Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Adaptive Static Analysis for Android Malware Detection: A Machine Learning-Based Security Framework

[1]Hiral Patel [2]Dr. Mukta Agarwal
[1] *Research Scholar, Department of Computer Science, Indus University, Ahmedabad, Gujarat, India*
E-mail: [1]patelhiral.21.rs@indusuni.ac.in
[2]*Assistant Professor, Department of Computer Science, Indus University, Ahmedabad, Gujarat, India*
E-mail: [2]muktaagarwal.dcs@indusuni.ac.in

ARTICLE INFO

Received: 22 Sep 2024

Revised: 12 Dec 2024

Accepted: 24 Dec 2024

ABSTRACT

Users and organisations face serious security concerns as a result of the surge in malware threats brought on by the quick expansion of Android applications. Machine learning (ML)-based methods are a viable substitute for traditional signature-based malware detection systems, which find it difficult to keep up with changing threats. In this research, we investigate how to identify malware in Android apps using machine learning (ML) approaches. The study of various data pre-processing, dimensionality reduction, and classification methods is the main focus, and the generalisation capacity of the learnt models is evaluated using publicly available dataset. We place a strong emphasis on using feature selection (FS) techniques to minimise the dimensionality of the data and find the most pertinent features for classifying Android malware, which makes this process more explainable. Our methodology can determine which features are most essential to identifying an app as malicious. High accuracy in detecting malware in Android apps is achieved by the suggested method.

Keywords: Android malware detection; Machine learning; Static analysis; Feature extraction; Mobile security

1. INTRODUCTION

Due to the growth of the Internet and technological advancements in smartphones, the number of smartphone users has skyrocketed. This has promoted rivalry between different software sectors in an effort to provide consumers with robust platforms for their devices. Due to the ease and effectiveness of many applications and the ongoing advancements in smart device hardware and software, smartphone usage and related applications are rising quickly in this technological age (Statista, 2024). By 2027, there will likely be close to 7.7 billion smartphone users globally (Statista, 2024). The most popular operating system (OS) for mobile devices is Android. It had a 72.24% market share as of October 2024 (StatCounter, 2024). Apple iOS held the second-highest share at 27.28%, while Samsung, KaiOS, and other small suppliers split the remaining 0.48% of the market (StatCounter, 2024). Google Play is an online marketplace for Android apps that allows users to explore, download, and install a large selection of Android apps (AppBrain, 2024). As of october 2024, Google Play currently hosts 1.57 million apps (AppBrain, 2024). Because Android is so widely used, it is a more desirable target for fraudsters and is more vulnerable to viruses and malware. Malware operations consistently target and effectively infect Android devices, even with notable security improvements made by Google and original equipment manufacturers (OEMs) (Dassanayake, 2021). More than 98% of hacks that target mobile devices focus on Android (Kaspersky, 2020). Numerous attack vectors are used in these attacks, which take advantage of the dynamic attack surface that mobile devices reveal (Townsend, 2020).Android malware researchers have focused on machine learning algorithms in their quest for more potent malware detection solutions, as traditional antivirus techniques (such as fingerprinting and blacklisting) have shown themselves to be ineffective and limited in protecting end users in the mobile domain (Timothy, 2022), especially against encrypted and zero-day malware. Supervised and unsupervised machine learning techniques efficiently extract app features through static and dynamic analysis and classify Android apps into two categories: malicious and benign (Kosmidis & Kalloniatis, 2017). In static analysis, features are extracted from Java bytecode or the AndroidManifest.xml file,

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

which includes a collection of features and contextual data, including permissions needed by the application (Kapratwar et al., 2017). The app is analyzed without running it. Static analysis is simple and faster than other analysis types. Dynamic analysis is used to find malicious activity in running apps. It gathers all the system calls the application performs while running. Moreover, unidentified application signatures are also successfully analyzed by dynamic analysis (Bhatia & Kaushal, 2017). However, dynamic analysis is costly and time-consuming. It requires more computational resources and also increases complexity, decreasing its popularity.

Paper Contributions

This work focuses on using machine learning approaches to detect malware in Android applications, and its main contributions are the following:

- presents a static analysis approach to enhance malware detection accuracy. Extraction of relevant features from static sources, enabling a more comprehensive representation of Android applications.
- Pre-processing apply on dataset and Feature selection techniques are applied to retain the most significant attributes, improving the efficiency and accuracy of classification.
- A classification model is developed to distinguish between benign and malicious applications. Evaluates various machine learning classifiers to identify the most effective one for Android malware detection.
- The final model is evaluated using standard evaluation metrics (e.g., accuracy, precision, recall, F1-score, and AUC_ROC) to assess its effectiveness.

This is how the remaining part of the paper is structured. An overview of the Android application, ML classifiers and dataset is given in Section-2. Section-3 describes related work. The suggested work and methods address section-4. Experimental results are presented in section-5. The paper concludes with a conclusion and recommendations for further work in section-6.

2. BACKGROUND

Android malware detection using machine learning is a complicated subject that necessitates knowledge of both Android and machine learning principles. The pertinent topics are briefly introduced here.

This section provides a quick review of Android malware detection, including ML approaches, datasets, and algorithms. In Section 2.1, an overview of Android apps is given, and in Section 2.2, malware types are discussed. Next, Sections 2.3 and 2.4 describe machine learning algorithms and measures for evaluating them, respectively.

2.1 Android application structure

Android OS is an open-source mobile operating system built on the Linux platform. It was first released in 2008 and has since undergone numerous iterations with new releases occurring every few months. The basic elements of an Android app must be understood in order to comprehend how malware might take advantage of the Android OS. The components of the Android package kit (APK) file for an Android application are shown in Figure 1 (Palma et al., 2024).

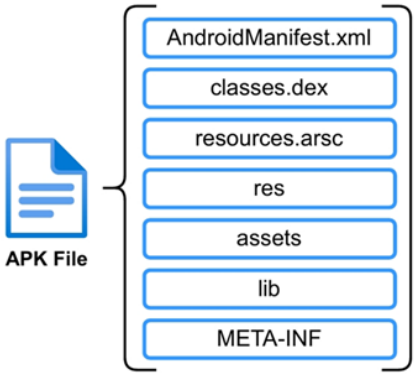

Fig-1 Components of an APK file

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A deeper comprehension of some of the most important security features of an Android app is made possible by knowledge of its architecture. For example, in order to execute some functions, apps need permissions from the system. Frequently, malware takes use of these permissions and capabilities to conduct assaults. Applications use permissions to communicate with each other and gain access to hardware and other aspects of the system. The manifest file must specify the permissions the application uses. The AndroidManifest.xml file, along with the permissions the app requests, is therefore important in identifying whether an app is malicious.

2.2 Malware attacks on android

The most frequent instance of an Android danger is malware attacks. Several researchers have defined malware differently based on the harm they inflict. The term malware refers to any malicious application that contains malicious code and has the ultimate goal of gaining unauthorized access and engaging in unethical or illegal behaviour.

Mobile malware that damages or steals data from an Android-based device is known as Android malware and is especially associated with the Android operating system. Trojan horses, adware, ransomware, worms, botnets, backdoors, and spyware are some of the classifications for these (Faruki et al., 2014).

2.3 Machine learning algorithms

In this section, a brief description of the Machine learning algorithms used in this research is presented. Random forest (RF) is an ensemble technique that combines the results of several decision trees and obtain a single result (Breiman, 2001). In order to optimize the margin between classes in a dataset, the Support Vector Machine (SVM) finds the best hyperplane. Support vectors, or important data points, that affect the position of the hyperplane are identified. In the event that the data cannot be separated linearly, SVM uses the kernel method to transfer it onto a higher-dimensional space into which separation is easier (Analytics Vidhya, 2021). A data point is classified using K-nearest neighbors (KNN) (Wikipedia, 2024) by a majority vote of its neighbors, and it is given to the class that is most prevalent among its K nearest neighbors. Based on Bayes' Theorem, Naïve Bayes (NB) (Alpaydin, 2010) classifiers employ a probabilistic methodology that creates posterior probabilities by integrating prior probability distributions. A traditional multilayer perceptron (MLP) (Bishop, 1995) is another method we take into consideration as a classifier. The advantages of an MLP include learning non-linear models, managing massive amounts of input data, training models in real-time (online learning), and producing predictions quickly after training. It is more computationally expensive, though. More susceptible to feature scaling than other classifiers. Without using implementations of deep learning techniques, we employ the scikit-learn library's default version of MLP.

2.4 Evaluation Metrics

We employed 5-fold cross validation to assess the effectiveness of our methods. Five equal-sized sub-samples were randomly selected from the original sample in order to perform 5-fold cross validation. Only one subsample was kept for analysis. while training was conducted with the remaining four. The procedure was carried out five times, with a different subsample being used for testing each time. After that, a single estimate was created by averaging the results. The metrics we used for the evaluation of the algorithms are Accuracy, F1-score, recall, precision and AUC_ROC.

In this research, we use the following terms: A true positive (TP) is when a malicious app is accurately classified as malicious, a true negative (TN) is when a benign app is classified as benign, a false positive (FP) is when a benign app is classified as malicious, and a false negative (FN) When a malicious app is categorized as benign. The model's fraction of accurate predictions is expressed by the accuracy (Accuracy) evaluation metric, which is provided by

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Accuracy can be deceptive in highly uneven situations, and alternative metrics are employed for evaluation. Precision often known as the positive predictive value, is provided by

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Recall, commonly referred to as the true positive rate (TPR) or sensitivity, is calculated as

$$\text{Recall}=\frac{TP}{TP+FN} \quad (3)$$

the precision and recall metrics' harmonic mean yields the F1-score. which is calculated as

$$\text{F1-score}=\frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

3. RELATED WORK

Our work's literature review part looks at a number of published publications on malware detection with machine learning algorithms.

Proposes MaLDroide, a machine learning framework that uses a Support Vector Machine classifier to detect Android malware with 99.8% accuracy (García, 2015). Suggests an anomaly-based malware detection framework for the Android platform. It analyses behavioural patterns and tests different machine learning algorithms using a dataset of malicious and benign apps. The best methods for malware detection are Random Forest and Support Vector Machine (Ali et al., 2017). Uses and assesses two machine learning methods, K-Nearest Neighbours (KNN) and Support Vector Machine (SVM), for Android malware detection. It discovers that KNN has a true positive rate of more than 80.00% and an average accuracy of 80.50% (Kakavand et al., 2018). Based on static analysis of application data, suggests an intelligent model that uses machine learning algorithms to identify malicious apps on Android handsets (Fiky et al., 2021). Introduces a machine learning-based method for identifying Android malware by using the NATICUSdroid (Android Permission) Dataset to extract features like permissions requested by Android apps and train different machine learning models. Random Forest achieved the highest accuracy of 97.20% (Iqubal & Payal, 2024). Presents a static-based machine learning approach for detecting Android malware using permissions and API calls as features, and evaluates the performance of three machine learning algorithms (SVM, KNN, and Naive Bayes) on a new Android malware dataset (Shatnawi et al., 2022). Provide a framework that achieves 99.34% accuracy in early Android malware detection through the use of machine learning and system call analysis (Zhang 2022). Suggests a machine learning-based method for identifying Android malware that uses contextual features from the program in addition to permissions and API calls. It demonstrates that the accuracy of detecting Android malware using the chosen contextual features is extremely high, at around 99.4% (AlJarrah et al., 2022). Offers a thorough investigation of 27 distinct machine learning models as well as a deep neural network model for Android malware detection. Hyperparameter tweaking and feature importance analysis are used to determine which model is the most accurate and comprehensible (Giannakas et al., 2022). Created a framework that uses machine learning methods to accurately identify and categorise various malware file types as harmful or benign (Sethi et al., 2017). Offers a framework for applying machine learning algorithms to malware detection and assesses the effectiveness of four distinct algorithms (Support Vector Machine, Decision Tree, Random Forest, and Gradient Boosting) using metrics like area under the ROC curve, execution time, accuracy, and false positive/negative rates (Rkhouya & Chougdali, 2021). Offers both static and dynamic analysis for malware detection using machine learning algorithms, especially decision tree-based techniques like random forests, with dynamic analysis offering a higher classification accuracy of up to 95.95% (Palša et al., 2022). Outlines the potential of machine learning approaches to improve the accuracy and effectiveness of malware detection systems (Almuqren et al., 2023). Offers a unique method for detecting malware that separates malware from clean files with 97% accuracy by leveraging static information taken from Windows PE files, such as function call frequency and opcode frequency (Singla et al., 2015). Proposes a feature selection technique for malware detection that, in comparison to previous approaches, can boost the detection rate while lowering the system's complexity (Jiang et al., 2011). The features chosen by the suggested ensemble feature selection approach are representational (best represent the total data) and discriminative (most differentiable between malware and benign). For malware identification, a thorough metric is created to preserve the most instructive properties (Zhang et al., 2016). Support Vector Machines (SVM) were used for classification and Principal Component Analysis (PCA) for feature selection, which yielded the highest malware detection accuracy with the fewest features (Khammas et al., 2015). Developed and analysed a collection of clean and malicious files, discovered that nine features could differentiate malware from goodware with 99.60% accuracy, and employed a unique set of classification algorithms that improved training speed and accuracy (Cepeda et al., 2016). The malware detection models' computational cost

Research Article

(time and memory) was greatly decreased by feature selection utilising information gain without sacrificing accuracy (Rahman et al., 2023). The accuracy of Android malware detection is increased by presenting a multi-tiered feature selection approach, with Random Forest classification attaining the greatest accuracy of 96.28% (Bhat & Dutta, 2022). On two distinct datasets, the XGB classifier and the suggested hybrid feature selection method produced malware detection accuracies of 99.26% and 98.9%, respectively (Roseline & Geetha, 2019). Proposed DynaMalDroid, a framework based on dynamic analysis for machine learning-based Android malware detection, with 99.3% and 99.5% detection accuracies for the Support Vector Machine and AdaBoost classifiers, respectively (Manzil et al. 2022). Identifies various Windows malware, such as Trojan, Backdoor, Trojan Downloader, Trojan Dropper, Virus, and Worm, by using machine learning algorithms to extract both static and dynamic information from malware samples. The accuracy of the static analysis strategy in identifying malware was greater (99.36%) than that of the dynamic analysis approach (94.64%) (Dr et al., 2023). Malware can be swiftly detected using the suggested deep learning and malware image approach, which eliminates the requirement for conventional static or dynamic analysis (Choi et al., 2017). Utilizing stacked autoencoders derived from Windows API calls, a deep learning framework enhances malware detection performance in contrast to conventional shallow learning techniques (Hardy et al., 2016). Proposed DroidDetector, a deep learning-based Android malware detection engine that has a 96.76% accuracy rate in automatically determining whether an Android app is malicious (Yuan et al., 2016). Offers a deep learning-based malware detection system that outperforms earlier findings on the Malicia dataset, achieving high accuracy and low false positive rates without the requirement for custom feature engineering (Sewak et al., 2018). To determine if a particular portable executable (PE) file is malware or not, a deep neural network-based solution is suggested (Mane et al., 2022). Suggests a novel deep learning-based method for malware detection that can successfully identify novel, unidentified malware samples without the need for laborious feature engineering (Kumar & Bagane, 2020). A hybrid cloud-based malware detection solution that uses deep learning models that continuously learn from customer data, combining static and dynamic analysis to balance detection accuracy and reaction time (De Paola et al., 2018). A hybrid machine learning method for malware detection that enhances overall detection accuracy by utilising ensemble approaches and data preparation strategies (Bandlapalli et al., 2024). Suggest a novel hybrid machine learning technique to more accurately detect, categorise, and identify altered malware traces automatically than with conventional techniques (Yang et al., 2015). Suggests a hybrid machine learning model that achieves excellent malware detection rates with low false positives by combining deep learning with random forest (Yoo et al., 2021). Uses a dataset of 10,000 harmful and benign files with 78 features and 6 malware classes to demonstrate a hybrid machine learning approach for malware detection and classification ((Eid & Allah 2022). Suggested model outperformed five other cutting-edge methods in malware detection by using a random forest, with an accuracy of 99.7% (Kumar et al., 2020). Gives a methodical and thorough review of machine learning approaches, particularly deep learning, for malware detection and classification. It covers the features and methods employed, the difficulties and constraints of conventional machine learning, and the most recent advancements and trends in the field (Gibert, 2020). Suggest a novel CNN-based IoT malware detection architecture (iMDA) that enhances malware detection capabilities on IoT devices by integrating many feature learning techniques (Asam & Kamble 2022). Demonstrate that IoT malware differs from Android malware in its fundamental features by utilising deep learning and control flow graphs to develop a detection technique for IoT malware (Alasmary & Ravi 2019). Using ELF binary file byte sequences as input features, a deep learning-based Bi-GRU-CNN model was presented to detect and categorise IoT malware. The model's accuracy was 100% for IoT malware detection and 98% for IoT malware family classification (Chaganti et al., 2022). Offers a framework for classifying and detecting malware on Internet of Things devices that employs machine learning methods to identify and detect different kinds of malware via static analysis (Khirid et al., 2022). Investigates the detection of Android malware in Internet of Things environments using machine learning approaches such as Naive Bayes, K-Nearest Neighbors, Decision Tree, and Random Forest; the Decision Tree model has the best accuracy of 95% (Sharma & Babbar, 2023). Identify frequent opcode sequences in malicious IoT applications using sequential pattern mining. Then, assess how well these maximal frequent patterns (MFPs) work as a classification feature for different machine learning algorithms to accurately identify hidden IoT malware, including polymorphic malware (Darabian et al., 2020). Suggests a machine learning method for identifying malware in Android apps, emphasizing methods for data pre-processing, dimensionality reduction, and classification. The method is tested on both real-world Android apps and public domain datasets (Palma et al. 2024). Examines and contrasts the performance and resource consumption of many machine learning techniques, such as

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Random Forest and AdaBoost, for the classification of Android malware in the CICAndMal2017 dataset (Ferreira et al., 2021). Suggests a thorough framework for detecting and identifying Android malware using machine learning algorithms. The best-performing models are Random Forest and XGBoost, which yield high detection rates with little false positives (Iqubal et al., 2024). An explainable machine learning-based lightweight Android malware detection solution that uses SHAP (Shapley Additive Explanation) values to explain the model is capable of detecting dangerous and benign apps with over 98% accuracy while leaving a minimal impact on the device (Alani & Awad, 2022). Using static analysis of function call graph APIs, this explainable method for Android malware detection links the behaviour of the virus to its family and offers insightful justifications for the model's conclusions (Soi et al., 2024).

4. PROPOSED WORK

The technique of the suggested approach is introduced and discussed in this part. The approach is illustrated in Figure 2, and the ensuing subsections go over each stage.

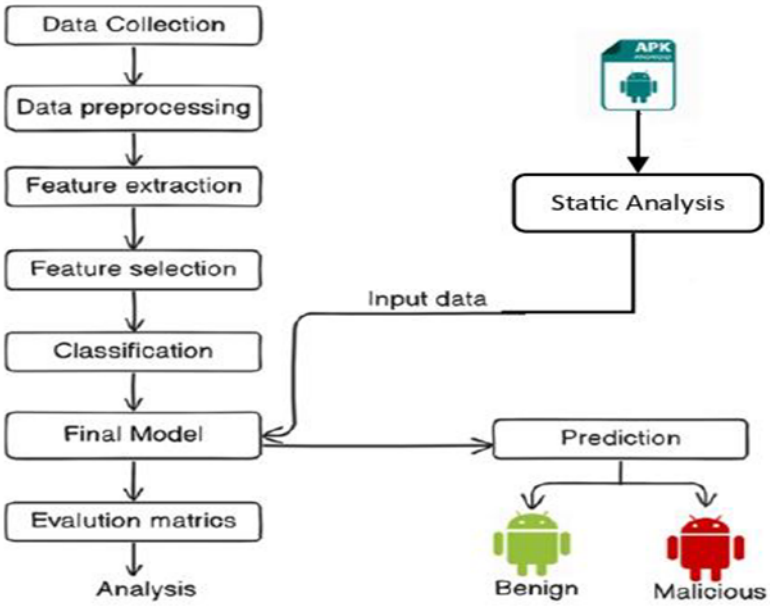

Fig-2 A framework for Android malware detection approach using machine learning techniques with static analysis.

4.1 Data collection and Data preprocessing

We conducted our research using the Drebin Dataset. 9476 benign apps and 5560 malicious apps from 179 distinct malware families make up the Drebin dataset, which was originally made public in 2014. It includes 215 features that were taken from 15,036 applications. The Drebin dataset only has static features.

A binary classification issue is used to design the Android malware detection task. We consider a malicious app as a positive sample and a benign app as a negative sample. To appropriately prepare the data and evaluate their effect on the model's performance, data preprocessing techniques are used. There are four broad categories into which data pre-processing may be divided: transformation, reduction, integration, and cleaning (Baheti). Data cleaning involves dealing with missing information, which can be accomplished in a number of ways, including eliminating examples with missing values or imputation of missing values. Additionally, it covers attribute conversions like label encoding as well as reformatting the data to guarantee common forms. Finding outliers and smoothing noisy data are two aspects of data cleaning. Combining information from several sources into a single dataset is known as data integration. In order to preserve the integrity of the original data, data reduction techniques seek to produce a reduced representation in terms of volume. Dimensionality reduction and numerosity reduction—which includes instance sampling—are the two primary methods for data reduction. One technique for balancing unbalanced data is

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

instance sampling. For sampling, we employed random oversampling, which balanced the data by duplicating minority class samples. Finally, data transformation seeks to modify the format, value, or structure of the data in order to mold it into a suitable form. The two most popular methods are discretization and normalization. We used the min–max normalization technique for data transformation.

Algorithm 1 ConvertCategoricalFeaturesToNumeric

```
Require: dataset, num_rows, num_cols
    for col = 0 to num_cols - 1 do
      if !(is_numeric_string(dataset[0][col])) then
          mapping_count = 0
        for row = 0 to num_rows - 1 do
            value = dataset[row][col]
          if value is not in mapping then
              mapping[value] = mapping_count
              mapping_count = mapping_count + 1
          end if
        code = mapping[value]
        dataset[row][col] = string representation of code
      end for
    end if
  end for
 for row = 0 to num_rows - 1 do
   for col = 0 to num_cols - 1 do
       numeric_value ← convert dataset[row][col] to integer
      if numeric_value < 0 then
          dataset[row][col] ← "NaN"
       end if
    end for
  end for
```

4.2 Feature extraction and selection

Feature selection methods like the relevance-redundancy feature selection (RRFS) filter approach were used to reduce dimensionality. With RRFS, redundant and weakly relevant features are eliminated while the relevant ones that enhance the model's value are retained. Two distinct measures of relevance were tested: the unsupervised measure MM(Mean-Median) and the supervised measure FR (Fisher's Ratio). With the FR measure of the RRFS method, a subset of the most relevant features is acquired. The dataset's redundant features were eliminated using the Absolute Cosine (AC) redundancy measure. The AC was employed as a redundancy measure, and a maximum similarity (ms) of 0.3 was permitted between successive feature pairs.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Algorithm 2 RelevanceRedundancyFeatureSelection

Require: dataset, class_name, relevance_measure, ms

```
if relevance_measure == "FR" then
    dataset ← relevancemeasure_FR(dataset, class_name)
else if relevance_measure == "MM" then
    dataset ← relevancemeasure_MM(dataset)
else
    print  “error”
    exit
end if
    dataset ← RedundancyMeasure_AC(dataset, ms)
```

After the completion of data preprocessing and feature selection, data were split into two parts: training and testing, based on stratified K-fold CV method. Stratified K-fold CV uses stratified sampling to divide the data into K folds of roughly similar size. where K is equal to the number of instances. The value of K in our study was 5. With the help of the training set, the model is trained to predict whether an application is malicious or benign based on input data. We used 5 different machine learning classifier SVM, KNN, NB, RF, and MLP to train our model. The testing set makes it possible to analyse the model using common evaluation measures.

We also test our model on a real-time android application. We train our ML model using a dataset and then we evaluate a real-time app with that model. The created method was further tested using APKs that were found online.

5. EXPERIMENTAL RESULTS

We now present the experimental assessment procedure, which was carried out with Python and the "scikit-learn" machine learning library. We have taken into account the classifiers indicated in Section 2.3 as well as the evaluation measures outlined in Section 2.4.

The Drebin dataset was used for the research. Only static features are taken into consideration because the suggested method is based on static analysis. The Drebin dataset contained 15,036 applications with 215 features. Among them, 9476 are benign apps and 5560 are malicious apps.

To determine whether there were any instances of severe imbalance, the class distribution in the datasets was examined. The ratio between class labels in the Drebin datasets is roughly one-third, thus while it is not quite balanced, it cannot be regarded as imbalanced. Dataset contains 1 categorical and 214 non-categorical features. The dataset contains no missing values. So, less data pre-processing tasks were required.

The number of features in the dataset was greatly decreased by the RRFS technique. Dimensionality was reduced more significantly by the supervised relevance measure FR than by the unsupervised relevance measure MM. A subset of the most relevant features is acquired using the FR measure. It reduces the dataset’s dimensionality, with a reduction of 56%. The original number of features in the dataset was 215, and the number of features after applying the RRFS approach was 94. Below are the features that were selected by RRFS (FR).

'transact' 'SEND_SMS' 'Ljava.lang.Class.getCanonicalName'
'android.telephony.SmsManager' 'Ljava.lang.Class.getField' 'RECEIVE_SMS'
'Ljavax.crypto.spec.SecretKeySpec' 'WRITE_SMS' 'READ_SYNC_SETTINGS'
'TelephonyManager.getSubscriberId' 'CAMERA' 'AUTHENTICATE_ACCOUNTS'

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

'Ljava.lang.Class.forName' 'INSTALL_PACKAGES' 'READ_HISTORY_BOOKMARKS'
'android.telephony.gsm.SmsManager' 'android.intent.action.PACKAGE_REPLACED' 'Binder'
'android.intent.action.SEND_MULTIPLE' 'abortBroadcast' 'URLClassLoader'
'ACCESS_LOCATION_EXTRA_COMMANDS' 'NFC' 'MODIFY_AUDIO_SETTINGS'
'WRITE_APN_SETTINGS' 'android.intent.action.TIME_SET' 'BROADCAST_STICKY'
'BIND_REMOTEVIEWS' 'android.intent.action.PACKAGE_REMOVED'
'getCallingPid' 'READ_PROFILE' 'READ_SYNC_STATS' 'createSubprocess'
'WAKE_LOCK' 'android.intent.action.TIMEZONE_CHANGED' 'RESTART_PACKAGES'
'android.intent.action.PACKAGE_ADDED' 'chmod' 'Ljava.lang.Class.getDeclaredClasses'
'android.intent.action.ACTION_POWER_DISCONNECTED'
'TelephonyManager.getSimSerialNumber' 'PathClassLoader'
'TelephonyManager.getCallState' 'BLUETOOTH' 'READ_CALENDAR'
'READ_EXTERNAL_STORAGE' 'Runtime.load' 'TelephonyManager.getSimCountryIso' 'READ_CALL_LOG'
'SUBSCRIBED_FEEDS_WRITE' 'sendMultipartTextMessage' 'HttpPost.init'
'PackageInstaller' 'android.intent.action.ACTION_SHUTDOWN' 'remount'
'Ljava.lang.Class.getClasses' 'TelephonyManager.isNetworkRoaming'
'sendDataMessage' 'WRITE_CALENDAR' 'SUBSCRIBED_FEEDS_READ' 'chown'
'HttpUriRequest' 'CHANGE_WIFI_MULTICAST_STATE' 'MASTER_CLEAR'
'DELETE_PACKAGES' 'GET_TASKS' 'android.intent.action.PACKAGE_DATA_CLEARED'
'UPDATE_DEVICE_STATS' 'GLOBAL_SEARCH' 'WRITE_CALL_LOG'
'android.intent.action.PACKAGE_CHANGED' 'REORDER_TASKS' 'DELETE_CACHE_FILES'
'android.intent.action.NEW_OUTGOING_CALL' 'SET_WALLPAPER' 'divideMessage'
'WRITE_USER_DICTIONARY' 'BIND_INPUT_METHOD' 'Runtime.exec'
'WRITE_PROFILE' 'PROCESS_OUTGOING_CALLS' 'BIND_WALLPAPER'
'CALL_PRIVILEGED' 'BATTERY_STATS' 'READ_USER_DICTIONARY'
'ACCESS_COARSE_LOCATION' 'READ_SOCIAL_STREAM' 'RECEIVE_WAP_PUSH'
'android.intent.action.SENDTO' 'WRITE_SETTINGS' 'DUMP'
'TelephonyManager.getNetworkOperator' 'SET_TIME' '/system/bin'

Fig-3 Most Relevant Features selected from dataset using RRFS method

In addition to reducing dimensionality, RRFS makes it possible to identify the most relevant features for Android app malware detection, which is crucial for the suggested methodology. The most relevant features in the dataset are permissions and methods.SEND_SMS and android.telephony.SmsManager are two of the most important features for Android malware detection. According to our findings, permissions and SMS-related functionality are the most telling indicators of malware in Android apps. After selecting the most relevant features, we tested our model for different classifiers (MLP, NB, RF, SVM, and KNN) and calculated different evaluation metrics for each classifier. Table-1 shows the results of all evaluation metrics.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table-1 Evaluation Metrics obtained with each classifier (MLP, NB, RF, SVM, and KNN)

Classifier	Accuracy(%)	F1-Score(%)	Precision(%)	Recall(%)	AUC_ROC(%)
MLP	97.94	97.94	98.14	97.74	97.94
NB	92.87	92.91	92.54	93.32	92.87
RF	97.99	97.99	98.25	97.73	97.99
SVM	97.63	97.62	98.01	97.22	97.63
KNN	93.60	93.92	89.74	98.84	93.60

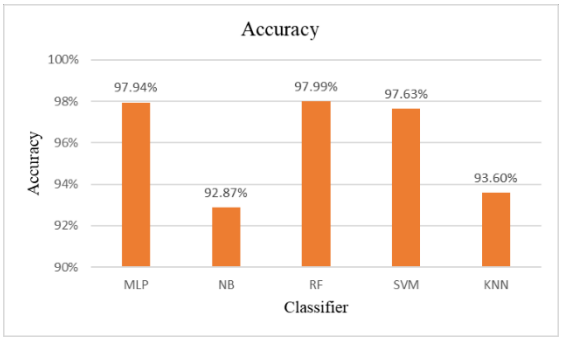

Fig-4 Accuracy (%) obtained with each classifier

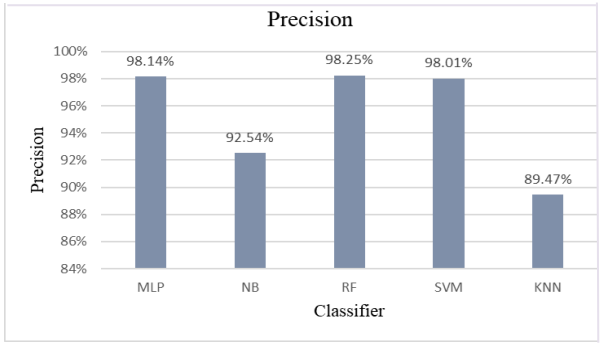

Fig-5 Precision (%) obtained with each classifier

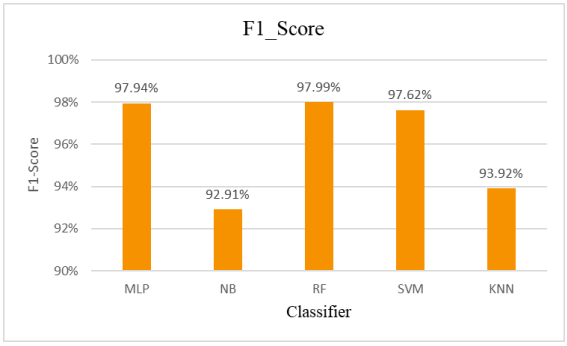

Fig-6 F1-score (%) obtained with each classifier

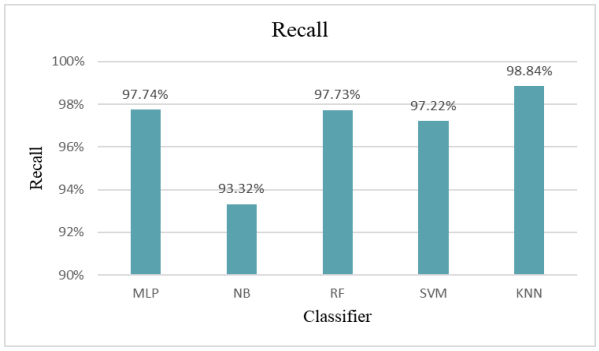

Fig-7 Recall (%) obtained with each classifier

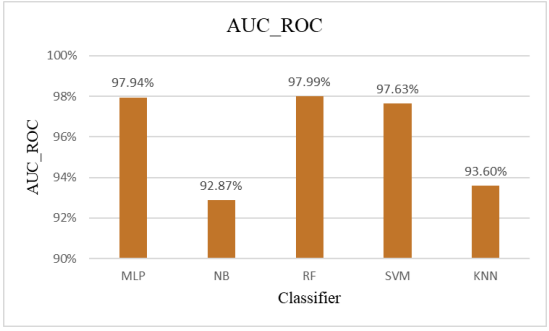

Fig-8 AUC-ROC (%) obtained with each classifier

The results obtained for the dataset and five distinct classifiers in terms of accuracy, Precision, F1-score, Recall, and AUC-ROC metrics are summarised in Figures 4–8 respectively.

APKs from the internet were used to test the developed method further. We got the APK for testing purposes from apkmirror. The APK file 'com.microsoft.bing_30.1.421216001-421216001_minAPI26(armeabi-v7a)_apkmirror.com.apk' was used to evaluate the model. Androguard is a Python library and utility for working with

Research Article

Android files that makes it possible to extract features from Android app files. As a result, this module takes static features out of the APK file. Then we had mapped these extracted features and most relevant features, which can then classify/predict the Android application as benign or malicious. We correctly classified it as a benign app.

Furthermore, some of the malware samples that were evaluated employed obfuscation techniques, which are known to be a static analysis weakness. Additionally, the datasets used have a significant influence on the prediction that is produced.

6. CONCLUSION AND FUTURE SCOPE

Malware in Android apps impacts millions of people globally and is always changing. Therefore, detecting it is a current and pertinent issue. ML techniques have been suggested as a way to reduce malware in mobile applications in recent years. The proposed framework detects malware from Android apps by performing its static analysis. In order to identify malware in real-world applications, we used feature selection techniques to train our suggested framework. This challenge was designed as a binary classification problem using the Drebin public domain dataset. Firstly, we perform data preprocessing on the dataset to clean data, reduce dimensionality, and remove data redundancy. After that, we extract static features from the dataset and select relevant features to train the model. Experiments were performed with SVM, RF, NB, KNN, and MLP classifiers. Results showing that the RF classifier gives the best results among all the classifiers.

The proposed approach can identify the most important features to categorize an app as malware and greatly decrease the data dimensionality while obtaining good results in detecting malware in Android applications across the different evaluation metrics.

In future, the proposed method might be expanded to hybrid analysis to solve this issue. More recent datasets should be made accessible and utilised or should use DL approaches.

REFERENCES

[1] AppBrain. (2024). *Number of Android apps on Google Play*. Retrieved October 19, 2024, from https://www.appbrain.com/stats/number-of-android-apps

[2] Bhatia, T., & Kaushal, R. (2017). Malware detection in Android based on dynamic analysis. *Proceedings of the 2017 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, London, UK, 19–20 June 2017, pp. 1–6.

[3] Dassanayake, D. (2021). Millions of Android phones infected by dangerous malware. These phones are at risk. *Express*. Retrieved October 19, 2024, from https://www.express.co.uk/life-style/science-technology/1527645/Millions-Android-phones-infected-dangerous-malware-these-phones-at-risk

[4] Kapratwar, A., Di Troia, F., & Stamp, M. (2017). Static and dynamic analysis of Android malware. *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, Porto, Portugal, 19–21 February 2017.

[5] Kaspersky. (2020). Mobile security: Android vs. iOS - which one is safer? Retrieved October 19, 2024, from https://www.kaspersky.com/resource-center/threats/android-vs-iphone-mobile-security

[6] Kosmidis, K., & Kalloniatis, C. (2017). Machine learning and images for malware detection and classification. *Proceedings of the 21st Pan-Hellenic Conference on Informatics*.

[7] StatCounter. (2024). *Mobile operating system market share worldwide*. Retrieved October 19, 2024, from https://gs.statcounter.com/os-market-share/mobile/worldwide

[8] Statista. (2024). Number of smartphone mobile network subscriptions worldwide from 2016 to 2023, with forecasts from 2023 to 2028 (in millions). Retrieved October 19, 2024, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[9] Timothy, M. (2022). Why you should uninstall your Android antivirus software. Retrieved October 19, 2024, from https://www.makeuseof.com/uninstall-android-antivirus/

[10] Townsend, K. (2020). How smartphones have become one of the largest attack surfaces.

[11] Palma, C., Ferreira, A., & Figueiredo, M. (2024). Explainable machine learning for malware detection on Android applications. *Information, 15*(1), 25.

[12] Faruki, P., Bharmal, A., Ganmoor, V., Laxmi, V., Gaur, M. S., & Zia, T. (2014). Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys & Tutorials, 17*(2), 998–1022. https://doi.org/10.1109/COMST.2014.2368999
[13] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324
[14] Analytics Vidhya. (2021, October). *Support Vector Machines (SVM) – A complete guide for beginners*. Retrieved October 20 , 2024, from https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
[15] Wikipedia. (2024). *K-nearest neighbors algorithm*. Retrieved October 20, 2024, from https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[16] Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed.). The MIT Press.
[17] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.
[18] Ali, M. A., Svetinovic, D., Aung, Z., & Lukman, S. (2017). Malware detection in android mobile platform using machine learning algorithms. 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), 763–768. https://doi.org/10.1109/ICTUS.2017.8286109
[19] AlJarrah, M., Yaseen, Q., & Mustafa, A. (2022). A context-aware Android malware detection approach using machine learning. Information, 13(12), 563. https://doi.org/10.3390/info13120563
[20] Fiky, A. H. E., Elshenawy, A., & Madkour, M. A. (2021). Detection of Android malware using machine learning. 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 9–16. https://doi.org/10.1109/MIUCC52538.2021.9447661
[21] García, B. T. (2015). A framework for detection of malicious software in Android handheld systems using machine learning techniques. Semantic Scholar. Retrieved October 20, 2024, from https://www.semanticscholar.org/paper/A-framework-for-detection-of-malicious-software-in-Garc%C3%ADa/5f41c797f3006bca6b045f47a73f8e22c31208e4
[22] Giannakas, F., Kouliaridis, V., & Kambourakis, G. (2022). A closer look at machine learning effectiveness in Android malware detection. Information, 14(1), 2. https://doi.org/10.3390/info14010002
[23] Iqubal, A., & Payal, A. (2024). Malware detection technique for Android devices using machine learning algorithms. 2024 International Conference on Computing, Sciences and Communications (ICCSC), 1–6. https://doi.org/10.1109/ICCSC62048.2024.10830310
[24] Kakavand, M., Dabbagh, M., & Dehghantanha, A. (2018). Application of machine learning algorithms for Android malware detection. Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems, 32–36. https://doi.org/10.1145/3293475.3293489
[25] Sethi, K., Chaudhary, S. K., Tripathy, B. K., & Bera, P. (2017). A novel malware analysis for malware detection and classification using machine learning algorithms. Proceedings of the 10th International Conference on Security of Information and Networks, 107–113. https://doi.org/10.1145/3136825.3136883
[26] Shatnawi, A. S., Yassen, Q., & Yateem, A. (2022). An Android malware detection approach based on static feature analysis using machine learning algorithms. Procedia Computer Science, 198, 653–658. https://doi.org/10.1016/j.procs.2022.03.086
[27] Zhang, X., et al. (2022). An early detection of Android malware using system calls-based machine learning model. Proceedings of the 17th International Conference on Availability, Reliability and Security, 1–9. https://doi.org/10.1145/3538969.3544413
[28] Bhat, P., & Dutta, K. (2022). A multi-tiered feature selection model for android malware detection based on feature discrimination and information gain. *Journal of King Saud University - Computer and Information Sciences, 34*(10), 9464–9477. https://doi.org/10.1016/j.jksuci.2021.11.004
[29] Cepeda, C., Chia Tien, D. L., & Ordóñez, P. (2016). Feature selection and improving classification performance for malware detection. *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, 560–566. https://doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.87

[30] Jiang, Q., Zhao, X., & Huang, K. (2011). A feature selection method for malware detection. *2011 IEEE International Conference on Information and Automation*, 890–895. https://doi.org/10.1109/ICINFA.2011.5949122

[31] Khammas, B. M., Monemi, A., Bassi, J. S., Ismail, I., Mohd Nor, S., & Marsono, M. N. (2015). Feature selection and machine learning classification for malware detection. *Jurnal Teknologi*. https://doi.org/10.11113/jt.v77.3558

[32] Palša, J., Hurtuk, J., Chovanec, M., & Chovancová, E. (2022). Using machine learning algorithms to detect malware by applying static and dynamic analysis methods. *Acta Polytech Hung, 19*(7), 177–196. https://doi.org/10.12700/APH.19.7.2022.7.10

[33] Rahman, M. A., Islam, S., Nugroho, Y. S., Al Irsyadi, F. Y., & Hossain, M. J. (2023). An exploratory analysis of feature selection for malware detection with simple machine learning algorithms. *Journal of Communications Software and Systems, 19*(3), 207–219.

[34] Rkhouya, S., & Chougdali, K. (2021). Malware detection using a machine-learning based approach. *International Journal of Information Technology and Applied Sciences (IJITAS, 3*(4), 167–171.

[35] Singla, S., Gandotra, E., Bansal, D., & Sofat, S. (2015). A novel approach to malware detection using static classification. *International Journal of Computer Science and Information Security, 13*(3), 1–5.

[36] Zhang, X.-Y., Wang, S., Zhang, L., Zhang, C., & Li, C. (2016). Ensemble feature selection with discriminative and representative properties for malware detection. *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 674–675. https://doi.org/10.1109/INFCOMW.2016.7562161

[37] Almuqren, A., Frikha, M., & Albuali, A. (2023). Automated malware detection based on a machine learning algorithm. *2023 IEEE Tenth International Conference on Communications and Networking (ComNet)*, 1–12. https://doi.org/10.1109/ComNet60156.2023.10366550

[38] De Paola, A., Gaglio, S., Re, G. L., & Morana, M. (2018). A hybrid system for malware detection on big data. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 45–50. https://doi.org/10.1109/INFCOMW.2018.8406963

[39] Dr, K. D., Shanthi, A., Kokila, G., & Batcha, J. (2023). Machine learning is used for static and dynamic malware analysis. *Semantic Scholar*. Retrieved October 8, 2024, from https://www.semanticscholar.org/paper/MACHINE-LEARNING-IS-USED-FOR-STATIC-AND-DYNAMIC-Dr-Shanthi/786e02dcfaec4e3ca22bd11aea92f95fbb44d65b

[40] Choi, S., Jang, S., Kim, Y., & Kim, J. (2017). Malware detection using malware image and deep learning. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 1193–1195. https://doi.org/10.1109/ICTC.2017.8190895

[41] Hardy, W., Chen, L., Hou, S., Ye, Y., & Li, X. (2016). DL 4 MD: A deep learning framework for intelligent malware detection. *Semantic Scholar*. Retrieved October 11, 2024, from https://www.semanticscholar.org/paper/DL-4-MD-%3A-A-Deep-Learning-Framework-for-Intelligent-Hardy-Chen/a6cc849f7c691e232360e9b71da0e431af2aa1e3

[42] Kumar, G. S., & Bagane, P. (2020). Detection of malware using deep learning techniques. *International Journal of Scientific & Technology Research*. Retrieved October 11, 2024, from https://www.semanticscholar.org/paper/Detection-Of-Malware-Using-Deep-Learning-Techniques-Kumar-Bagane/305357688d483af70387bc4221d747ddd562d28a

[43] Manzil, H. H. R., & Naik, M. S. (2022). DynaMalDroid: Dynamic analysis-based detection framework for Android malware using machine learning techniques. *2022 International Conference on Knowledge Engineering and Communication Systems (ICKES)*, 1–6. https://doi.org/10.1109/ICKECS56523.2022.10060106

[44] Mane, T., Nimase, P., Parihar, P., & Chandankhede, P. (2022). Review of malware detection using deep learning. *Springer Advances in Science, Technology & Innovation*, 1397, 255–262. https://doi.org/10.1007/978-981-16-5301-8_19

[45] Roseline, S. A., & Geetha, S. (2019). An efficient malware detection system using hybrid feature selection methods. *International Journal of Engineering and Advanced Technology*, 9(1S3), 224–228.

[46] Sewak, M., Sahay, S. K., & Rathore, H. (2018). An investigation of a deep learning-based malware detection system. *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 1–5. https://doi.org/10.1145/3230833.3230835

[47] Yuan, Z., Lu, Y., & Xue, Y. (2016). DroidDetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, 21(1), 114–123.

[48] Asam, M., ..., & Kamble, A. R. (2022). IoT malware detection architecture using a novel channel boosted and squeezed CNN. *Scientific Reports, 12*(1), 15498.

[49] Alasmary, H., ..., & Ravi, V. (2019). Analyzing and detecting emerging Internet of Things malware: A graph-based approach. *IEEE Internet of Things Journal, 6*(5), 8977–8988. https://doi.org/10.1109/JIOT.2019.2925929

[50] Bandlapalli, S., Janarthan, S. N., Ragul, S., & Sujatha, G. (2024). Identifying malware using machine learning ensemble model. *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI),* 1–6. https://ieeexplore.ieee.org/abstract/document/10602181/

[51] Chaganti, R., Ravi, V., & Pham, T. D. (2022). Deep learning-based cross-architecture internet of things malware detection and classification. *Computers & Security, 120*, 102779. https://doi.org/10.1016/j.cose.2022.102779

[52] Eid, M. M., ..., & Allah, M. I. F. (2022). Detection and classification of malware using guided whale optimization algorithm for voting ensemble. *JCIM, 10*(1), 34–42. https://doi.org/10.54216/JCIM.100102

[53] Gibert, D. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends, and challenges. *Journal of Network and Computer Applications, 153*. https://doi.org/10.1016/j.jnca.2019.102526

[54] Khirid, S., Veer, S., Gupta, T., Waychal, V., & Kamble, A. R. (2022). Malware detection and classification framework for IoT devices. *IJARSCT*, 1–8. https://doi.org/10.48175/IJARSCT-3877

[55] Kumar, A., Abhishek, K., Shandilya, S. K., & Ghalib, M. R. (2020). Malware analysis through random forest approach. *JWE*. https://doi.org/10.13052/jwe1540-9589.195610

[56] Yang, R. R., Kang, V., Albouq, S., & Zohdy, M. A. (2015). Application of hybrid machine learning to detect and remove malware. *Transactions on Machine Learning and Artificial Intelligence, 3*(4), 16.

[57] Yoo, S., Kim, S., Kim, S., & Kang, B. B. (2021). AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. *Information Sciences, 546*, 420–435. https://doi.org/10.1016/j.ins.2020.08.082

[58] Alani, M. M., & Awad, A. I. (2022). Paired: An explainable lightweight android malware detection system. *IEEE Access, 10*, 73214–73228.

[59] Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., & Choo, K. R. (2020). An opcode-based technique for polymorphic Internet of Things malware detection. *Concurrency and Computation, 32*(6), e5173. https://doi.org/10.1002/cpe.5173

[60] Ferreira, A. P., Gupta, C., Inácio, P. R. M., & Freire, M. M. (2021). Behaviour-based malware detection in mobile Android platforms using machine learning algorithms. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 12*(4), 62–88. https://doi.org/10.22667/JOWUA.2021.12.31.062

[61] Iqubal, A., Happy, Tiwari, S. K., Azad, S., & Paswan, M. K. (2024). Android-based malware detection technique using machine learning algorithms. *Proceedings of the First International Conference on Pioneering Developments in Computer Science & Digital Technologies (IC2SDT)*, 1–6. https://doi.org/10.1109/IC2SDT62152.2024.10696330

[62] Palma, C., Ferreira, A., & Figueiredo, M. (2024). Explainable machine learning for malware detection on Android applications. *Information, 15*(1), 25.

[63] Sharma, A., & Babbar, H. (2023). An analysis of Android malware and IoT attack detection with machine learning. *Proceedings of the 3rd International Conference on Intelligent Technologies (CONIT)*, 1–5. https://doi.org/10.1109/CONIT59222.2023.10205931

[64] Soi, D., Sanna, A., Maiorca, D., & Giacinto, G. (2024). Enhancing Android malware detection explainability through function call graph APIs. *Journal of Information Security and Applications, 80*, 103691. https://doi.org/10.1016/j.jisa.2023.103691

[65] Baheti, P. (2021, August 31). A simple guide to data preprocessing in machine learning. *V7 Labs*. Retrieved October 2, 2024, from https://www.v7labs.com/blog/data-preprocessing-guide