the proposed system to be incorporated into broader cybersecurity infrastructures or combined with additional security tools.

❖ Maintenance and Update Efficiency: The modular architecture of the system, with clear separation between backend processing and frontend presentation, simplifies maintenance and updates. Enhancements to the machine learning models or user interface can be implemented with minimal disruption to the overall system.

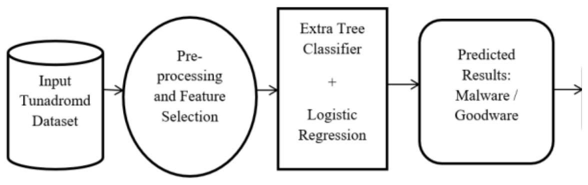
❖ Transparency and Interpretability: The use of Logistic Regression, in particular, offers a degree of interpretability, allowing users to understand the decision-making process of the model. This transparency is crucial for gaining trust and facilitating informed decisions in cybersecurity operations.

❖ Resource Optimization: By using machine learning algorithms that are computationally efficient, the proposed system optimizes resource usage. This makes it feasible to deploy on a variety of platforms, including those with limited computational power, such as mobile devices.

❖ Overall, the proposed system presents a robust and versatile solution for malware detection, combining high accuracy, efficiency, and user-friendliness with the capability to adapt and scale to meet future cybersecurity challenges.

4.IMPLEMENTATION

4.1SYSTEM ARCHITECTURE


Input Tunadromd Dataset
Pre-processing and Feature Selection
Extra Tree Classifier + Logistic Regression
Predicted Results: Malware / Goodware

4.2.MODULES:

- ❖ Data Collection
- ❖ Dataset
- ❖ Data Preparation
- ❖ Feature Extraction
- ❖ Splitting the dataset
- ❖ Model Selection
- ❖ Analyze and Prediction
- ❖ Accuracy on test set
- ❖ Saving the Trained Model
- ❖ Prediction Module
- ❖ Model Evaluation Module

Data Collection:

❖ In the first module of the Malware Analysis and Detection Using Machine Learning Algorithm, we make the data collection process. This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.

❖ There are several techniques to collect the data, like web scraping, manual interventions. The dataset is located in the model folder. The dataset is referred from the popular dataset repository called kaggle. The following is the link of the dataset:

❖ Kaggle Dataset Link: https://www.kaggle.com/datasets/jayaprakashpondy/android-malware

Dataset:

❖ In this module, we use the dataset which is the primary source of data for the system. This dataset contains 4465 instances and 242 attributes, with a target attribute for classification (malware vs. goodware).

❖ User Input: Data provided by users through the web interface, allowing for real-time malware detection based on user-uploaded files or input data.

Data Preparation:

❖ This module is responsible for preparing the TUNADROMD dataset for analysis. It involves tasks such as data cleaning, normalization, and feature selection. Specifically, 23 relevant attributes are selected from the original 242 attributes to optimize the machine learning models.

❖ Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.).

❖ Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data.

❖ Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis.

❖ The labels are cleaned to correct spelling errors (e.g., 'Begnin' to 'Benign').

❖ NaN values are dropped from the dataset.

❖ The data is balanced using the RandomOverSampler from imblearn to handle class imbalance between 'Malware' and 'Benign' applications.

Feature Extraction:

❖ If the dataset contains raw binaries or other non-numeric data, extract features that can be used by the machine learning models. This may involve static analysis (e.g., analyzing the binary's structure) or dynamic analysis (e.g., monitoring the binary's behavior during execution).

❖ A subset of features (permissions) is selected for model training to reduce dimensionality and focus on relevant attributes.

Splitting the dataset:

❖ Data Splitting and Validation is crucial for training and evaluating the model. This module divides the dataset into training, validation, and testing sets. It ensures that the model's performance is assessed accurately using proper validation techniques like cross-validation. Split the dataset into train and test. 80% train data and 20% test data.

Model Selection:

❖ This module handles the training of the machine learning models using the preprocessed data. It implements the Extra Tree Classifier and Logistic Regression algorithms.

Extra Tree Classifier:

❖ The Extra Tree Classifier, also known as Extremely Randomized Trees, is an ensemble learning method primarily used for classification and regression tasks. It is part of the broader family of decision tree methods and is implemented in the sklearn library under the ExtraTreesClassifier module.

❖ The Extra Tree Classifier constructs a forest of randomized decision trees. Unlike traditional decision trees where each split is determined by selecting the best feature threshold combination to minimize a criterion like Gini impurity or information gain, Extra Trees selects the split points randomly. This randomness is introduced in two main aspects:

❖ For each feature, random split points are generated.

❖ A subset of features is selected at random for the split.

❖ This results in a high variance reduction due to averaging multiple trees, which enhances the model's robustness and prevents overfitting.

Logistic Regression:

❖ Logistic Regression is a fundamental statistical technique used for binary classification tasks. Despite its name, it is a classification algorithm, not a regression algorithm. Logistic Regression estimates the probability that a given input belongs to a particular class. It is implemented in the sklearn library under the LogisticRegression module.

❖ Logistic Regression models the relationship between the dependent variable (target) and one or more independent variables (features) by using the logistic function, also known as the sigmoid function. The sigmoid function maps any real-valued number into the range (0, 1), which is interpreted as a probability.

❖ Logistic Regression is a versatile and powerful method for binary classification tasks, offering a balance between simplicity and performance. It is widely used due to its ease of interpretation and ability to handle large datasets efficiently. By incorporating regularization techniques, it can be adapted to avoid overfitting and perform well in various scenarios.

Analyze and Prediction:

❖ This module performs feature selection to identify the most impactful attributes for malware classification. It ensures that the selected features contribute significantly to the model's performance.

Accuracy on test set:

❖ Once the model is trained, it needs to be evaluated for its performance. This module involves splitting the dataset into training and testing subsets and assessing the model's accuracy, precision, recall, and F1-score.

❖ The evaluation metrics provide insights into the system's ability to predict and detect android malware. The Extra Tree Classifier achieves a training accuracy of 97.42% and a testing accuracy of 97.23%. The Logistic Regression model attains a training accuracy of 94.84% and a testing accuracy of 93.67%.

Saving the Trained Model:

❖ Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into an .h5 or .pkl file using a library like pickle.

❖ Make sure you have pickle installed in your environment.

❖ Next, let’s import the module and dump the model into .pkl file.

Prediction Module:

❖ This module handles real-time predictions using the trained models. Users can input new data through the frontend, and the module processes this data to classify it as malware or benign.

Model Evaluation Module

❖ This module evaluates the performance of the trained models using the testing dataset. It calculates accuracy metrics and other performance indicators to assess model effectiveness.

❖ Evaluate model accuracy, precision, recall, and F1-score.

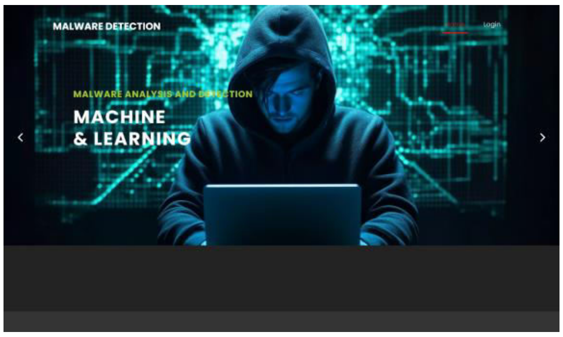
❖ Generate confusion matrices for both models.

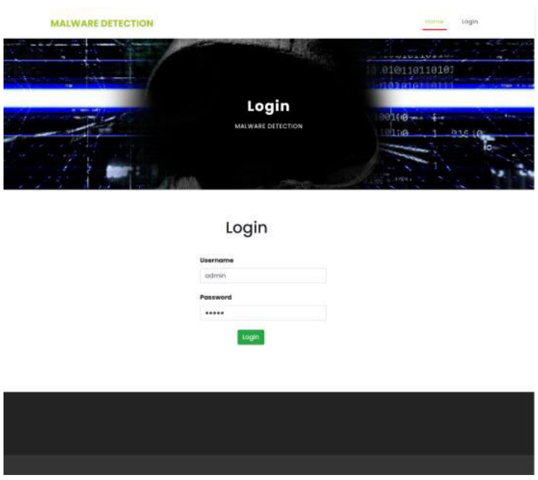
❖ Compare the performance of the Extra Tree Classifier and Logistic Regression models.

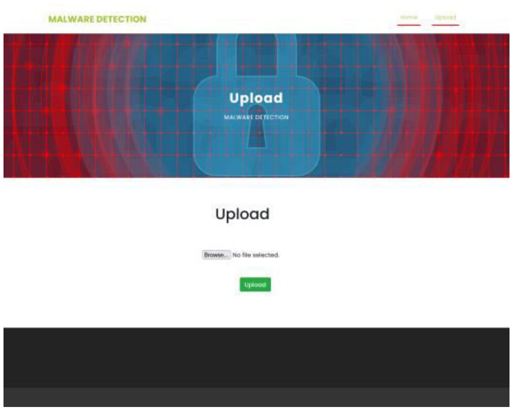
❖ Accuracy, precision, recall, and F1-score are used to evaluate model performance.

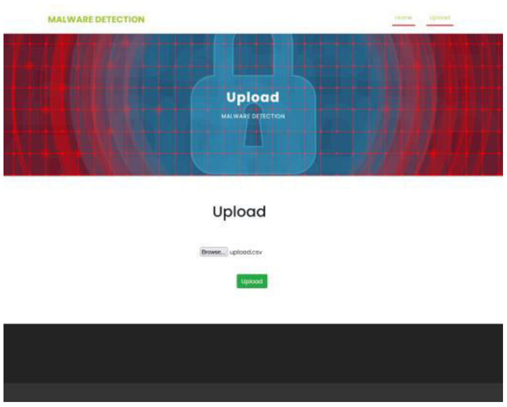
❖ Confusion matrix is visualized using seaborn heatmap to understand the classification results.

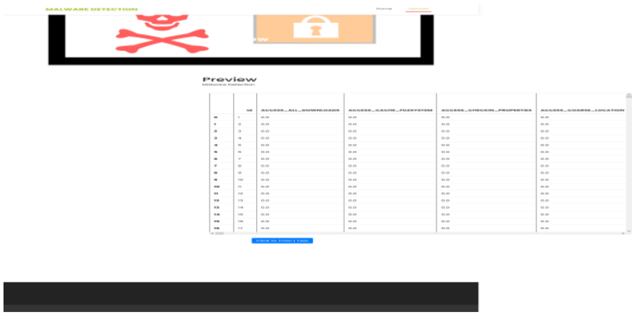
5.RESULTS

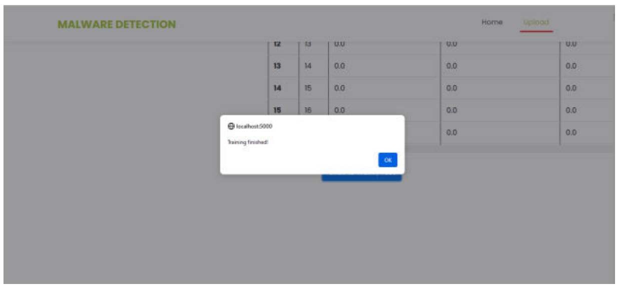


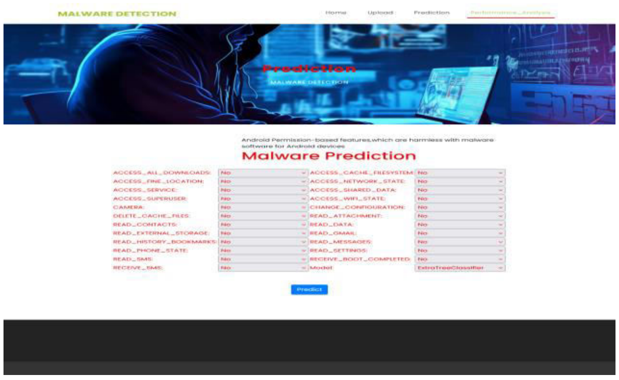








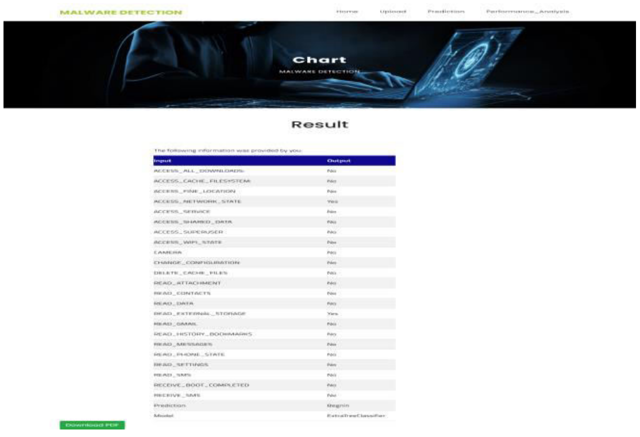




Android Permission-based features,which are harmless with malware software for Android devices

Malware Prediction

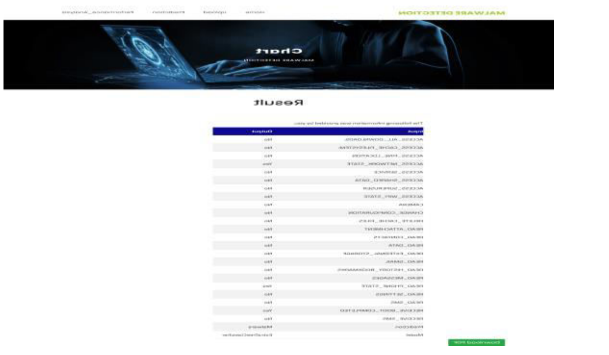
ACCESS_ALL_DOWNLOADS:	No	ACCESS_CACHE_FILESYSTEM:	No
ACCESS_FINE_LOCATION:	No	ACCESS_NETWORK_STATE:	No
ACCESS_SERVICE:	No	ACCESS_SHARED_DATA:	No
ACCESS_SUPERUSER:	No	ACCESS_WIFI_STATE:	No
CAMERA:	No	CHANGE_CONFIGURATION:	No
DELETE_CACHE_FILES:	No	READ_ATTACHMENT:	No
READ_CONTACTS:	No	READ_DATA:	No
READ_EXTERNAL_STORAGE:	No	READ_GMAIL:	No
READ_HISTORY_BOOKMARKS:	No	READ_MESSAGES:	No
READ_PHONE_STATE:	No	READ_SETTINGS:	No
READ_SMS:	No	RECEIVE_BOOT_COMPLETED:	No
RECEIVE_SMS:	No	Model	ExtraTreeClassifier

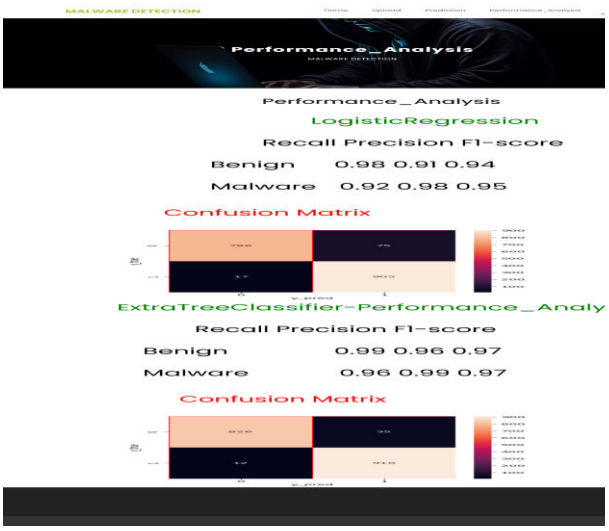


Result

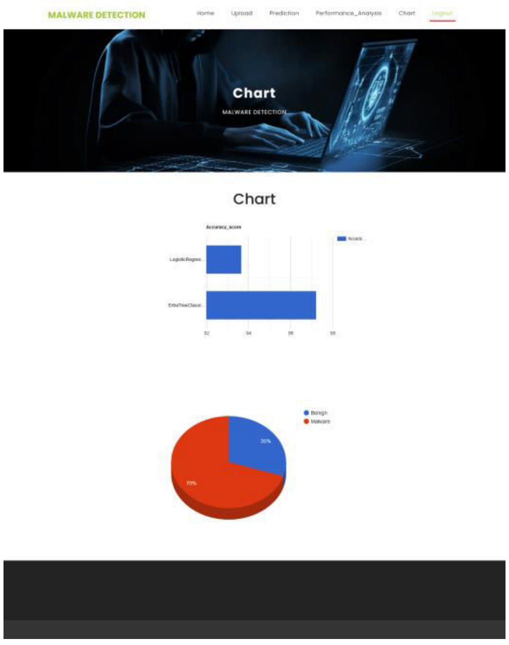
The following information was provided by you:

Input	Output
ACCESS_ALL_DOWNLOADS:	No
ACCESS_CACHE_FILESYSTEM	No
ACCESS_FINE_LOCATION	No
ACCESS_NETWORK_STATE	Yes
ACCESS_SERVICE	No
ACCESS_SHARED_DATA	No
ACCESS_SUPERUSER	No
ACCESS_WIFI_STATE	No
CAMERA	No
CHANGE_CONFIGURATION	No
DELETE_CACHE_FILES	No
READ_ATTACHMENT	No
READ_CONTACTS	No
READ_DATA	No
READ_EXTERNAL_STORAGE	Yes
READ_GMAIL	No
READ_HISTORY_BOOKMARKS	No
READ_MESSAGES	No
READ_PHONE_STATE	No
READ_SETTINGS	No
READ_SMS	No
RECEIVE_BOOT_COMPLETED	No
RECEIVE_SMS	No
Prediction	Begnin
Model	ExtraTreeClassifier





```
Performance_Analysis
LogisticRegression
Recall Precision F1-score
Benign     0.98 0.91 0.94
Malware    0.92 0.98 0.95
Confusion Matrix
ExtraTreeClassifier-Performance_Analy
Recall Precision F1-score
Benign          0.99 0.96 0.97
Malware         0.96 0.99 0.97
Confusion Matrix
```



CONCLUSION

Within the scope of this work, our primary focus was on developing a novel IPR-EODL approach for Android Malware Recognition that is both effective and automated. The objective of the IPREODL approach is to correctly detect and classify the malicious software that is designed to infect Android devices in such a manner that security may be accomplished. In the IPR-EODL approach, there is a three-stage procedure that is involved. These stages include data preprocessing, CA-LSTM-based Android malware detection, and EO-based hyperparameter tuning. Within the scope of this study, the IPR-EODL approach has used the CALSTM model in order to identify malicious software for Android devices. The EO algorithm is used in the IPR-EODL methodology for the hyperparameter tuning procedure. This is done in order to improve the performance of the CA-LSTM approach. On a benchmark Android malware database, the IPR-EODL approach was tested and evaluated via the process of experimentation and assessment. The IPR-EODL approach has been shown to have a considerable performance on the Android malware detection process, with a maximum accuracy of 99.18%, as shown by the extensive results. It is possible that future work may concentrate on the development of an ensemble classifier for an improved malware detection procedure implemented on Android.

REFERENCES

[1] A. Gómez and A. Muñoz, "Deep learning-based attack detection and classification in Android devices," Electronics, vol. 12, no. 15, p. 3253, Jul. 2023.

[2] Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy, "On the impact of sample duplication in machine-learning-based Android malware

detection," ACM Trans. Softw. Eng. Methodol., vol. 30, no. 3, pp. 1–38, Jul. 2021.

[3] H. Wang, W. Zhang, and H. He, "You are what the permissions told me! Android malware detection based on hybrid tactics," J. Inf. Secur. Appl., vol. 66, May 2022, Art. no. 103159.

[4] H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, "Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses," Forensic Sci. Int., Digit. Invest., vol. 44, Mar. 2023, Art. no. 301511.

[5] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-LADY: Deep learning-based Android malware detection using Dynamic features," J. Internet Serv. Inf. Secur., vol. 11, no. 2, pp. 34–45, 2021.

[6] M. Ibrahim, B. Issa, and M. B. Jasser, "A method for automatic Android malware detection based on static analysis and deep learning," IEEE Access, vol. 10, pp. 117334–117352, 2022.

[7] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification," Appl. Sci., vol. 13, no. 4, p. 2172, Feb. 2023.

[8] A. Batouche and H. Jahankhani, "A comprehensive approach to Android malware detection using machine learning," in Information Security Technologies for Controlling Pandemics, 2021, pp. 171–212.

[9] P. Bhat and K. Dutta, "A multi-tiered feature selection model for Android malware detection based on feature discrimination and information gain," J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 10, pp. 9464–9477, Nov. 2022.