Android Malware Detection Project

Team-12

Teja Nagireddy
Arizona State University
1226129686
tnagire1@asu.edu

Lushaank Kancherla
Arizona State University
1224601664
lkancher@asu.edu

Briana Rajan
Arizona State University
1225414593
brajan3@asu.edu

ABSTRACT

With the rapid development of the Android operating system, malware is spreading and affecting an increasing number of mobile users worldwide. Malware has evolved over time and now comes in a variety of forms and functions. To detect and remove malware, current signature-based methods are used.

Such detection techniques, however, have limitations. Signature-based solutions cannot detect newly released malware with high accuracy, and the detection process is time consuming. As a result, new detection methods are proposed to address current issues. Machine learning techniques are widely regarded as the most effective and efficient malware detection methods. Several machine learning methods will be used in this project to detect malware based on permission usage features:

KEYWORDS

Malware Detection, Machine Learning, Classification, F1 Score, Recall, Precision, confusion matrix

1. INTRODUCTION

Mobile computing has achieved a level that has never been seen before that the number of smartphone users reaches 6.1 billion in 2020 [1]. Android is currently the most used mobile device platform over the world, occupying 85% of the market [2]. Users are moving away from data processing platforms like desktop computers and more towards mobile platforms. Smartphones are frequently used for messaging, e-commerce, shopping, online banking, and other services. These applications typically contain users' private and sensitive information. Malware is software that is designed to infiltrate and harm a Smartphone and retrieve the data from these applications without the owner's knowledge. Malware is a catch-all term for all types of computer threats. Malware is divided into two types: file infectors and stand-alone malware. Malware can also be classified according to its specific action, such as worms, backdoors, trojans, rootkits, spyware, and adware. Fake apps may trick users into installing malware software that is harmful to the mobile platform. Malware applications disguised as benign software that are installed on users' smartphones can access various areas of the device and capture significant data in transit, resulting in serious data leakage. As a result, from a security standpoint, it is critical for Android users to be cautious with their devices and raise their awareness in order to protect sensitive information in a recent study, the researchers exposed that Android devices are the most exposed to critical attacks. What's more, another study shows that more than 90% of Android malicious attacks exploit the known vulnerabilities [3]. To address existing security issues, researchers have developed Android malware detection tools using a variety of approaches. However, current methods based on signatures or behaviors cannot meet the malware detection requirements. Because of the nature of the learning algorithm, malware detection methods based on machine learning techniques are proposed.

In this research, malware detection was accomplished through classification utilizing many machine learning algorithms, including Decision Tree, Random Forest, Gradient Boosting, Ada Boost, Gaussian Naive Bayes, and Support Vector Machine. we will compare the performance of several classifiers on our datasets.

2. BACKGROUND

In this section, we will discuss more in detail about Malware and Features.

2.1 Malware

Malware, an abbreviation for "malicious software," refers to any intrusive program created by cybercriminals (sometimes known as "hackers") to steal data and damage or destroy computers and computer systems. Malware examples include viruses, worms, Trojan viruses, spyware, adware, and ransomware. Recent malware attacks have resulted in massive data leaks.

2.2 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

2.2.1 Classification

Classification is a supervised machine learning process in which the model attempts to predict the proper label of

incoming data. In classification, the model is fully trained using the training data before being tested on test data and used to predict fresh unseen data.

2.2.2 Features

A feature is an individual quantifiable property or characteristic of a phenomenon in machine learning. Choosing informative, discriminating, and independent features is a critical component of classification algorithms. we used three different features as part of this project. which are given below.

2.3 Android Component

An Android application is made up of several key components. The application manifest file links these loosely linked components by providing a description of each component and how they interact. The manifest file also contains information about the program, device configuration, platform requirements, external libraries, and required permissions. The following are the essential components of an Android app: Permissions, Activity, Receiver, and Content providers. App components are the fundamental building pieces of an Android application. Each component serves as a point of entry into your software for the system or the user. We are using some of these android components as features in our project to train machine learning models.

2.3.1 Permissions

To protect Android users' personal information and data privacy, applications require permission to access phone resources such as GPS position, Bluetooth, contact list, and microphone. This permissions-based technique also aids in accessing resources in isolated programs and transferring data between the Android system and other applications this information is present in manifest.xml file.

PERMISSIONS
INTERNET
ACCESS_NETWORK_STATE
WRITE_EXTERNAL_STORAGE
VIBRATE
READ_PHONE_STATE
ACCESS_FINE_LOCATION
WAKE_LOCK
READ_CONTACTS
WRITE_CONTACTS

Table 1: Sample Permissions

2.3.2 Intent

You may use an intent, which is a message object, to request an action from another app component. Intents have three fundamental use cases: beginning an action, establishing a service, and delivering a broadcast. Intents can help to streamline communication between components in a variety of ways. There are two types of intentions: To declare which application will carry out an express intention, either the package name of the targeted application or a fully qualified component class name must be specified. You usually use an explicit intent to start a component in your own app since you are familiar with the class name of the activity or service you want to begin.

Intents
SET_TIMER
SET_ALARM
SHOW_ALARMS
IMAGE_CAPTURE

Table 2: Sample Intents

3 Approach/Methodology

To complete our project, we developed a suite of Python scripts to automate this analysis. The “core” of the software is a permission analyzer sub-package that accepts an APK file path , analyzes the app’s permissions, and saves the data to a xml file for later analysis and comparison. For the malicious apps from the Malware Genome, we had to manually categorize them since Android manifest files do not contain application categories since they may vary by marketplace. We developed Python scripts that iterate over the APK files from the dataset directories), use the APKTOOL to unpackage the apk files and run our permission analyzer script to update the data file with the permissions.

3.1 Dataset

We used around 900 api’s out of which 500 are benign and 400 are malicious we randomly divided 20% of data into testing data set and 80% data into training data set we train the machine learning models on training data set and test the trained model using testing data set.

4 RESULT AND ANALYSIS

4.1 Malware Detection Models

4.1.1 Logistic Regression

Logistic Regression works as a parametric binary classification approach. It is used to determine if events will succeed or fail. It can only take binary values. It can anticipate the result with the use of observations from the training data set. It does not need a large amount of computing power. Much easier to make and far less difficult. Training may be given effectively. It can classify

unknown records fast. The accuracy of this technique is extremely high just for short datasets. It is incapable of dealing with larger dataset characteristics. It is possible to construct linear boundaries. It cannot handle nonlinear scenarios due to its linear decision surface and It is unable of predicting right results.

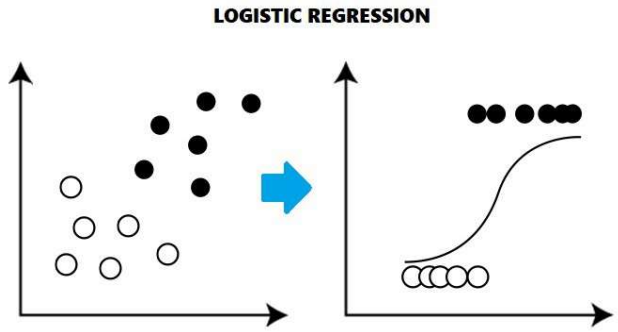

Fig 1: Logistic Regression

Recall	Precision	F1	TP	FN	FP	TN
0.934	0.934	0.934	99	5	5	71

Table 3: Output metrics for Logistic Regression model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.621	0.921	0.742	41	50	7	82

Table 4: Output metrics for Logistic Regression model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.888	0.975	0.969	96	3	2	79

Table 5: Output metrics for Logistic Regression model on feature-data-set-3.

4.1.2 Random Forest Classifier

The Random Forest Classifier may be applied to both regression and classification problems. It employs the Ensemble learning idea. To tackle the difficult problem, it integrates several classifiers. This technique improves the model's performance. Over-fitting can be avoided. When there is a vast amount of data, it is quite accurate. It is effective with both categorical and continuous data. Missing values can be handled automatically. It is optional to scale the features. Normalization is not necessary. The shortcomings of the technique are the model consumes more time during training and algorithm takes more RAM to execute. A slight change in the data might affect the technique .

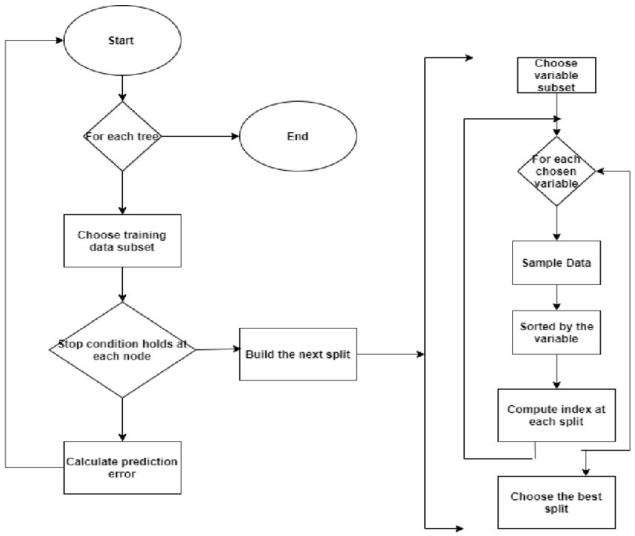

Fig 2: *Random Forest Classifier*

Recall	Precision	F1	TP	FN	FP	TN
0.922	0.934	0.928	98	6	5	71

Table 6: Output metrics for Random Forest Classifier model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.621	0.921	0.742	41	50	7	82

Table 7: Output metrics for Random Forest Classifier model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.963	0.988	0.976	96	3	1	80

Table 8: Output metrics for Random Forest Classifier model on feature-data-set-3

4.1.3 Decision Tree Classifier

As the name implies, a decision tree is a type of tree structure that operates according to the principle of conditions. It contains effective algorithms that are utilized for predictive analysis. Its primary characteristics include internal nodes, branches, and a

terminal node. Every leaf node denotes the class label, whereas every internal node represents a "test" on an attribute. Branches carry the results of the tests. When it comes to supervised learning approaches, this algorithm is the most used. It is employed in both classification and regression. It is frequently referred to as "CART," which stands for Classification and Regression Tree. Due to their stability and dependability, tree algorithms are always recommended.

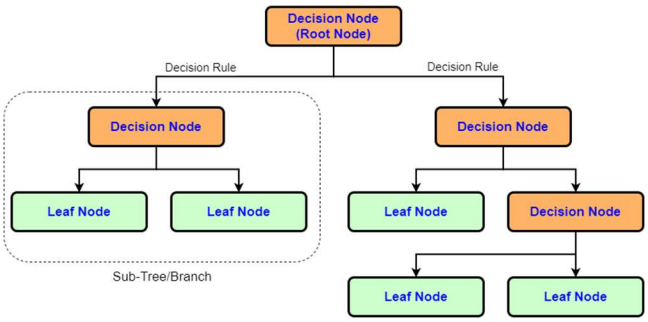

Fig 3: *Decision Tree Classifier*

Recall	Precision	F1	TP	FN	FP	TN
0.891	0.868	0.88	96	8	10	66

Table 9: Output metrics for Decision Tree Classifier model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.621	0.921	0.742	41	50	7	82

Table 10: Output metrics for Decision Tree Classifier model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.888	0.975	0.929	89	10	2	79

Table 11: Output metrics for Decision Tree Classifier model on feature-data-set-3.

4.1.4 Support Vector Machines(SVM)

Support vector machines (SVMs) are a group of supervised learning techniques for classifying data, performing regression analysis, and identifying outliers. The advantages of support vector machines is that it is efficient in high-dimensional environments. Still useful in situations where the number of dimensions exceeds the number of samples. It is also memory efficient since it only uses a portion of the training points (known as support vectors) in the decision function. Different Kernel functions may be given for the decision function, making it versatile. There are common kernels available, but you may also define your own kernels. Support vector machines have a variety of drawbacks, including the need to avoid over-fitting when selecting Kernel functions and regularization terms if the number of features exceeds the number of samples.

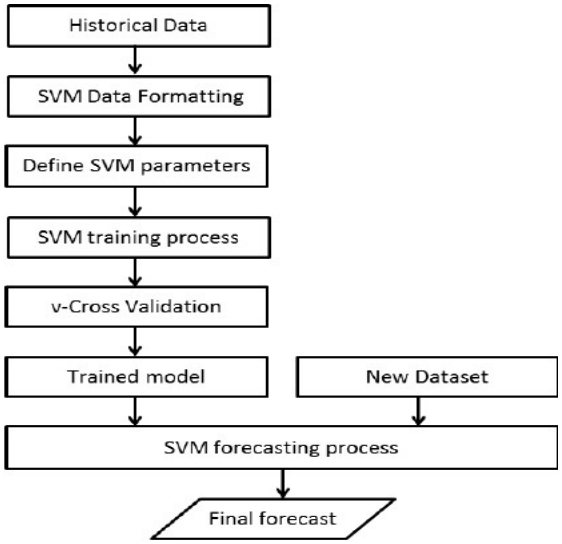

Fig 4: Support Vector Machines *Classifier*

Recall	Precision	F1	TP	FN	FP	TN
0.921	0.946	0.933	100	4	6	70

Table 12: Output metrics for Support Vector Machine Classifier model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.621	0.921	0.742	41	50	7	82

Table 13: Output metrics for Support Vector Machine Classifier model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.988	0.975	0.981	98	1	2	79

Table 14: Output metrics for Support Vector Machine Classifier model on feature-data-set-3.

4.1.5 ADA Boost Algorithm

AdaBoost, commonly referred to as Adaptive Boosting, is a machine learning technique used as an ensemble method. Decision trees with one level, or decision trees with only one split, are the most common AdaBoost technique. Decision Stumps are a common name for these trees. Building a model using the training dataset comes first in boosting approaches, followed by building a second model to fix the errors in the first model. This method is applied repeatedly until the errors are minimized and the dataset can be correctly forecasted.

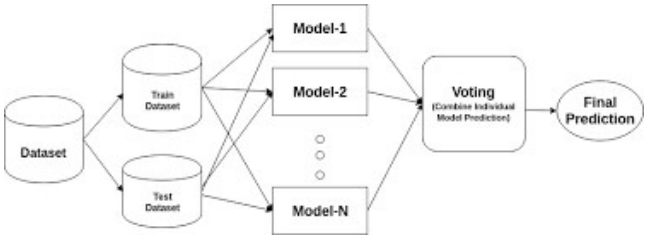

Fig 5: *ADA Boost Classifier*

Recall	Precision	F1	TP	FN	FP	TN
0.932	0.895	0.913	99	5	8	68

Table 15: Output metrics for Ada Boost Classifier model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.617	0.921	0.739	40	51	7	82

Table 16: Output metrics for Ada Boost Classifier model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.952	0.975	0.963	95	4	2	79

Table 17: Output metrics for Ada Boost Classifier model on feature-data-set-3.

4.2 **Comparison of Feature Extractions**

We employed three separate features to detect malware in this Malware detection. The features are accessed through Permissions, Main Activity, and Receive Intent. The permissions are simply utilized by the APK at the start of it, and we can use Permissions to determine if the APK is malicious or benign by monitoring anomalous permissions such as transmitting SEND-SMS permission. This sort of conduct is an APK sensitive security behavior. The rights requested by malware and benign APKs differ; the permissions requested by malware are distinct. As a result, the information is fed into machine learning models as training data by extracting all permissions characteristics from both the malware and benign APK data sets. This will help the machine learning model to get trained on the data and help in identifying the malicious application. The second feature we are using is User-Feature Hardware we have to declare the details of the Hardware we are using in manifest.xml file and the feature combinations requested by the malicious applications are very distinct from the features requested by benign applications so they can also be used as a feature set for machine learning model to distinguish between malicious and benign applications. coming to the last feature we used in our project it is Receive Intent, It is nothing more than an APK broadcast function that aids in the transmission and receiving of data from other APKs, Websites, Third-party apps, and so on. When an APK sends or receives data without the owner's knowledge, it is believed to be malware; hence, receive Intent is utilized as a characteristic to distinguish between Malware and begin. Among all the features, the permission feature is the most efficient since it includes all of the permissions (404) requested by Malware and Benign. In comparison to the Main Activity and Receive Intent features, which mostly include information in duplicate form, i.e., most of the APK gives the same information, which does not allow the ML model to learn new things about the APK and may even lower the F1 score. The selection of features play an important role in our project as the benign application may rationalize the set of features used by malicious applications so we must be very careful when selecting the features to train machine learning model. In our project the max f1-score was recorded when we used the combination of the multiple features i.e,feature-set-3.

Recall	Precision	F1	TP	FN	FP	TN
0.934	0.934	0.934	99	5	5	71

Table 3: Output metrics for Logistic Regression model on feature-data-set-1.

Recall	Precision	F1	TP	FN	FP	TN
0.621	0.921	0.742	41	50	7	82

Table 4: Output metrics for Logistic Regression model on feature-data-set-2.

Recall	Precision	F1	TP	FN	FP	TN
0.888	0.975	0.969	96	3	2	79

Table 5: Output metrics for Logistic Regression model on feature-data-set-3.

All the above data is obtained from LOGISTIC REGRESSION for all the 3 feature sets.

4.3 Comparison of Malware Detection ML Models

The Random Forest Classifier is an accuracy-focused approach that performs best when paired with proper fit; otherwise, it quickly gets overfat. Individual RFC decision trees with random selection can capture more complex feature patterns with the highest accuracy. RFC can also inform us how much each feature contributes to class prediction using Feature Importance and tree diagrams for enhanced comprehension. Logistic regression is used when your Y variable can take only two values, and if the data is linearly separable, it is more efficient to classify it into two separate classes. The true and false positive rate of random forest improves as the number of explanatory elements in a dataset grows. Because we are delivering vast amounts of data, random Forest produces more efficient data than others. A random forest is just a series of decision trees with their results aggregated into a single final result. They are especially beneficial since they can prevent overfitting while increasing inaccuracy due to bias. Random forests, on the other hand, are a robust modeling technique that outperforms a single decision tree. They employ a large number of decision trees to minimize overfitting and bias-related inaccuracy, resulting in usable results. The random forest can generalize better across data sets. Random forest beats decision trees because of the randomized feature selection. and Adaboost. Both techniques are applicable to classification and regression problems. The Random Forest and AdaBoost algorithms are both based on the formation of a forest of trees Adaboost, on the other hand, is more susceptible to overfitting than Random Forest.

Recall	Precision	F1	TP	FN	FP	TN
0.888	0.975	0.969	96	3	2	79

Table 5: Output metrics for Logistic Regression model on feature-data-set-3.

Recall	Precision	F1	TP	FN	FP	TN
0.963	0.988	0.976	96	3	1	80

Table 8: Output metrics for Random Forest Classifier model on feature-data-set-3

Recall	Precision	F1	TP	FN	FP	TN
0.888	0.975	0.929	89	10	2	79

Table 11: Output metrics for Decision Tree Classifier model on feature-data-set-3.

Recall	Precision	F1	TP	FN	FP	TN
0.988	0.975	0.981	98	1	2	79

Table 14: Output metrics for Support Vector Machine Classifier model on feature-data-set-3.

Recall	Precision	F1	TP	FN	FP	TN
0.952	0.975	0.963	95	4	2	79

Table 17: Output metrics for Ada Boost Classifier model on feature-data-set-3.

4.4 ADDITIONAL DATA-SET RESULTS

We took an additional data set of 913 api out of which 493 are benign and 420 are malicious and we extracted the features from those api's same as in the original data set and ran the trained machine learning models on the new data set and we got an F1-Score of 0.68 the low score is because we dropped around 430 features from the new data set to match the feature set of the original data set but when we trained the models with addition of features from the new data set and doing the same procedure as we did for original data set we got an F1-Score of 0.94 which is same as the original from this we can conclude that with addition of viable features to the machine learning model the prediction accuracy of the model increases.

Recall	Precision	F1	TP	FN	FP	TN
0.556	0.897	0.687	460	27	189	237

Table 18: Output metrics for the Random Forest Classifier model trained on original data set and tested on the additional data set.

Recall	Precision	F1	TP	FN	FP	TN
0.960	0.979	0.969	96	4	2	80

Table 19: Output metrics for the Random Forest Classifier model trained on 80% of additional data set and tested on the 20% of the additional data set.

5. RELATED WORK

• Yijin Geng, Junbo Wang, Shang Gao, and Wei Shi proposed Permission Sensitivity-Based Malicious Application Detection for Android. They successfully researched about models using permissions-based features.

• Department of Computer Information Systems, Jordan University of Science and Technology, Irbid, Jordan proposed a Survey on malware detection techniques

6. CONCLUSION

In this project malware detection was accomplished by using machine learning models such as Random Forest, Logistic Regression, SVM, AdaBoostClassifier, Decision Tree Classifier and in conjunction we used different type of feature sets. we got best F1-Score of 0.983 with Random Forest Classifier when we trained it using a feature set containing combination of permissions and intents(feature-set-3) and we got least F1-Score when we used feature set 2 which had Hardware User-Feature as the features this is because of very low features that are extracted which were only around 60 for the entirety of the data set. From this we can conclude that having more training data can facilitate better training of model and this in turn will increase the accuracy of prediction.

7. WORKLOAD

Team Members	Work Done
Teja Nagireddy	Code Development, Data Extraction, Data Collection
Lushaank Kancherla	Pair Programming, Writing of Report
Briana Rajan	Pair Programming, Writing of Report

REFERENCES

[1] J. Lopes, C. Serrao, L. Nunes, A. Almeida, J. Oliveria. Overview of machine learning methods for Android malware identification. European Union, 2019.
[2] J. Li, L. Sun, Q. Yan , Z. Li, W.Srisa-an , and H. Ye, Significant Permission Identification for Machine-Learning-Based Android Malware Detection. Industrial Informatics, vol. 14, No. 7, July 2018.
[3] M. Al-Janabi and A. M. Altamimi, A Comparative Analysis of Machine Learning Techniques for Classification and Detection of Malware. International Arab Conference on Informative Technology, 2020.