Analysis of Android Malware Permission Based Dataset Using Machine Learning

ISSM581: Research Project

Spring 2021

Sushmitha Suruliyandi Ramani (ssuruliy@student.concordia.ab.ca)
Manikanta Naralasetty (mnaralas@student.concordia.ab.ca)
Avneesh (alnu28@student.concordia.ab.ca)
Ambica Kothakapu (akothaka@student.concordia.ab.ca)
Shainija Rudraraju (srudrara@student.concordia.ab.ca)
Sai Harish Nasina (snasina@student.concordia.ab.ca)
Sirisha Metla (smetla@student.concordia.ab.ca)

Research Project

Submitted to Faculty of Graduate Studies
Concordia University Of Edmonton

In Partial Fulfilment of the
Requirements of ISSM-581 Course

Concordia University Of Edmonton
FACULTY OF GRADUATE STUDIES
Edmonton , Alberta

Advisor: Dr. Sergey Butakov (sergey.butakov@concordia.ab.ca)
Department of Information Systems Security Management
Concordia University of Edmonton,
Edmonton T5B 4E4, Alberta, Canada

Analysis of Android Malware Permission Based Dataset Using Machine Learning

Sushmitha Suruliyandi Ramani, Avneesh,

Manikanta Naralasetty, Ambica Kothakapu,

Shainija Rudraraju, Sai Harish Nasina,Sirisha Metla

Approved:

Sergey Butakov [Original Approval on File]

Sergey Butakov Date: June 23, 2021

Primary Supervisor

Patrick Kamau [Original Approval on File]

Patrick Kamau, PhD, MCIC, PChem. Date: June 23, 2021

Dean, Faculty of Graduate Studies

Table of Contents

List of Tables

List of Figures

Analysis of Android Malware Permission Based Dataset Using Machine Learning

Sushmitha Suruliyandi Ramani (ssuruliy@student.concordia.ab.ca)
Manikanta Naralasetty (mnaralas@student.concordia.ab.ca)
Avneesh (alnu28@student.concordia.ab.ca)
Ambica Kothakapu (akothaka@student.concordia.ab.ca)
Shainija Rudraraju (srudrara@student.concordia.ab.ca)
Sai Harish Nasina (snasina@student.concordia.ab.ca)
Sirisha Metla (smetla@student.concordia.ab.ca)
Sergey Butakov (sergey.butakov@concordia.ab.ca)

***Abstract*—Security threats in Android applications have grown in sync with Android's growth. Machine leaning can be used to add functionality to traditional antivirus systems. Such an approach necessitates the identification and labelling of a large amount of harmful and benign code in advance to use it for model training. The modeling techniques and its practical implementation using the Android Malware Permission based dataset to find whether it is benign, or malware have been proposed in this paper. Random Forest Classifier, Logistic Regression, Decision Tree Classifier, XGB Classifier are the machine learning algorithms implemented in this research. The overall performance of approximately 85% of accuracy has been achieved on the dataset. The dataset is freely available to the research community.**

Keywords— Machine learning, Android Malware, Permission, Dataset, Malware.

I. Introduction

This research looks at the problem of classifying Android applications whether it is benign or malware depending on the characteristics that can be extracted from the application package prior to its execution. The suspicious behavior of an application typically requires permissions that could be an indicator of suspicious activity being carried out by that application. This project superimposes machine learning algorithms on a raw dataset of Android permissions which have been classified as malware or benign to learn the characteristics of malware.

Android's file format for distributing and installing applications is APK, which stands for Android Package Kit (also known as Android Application Package). It contains all of the components that an app needs to properly run on your smartphone [1]. This research will help in uncovering the various patterns that are followed by an infected apk while accessing permissions. This will allow us to better prepare for such malware and help in early detection using machine learning algorithms.

In order to tackle the ever-changing security issues caused by malware there is a need of an up-to-date Android malware dataset. This dataset should be able to provide the various characteristic features of malware in Android APKs. Such characteristic features are identified based on the permissions that have been sought by the apk being analyzed. The dataset used for this research maps the permissions for each apk as used and not used. A binary mapping of this sort allows the machine learning models to learn the characteristics of the malware and better predict the nature of unknown APKs.

Various models exist in machine learning to apply to Android malware datasets to uncover the underlying features. However, there is a need for building novel models given the nature of the dataset. This will help train the model better on the existing dataset and deliver good test results when tested against relatively new set of APKs. Models that allow for binary classification of data will be used for the purpose of this research. Some popular types of Android malware are Trojans, Keyloggers, Spyware, Ransomware.

To protect against the breakout of Android malwares it is crucial to be able to defend the devices in real time. This is achieved through machine learning by checking for the permissions requested by different APKs. The dataset being used contain numerous such APKs identified as malware or benign and the permissions that they sought after. This will allow for models to be trained to detect malicious activity based on the permissions being requested by the application.

II. Literature Review

A. *Malware Detection Techniques*

Several malware detection techniques are used in real world to detect the malwares. The most popular detection techniques are signature-based detection and behavior-based detection.

1) Signature Based Detection
A fingerprint of the file is known as Signature which can be uniquely identify. In case of malware, these fingerprints are extracted from the common patterns followed by the malware to identify it. Most of the antivirus software use this detection technique. Signature based detection however performs poorly in case of unknown malware [2].

2) Behavior Based Detection
Behavior based detection techniques detect malware based on the behavior of the file with the environment. Behavior based techniques cover the short coming of signature-based

detection since they can detect malicious content based on the behavior. This is particularly helpful when the malware mutates and changes its code to avoid detection since it requests for the same permissions and system resources and is thus possible to detect [2].

3) Heuristic Based Detection

Heuristic based approaches make use of techniques from signature-based detection and behavior-based detection. IT makes use of various hybrid features such as opcodes, APIs, and DLLs. These techniques have enabled the detection of zero-day malwares which are very hard to detect. However, in the case of complex malwares this process proves to be a weak detection technique [2].

4) Deep Learning Based Detection

Deep learning is a subsection of Artificial Intelligence and can work on unlabeled data for detecting patterns among them. From these patterns the program can learn and can then detect malicious behavior from files. Deep learning also reduces the number of features required for detecting malware [2].

5) IOT Based Detection

Internet of Things devices are becoming more popular nowadays and include devices such as smart home appliances, mobile devices etc. Due to their growing popularity, they have become the target of attackers who previously focused on computers. The malware on Android devices is detecting by looking at the set of permissions being requested and classifying them as benign or malware [2].

6) Cloud Based Detection

Cloud technology is rapid growing these days. It has found its application in malware detection as well through security as a service. Users can upload any file to the cloud and detect if the file is malicious or not. This is possible due to the capacity to store large datasets on the cloud and thus enhance the detection techniques of mobile and personal computers by providing security as a service [2].

7) Machine Learning Algorithms

Most detection techniques make use of machine learning techniques to detect anomalous behavior among files. In the case of an apk file, the features that can be used to classify it as benign or malware are permissions, system calls etc. Once the features have been extracted, machine learning algorithms to it such as random forest, linear regression etc. can be applied [2].

B. *Machine Learning* Approaches to detect Malware

Malicious traffic can be detected through numerous ways which is divided into supervised learning, unsupervised learning, and reinforcement learning.

1) Supervised Learning

Supervised machine learning is one of the sub-categories of artificial intelligence and machine learning. In this technique labelled datasets are used to train and test the performance of the models and to identify the data accurately. During cross validation when the input is given to the model, the weights are adjusted until it is fitted properly [3]. Supervised machine learning is further divided into two categories for data mining: Classification and Regression.

i) Classification:

To accurately assign test data into certain categories, classification uses an algorithm. It identifies certain entries in the datasets which then tries to conclude that, this is how the attributes are labelled or identified. Linear Classifier, Random Forest Classifier, Decision Tree Classifier and XGBooster Classifier are the most used classification algorithms.

ii) Regression

Regression is a technique to find the relationship between the dependent and independent variables. It is commonly used to produce estimates for an organization’s sales revenue. Linear Regression, Logistic Regression, and Polynomial Regression are some of the popular regression algorithms [3].

2) Unsupervised Learning

Unsupervised machine learning uses the ML techniques to process based on unlabeled datasets and helps to analyses and form clusters depending on the data. This algorithm can provide data groupings and patterns automatically without human interventions. It can find the similarities and differences in the data. It is a good option for exploratory data analysis and cross-selling techniques [4]. Clustering, association, and dimensionality reduction are the three basic tasks that unsupervised learning models use.

3) Reinforcement Learning

When machine learning models are trained to make a series of decisions is called as reinforcement learning. The agent acquires to achieve its goal in an uncertain and potentially complex environment. An artificial intelligence undergoes a game-like occurrence in reinforcement learning. The computer uses the trial-and-error method to identify a solution for a problem [4].

C. *Existing Datasets*

Android malware datasets, which used one of the above detection techniques and machine learning approaches to detect malware.

CICMalDroid 2020:

- The total number of android samples are 17341.
- This dataset consists of recent and complex android malware samples up to 2018.
- It contains diverse samples from five different types such as SMS malware, banking malware, Adware, riskware, and benign malware which are examples of malicious software.
- When compared to other public databases, it has the most comprehensive set of static and dynamic functionality.

Data Collection:

The android samples were gathered through various sources like the Contagio security blog, AMD, VirusTotal, MalDozer, and other datasets which have all been included in recent research contributions. All the samples are gathered from December 2017 to 2018. To develop effective countermeasures and mitigation strategy, the cybersecurity researchers need to identify the malwares in Android apps.

The five categories of data in the dataset are SMS malware, banking malware, adware, riskware and benign malware.

Data Analysis:

CopperDroid was used as a VMI-based dynamic analysis framework, recreation of low level OS specific and high level android specific behaviours of samples are analyzed dynamically to collected data. The successful test runs are for 13717 samples while rest of them are failed due to the errors including timed-out, invalid APK files, and allocation of memory failures.

Preprocessing of CICMalDroid 2020 dataset:

In the pre-processing phase, three categories of dynamically observed behaviour are extracted from the captured log files and the modules are system calls, binder calls, and composite behaviours. This helped them to automatic reconstruction of system call semantics such as IPC, RPC, and complex android objects. The examples of composite behaviour that grouped together widely used low level system calls are FS_access (create, write), network_access (read, write), and fs_pipe access (read, write). Specifically, get_DisplayInfo, register_Callback, and composite actions fs access(write) are binder calls which can be obtained in JSON format from CopperDroid analysis.

With an F1- Score of 97.84 percent and a false positive rate of 2.76 percent, the model can detect and categorize malware. [5].

The Drebin Dataset

This dataset consists of 131,611 applications from various malware families, in which is samples are gathered from August 2010 to October 2012 [6]. Samples are collected through the applications from GooglePlay, various alternative app markets located in China and Russia, and other Android websites, malware forums and security blogs and the malwares from Android Genome Project. They used Anti-virus scanners to detect the malicious applications which helped them to ensure that the data are split accurately between benign and malware [7].

The dataset has been spilt into eight categories such as [8],

- Hardware components are used to set required permission by the software.
- Requested permissions, grant access to users to install or use their appropriate resources.
- App components has activities, services, content providers and broadcast receivers.
- Filtered intents which communicates between various components and applications.
- Restricted API, Used permissions and Suspicious API calls for sensitive data or resources.
- Network address, IP address, hostname, and the URL.

Family name	Top 5 features
FakeInstaller	sendSMS SEND_SMS android.hardware.telephony sendTextMessage
DroidKungFu	SIG_STR system/bin/su BATTERY_CHANGED_ACTION READ_PHONE_STATE getSubscriberId
GoldDream	sendSMS lebar.gicp.net DELETE_PACKAGES android.provider.Telephony.SMS _RECEIVED getSubscriberId
GingerMaster	USER_PRESENT getSubscriberId READ PHONE STATE system/bin/su HttpPost

Table 1: Top 5 Malware families and its features

Obfuscation Technology

To render it more complicate for reverse engineers to comprehend the apps or for detection technologies to identify them, malicious application developers would obscure static features to some level. The primary ways are Identifier Renaming, String Confusion, Call Indirection, Junk Code Insertion, Dynamic Code Loading [9].

Various static detection approaches have different counters for the obfuscation approaches mentioned above. Static detection often enhances detection accuracy by leveraging a multiple features.

Test Results

When this dataset is trained using the models of Logistic Regression, Decision Tree, Random Forest and Neural Network for the 42,570 samples are as below [8].

Model	Accuracy	Precision	Recall
Logistic Regression	99.45%	1	96.32%
Decision Tree	99.86%	1	95.91%
Random Forest	99.92%	1	95.85%
Neural Network	99.83%	1	99.95%

Table 2: Test results of Drebin dataset

The Android malware dataset is different from existing dataset since it makes use of obfuscation to better capture the permissions used by the apk files being analyzed. Obfuscation shortens the names of the application's classes, methods, and fields. Obfuscation is a security through obscurity technique which alters the code to prevent automatic or manual code analysis.

The dataset contains obfuscated samples which are used by malware authors to prevent detection without changing original malware. The models being trained on this dataset have a better scope of detecting the malware in the real world where the code is often obfuscated to avoid detection.

III. Methodology

1) Data engineering:
The APKs were downloaded from the CIC Dataset provided by University of New Brunswick. These APKs were classified using Virus Total into Benign and Malware. The permissions in these APKs were extracted and compared with a default permission list, if a match occurs those permissions were labelled as 1 otherwise 0. The columns were filtered based on the unique values they contain and the total number of columns are 536. Principal Component Analysis was performed on the dataset to reduce the number of columns. The columns were reduced to 300 after performing PCA analysis.

2) Model Building:
Scikit learn library was used to build the Logistic Regression model. The dataset was split into 20% test data and 80% training data.

3) Model verification:
The accuracy of the model was calculated using the accuracy score function.

IV. Android Malware PERMISSION BASED Dataset

Android Malware permission Based Dataset is a newly generated dataset which comprises of permission with benign and malware. The actual dataset consists of 10686 rows signifies the APKs and 536 columns represents the various Android permissions and apk files. These APKs were collected from the CIC Dataset by University of New Brunswick. The permission in these APKs is compared with the default list of permissions by using python code and if match occurs then it is labelled as malware else benign, which is used for the purpose of machine learning.

V. DATA COLLECTION

The APK samples was gathered from AndroZoo, which is a growing repository of Android apps extracted from a variety of sources, including the official Google Play app store. It currently contains 15,510,743 APKs, each of which has been (or will soon be) analyzed by dozens of Antivirus products to determine which applications are classified as Malware [10]. This was started in 2016 by researchers from the University of Luxembourg and is still growing. For this experiment, 10,686 APKs from AndroZoo were selected from 2013 to 2016 by filtering them by virus total detection with 0 to 30+ to obtain higher certainty of malware samples, allowing for efficient and broad Android application markets to be included.

VI. ML DATASET GENERATION

A. Feature extraction

Android applications are packaged as APK archives. The manifest, AndroidManifest.xml, is a critical component of this file. Android manifest.xml contains several elements, including the package name, permissions. App permissions protect user privacy by limiting access to Restricted data (such as device state and a user's contact information) and Restricted actions.

B. Data set generation

Android malware has emerged as the most serious threat to the widely used Android ecosystem. Several machine learning-based techniques for detecting Android malware are continually being developed.

Permissions are extracted from Malware and Benign applications in their respective folders using jadx, a Dex to Java decompiler through which each APK is unpacked and permissions are extracted using AndroidManifest.xml by setting the status to permission list which exists in Perm List and it constantly updating the list, and then combined into a single Comma Separated Values (.csv) for use in machine learning techniques [11].

The first column provides the "NAME" of the relevant APK, while the last column "CLASS" indicates if the application corresponds to the benign or malware training set. [Benign=0, Malware=1], in between Name and Class all are of permissions [0=The Android application does not use this permission, 1=The Android application uses this permission].

File Name	P 1	P 2	…	P N	CLASS
Malware APK 1	1	1	…	0	1
Malware APK 2	0	0	…	1	1
Benign APK 1	1	1	…	1	0
Benign APK 2	1	1	…	0	0

Table 3: Data Set Format

Table (3) shows how data is formatted, and data set is produced for 10685 APK samples i.e., 3000 samples of Benign and 7686 samples are malware, and 536 unique permissions are extracted [12].

Data set produced from the above extraction which is a sparce data set means that a large percentage of the values are zeros on each row for an Android Application Packages with the permissions used and class either malware or benign. and with 10686 APK samples on average 0.017 permissions are used of 536 permissions Table (4) shows the detailed count of the data set.

COUNT	SUM	AVERAGE	0's	1's
5727696	101806	0.017774337	5625890	101806

Table 4: Data Set Count

Table (5) shows permissions that are repeated and merged as one column in order to reduce the number of columns with unique permissions, in this case for permission.C2D_MESSAGE is given as [application's package + ".permission.C2D_MESSAGE"] in APK AndroidManifest.xml over 750 out of 10685 similarly for the other permissions mentioned below with their count on the side.

PERMISSION	Count	PERMISSION	Count
C2D_MESSAGE	750	WRITE_SETTINGS	21
JPUSH_MESSAG	41	RECEIVE	12
MAPS_RECEIVE	98	INSTALL_SHORTCUT	8
READ_SETTINGS	33	ACCESS_DOWNLOAD_MANAGER	6
WRITE_SETTINGS	20	ACCESS_DOWNLOAD_MANAGER_ ADVANCED	6

Table 5: Merged Permissions

VII. DATASET PREPROCESSING

1) *Removing unimportant columns:*

In Android Malware Permission Based Dataset, the column "NAME" is not an important feature so this is removed by using the drop command. This feature does not provide much importance in detecting the malware in building machine learning models and some of features are also removed using feature selection techniques.

2) *Splitting the dataset to train and test:*

Splitting of train and test dataset is to evaluate how good the machine learning models are performing. Training the data is to build a model and testing the data is to evaluate the models. This evaluation is done in our dataset.

3) *Feature Scaling:*

Feature scaling is a technique for normalizing the independent variables range or data components. It is also known as data normalization. On our dataset some of the techniques are used as follows,

Two feature scaling techniques below are used to analyze the attributes in the dataset which helped to identify top important features for developing the model.

A. *Principal Component Analysis*

PCA is an unsupervised learning technique that reduces the dimensionality of the data which is used frequently in machine learning. If the dataset has larger dimensionalities, it will be expensive when training the model, also this will reduce the accuracy or efficiency of the model. So, it is important to reduce the dimensions of the data and then train the system. But there should not be any loss of information while reducing the dimensions of the data. PCA helps to identify the important features of the data based on the variance. The feature which has the highest variance is the first principal component and arranges them in the order of feature's variance [13].

The graph is plotted with the xlabel as number of components to the ylabel as cumulative explained variance. It shows that there are approximately 300 out of 567 features contains important data which is used to train, test and create a model and the remaining features can be less important or null values. This further helps to improve the performance of the model. Models are created for 350 features to compare the performance based on the attributes given by PCA analysis.

```
Text(0, 0.5, 'Cumulative explained variance')
```

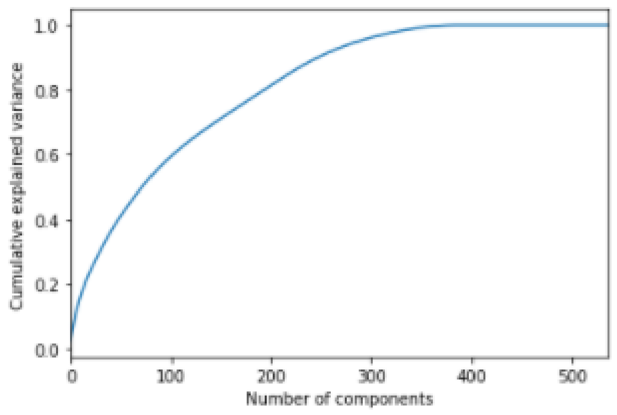

Figure 1: PCA-Cumulative variance

Then using PCA gives the top 10 components depends on its variance.

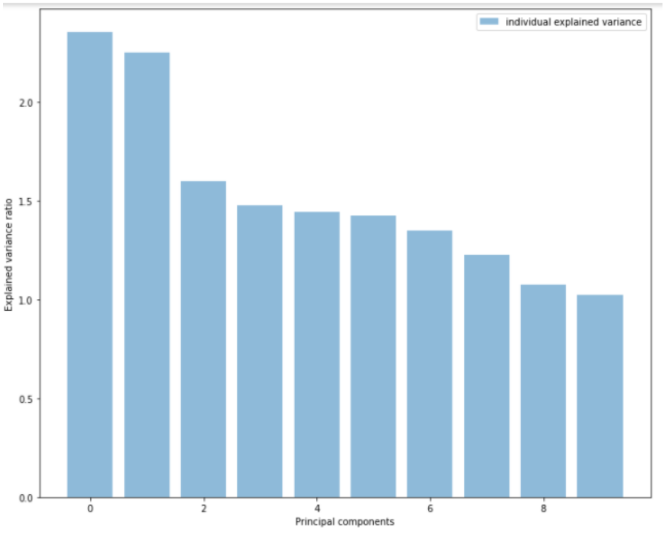

Figure 2: PCA-Variance

The above PCA variance graph shows that the first principal component (PCA1) has the variance of 2.25 which then decreases gradually, these 10 components are considered in this dataset in which these information can be used to test and train the model and predict its accuracy and another data processing is done by using feature importance.

B. *Feature Importance*

The important factor that has an impact on the performance of the machine learning malware detection system is the features present in the sample data [14]. Feature importance is a technique which assign a score to the input values or features and predict whether they are useful to target variable. This helps for dimensionality reduction and feature selection which helps in improving the accuracy and effectiveness of the model.

Here, Random Forest classifier is used to get the feature scores of the features using feature importance.

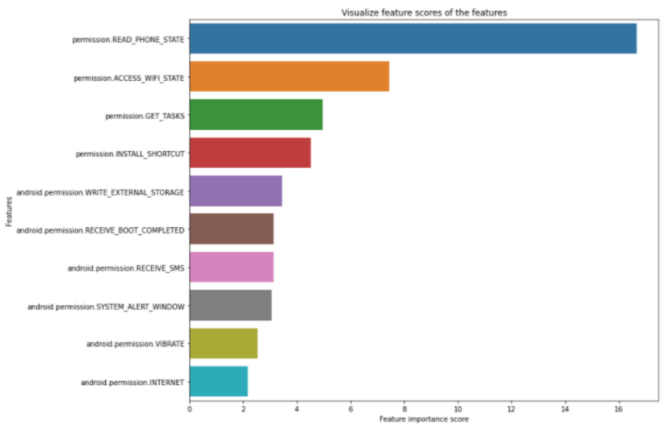

Figure 3: Visualization of feature score of the features.

The top 5 features of the Android malware dataset are,

Feature name	Feature score
permission.READ_PHONE_STATE	0.175198
permission.ACCESS_WIFI_STATE	0.078058
permission.INSTALL_SHORTCUT	0.041135
permission.GET_TASKS	0.040872
android.permission.SYSTEM_ALERT_WINDOW	0.036117

Table 6: Top 5 features of Android Malware Data Set

Feature importance method is used to get the top 300 features by using its variance or the feature score, which helps to identify the important features that helps to build an efficient model. DERBIN dataset has 113 Manifest permissions on which 108 are found similar from Android permission Based Data set's 526 Permissions i.e., 418 unique permissions are found up on compared with DERBIN dataset.

VIII. EXPERIMENTAL RESULTS

The efficiency of the ML-based models can be evaluated using metrics such as accuracy, precision, recall and f1-score. The overall effectiveness of the machine learning algorithms in detecting the malware based on the Android permissions is given by accuracy. To evaluate the performance of the classification or information retrieval precision and recall are used. It helps to define the false positive and false negative. Higher the F1-score the better the performance of the machine learning model.

Algorithms	ML Evaluation Metrics			
	Accuracy	Precision	Recall	F1-score
Logistic Regression	85.22%	0.81	0.81	0.81
Random Forest Classifier	85.73%	0.84	0.79	0.81
Decision Tree Classifier	83.72%	0.80	0.78	0.79
XGB Classifier	86.20%	0.85	0.80	0.82

Table 7: Result Analysis using PCA

By considering 300 features in the dataset, Logistic regression, Random Forest classifier, Decision Tree Classifier and XGBooster Classifier model have been trained and predicted its accuracy of the model.

By performing feature importance, the top 300 features are filtered out and built a machine learning models to detect malware. Accuracy of the model tells how effective the ML models in Table 7 in detecting the malware.

Algorithms	ML Evaluation Metrics			
	Accuracy	Precision	Recall	F1-score
Logistic Regression	84.94%	0.81	0.81	0.81
Random Forest Classifier	86.44%	0.84	0.79	0.81
Decision Tree Classifier	84.89%	0.80	0.78	0.79
XGB Classifier	86.00%	0.82	0.82	0.82

Table 8: Result Analysis using Feature Importance

From this result, the Random Forest Classifier Model has the highest accuracy of 86.44% in detecting malware and Decision Tree Classifier has the least accuracy of 84.89%. The other two models such as Logistic Regression and XGB Classifier also obtained better accuracy of 84.94% and 86.00% respectively.

A confusion matrix is a methodology for describing a classification algorithm's performance. When there are an uneven number of observations in each class or more than two classes in the dataset, classification accuracy alone might be deceptive. Calculating a confusion_matrix can help to understand whether the classification model is getting correctly and what kind of errors it is making.

		Predicted Label			
		0		1	
Actual Label	0	🟦 **74.46%**	🟥 **64.90%**	🟦 **25.53%**	🟥 **35.09%**
		🟩 **65.07%**	🟨 **66.06%**	🟩 **34.92%**	🟨 **33.93%**
	1	🟦 **10.51%**	🟥 **8.81%**	🟦 **89.48%**	🟥 **91.18%**
		🟩 **6.07%**	🟨 **5.55%**	🟩 **93.92%**	🟨 **94.44%**

🟦 Logistic Regression
🟥 Decision Tree Classifier
🟩 Random Forest Classifier
🟨 XGBooster Classifier

Table 9: Confusion Matrix of Android Malware Dataset

Confusion matrix of Android Malware Permission Based Dataset indicates that performance of Logistic Regression, Decision Tree Classifier, Random Forest Classifier and XGBooster Classifier algorithms are almost similar, which is around 80% of true positives. But when compared to all four algorithms, XGBooster Classifier has lesser number of false positives of 5.55% and Logistic Regression has lesser number of false negatives of 25.53%.

By using the feature scaling technique of Principal Component Analysis, attributes are scaled to various numbers of 10, 300 and 350. The machine learning algorithms such as Logistic Regression, Random Forest Classifier, Decision Tree Classifier and XBG Classifier are used, and its overall performance are compared to find the efficiency of the model when various levels of attributes are taken into consideration. This paper mainly focuses on the accuracy for 300 features but to differentiate the performance of the model through various attributes selection.

ML Algorithms	Accuracy for 10 features	Accuracy for 350 features
Logistic Regression	73.85%	85.31%
Random Forest Classifier	84.28%	85.50%
Decision Tree Classifier	81.34%	82.46%
XGBooster Classifier	83.21%	86.58%

Table 10: Accuracy Comparison

When compared the accuracy for 10 and 350 features, the performance of Logistic Regression model is more efficient for 350 features. Random Forest Classifier has higher accuracy of 84.28% for 10 attributes and XGBooster Classifier has higher accuracy of 86.58% for 350 attributes.

Even if 350 attributes are taken into consideration, the overall performance of the algorithms are similar by using the feature scaling techniques of Principal Component Analysis but there is a better difference in the performance of models selection of 10 features are considered.

IX. DISCUSSION

CICMalDroid vs Android-Malware-Permission-Based-Dataset

CICMalDroid 2020 dataset is extracted from 17,341 Android samples from various datasets, including the VirusTotal service, Contagio security blog, AMD, MalDozer, and others. upon comparing with our permission-based data set we have performed with 10685 APKs of 14 different markets and both are from the same source Androzoo of 15,510,743 APKS.

When it comes to feature extraction maldroid extracted Activities, Metadata, permissions, System features (camera and internet) which comes under static analysis Data set. Permission based data set is well organized compared to Maldroid data set within detail representation of each permission.

Considering the results of both datasets, CICMalDroid 2020 has high F1 score of 97.84 % compared to the Permission based Dataset which has 81%. Even though the F1 score is comparatively less than the CICMalDroid, the accuracy of 86.44% that has been obtained by performing Machine learning models is good enough to say that the Dataset which has been created is effective.

DERBIN vs Android-Malware-Permission-Based-Dataset

For over 123,453 applications from various markets and 5,560 recent malware samples from Chinese Markets, Russian Markets, and other sources, DREBIN performs a comprehensive static analysis by capturing as many features as possible from the application's manifest for feature extraction such as permissions, API calls, and network addresses. with the similar approach we have extracted permissions by labelling malware / benign though virus total detection rate.

READ_PHONE_STATE was noted in top 5 features among both the datasets, of DERBIN’s FakeInstaller DroidKungFu, GingerMaster families and Permission based dataset. As DERBIN using large number of APK samples their top feature varies from our Permission based dataset.

The accuracy obtained in DERBIN Dataset by performing Logistic regression, Decision Tree, random forest, Neural network is around 99.92%, whereas accuracy of the Permission Based Dataset is around 86.44%.

X. CONCLUSION

Once the Android Malware Permission Based dataset is created for different android permissions, it has been analyzed and cleaned. This is known as data preprocessing such as removing unwanted features, splitting the dataset and feature scaling to get final dataset. The dataset has 10686 rows and 538 columns. The rows represent the different apk.

Different types of machine learning techniques were used to detect malware from Android Malware Permission Based Dataset: Logistic regression, Random Forest Classifier, Decision tress classifier, XGB classifier.

Logistic Regression can be used to examine the association of the permissions with either benign or malware label. Random forest classifier was particularly useful in the case of missing values such that it can maintain high accuracy even with missing values. Decision tree and XGB classifier models work on structured data, they help to distinguish various permission patterns and their outcomes as benign or malware. All these models are compared with their accuracy and the effectiveness of the model is obtained.

The Android Malware Permission Based Dataset can be used to identify the malware based on the set of permissions that are requested from the system. The dataset has 537 columns which enlist the permissions requested by the apk files and their classification as Malware or Benign. For this paper we worked with 300 features to train the models. Extracting few more features with large number of APK samples helps in getting good results which benefits the developers working on ML techniques to detect malware. The models that have been trained on the dataset show promising results in detection of malware.

A possible future application of this research could be the use of these trained models in building an anti-virus system. The detection capabilities will allow for previously unknown malwares to be detected.

Dataset is available for download at
https://github.com/harrypro02/Android-Malware-Permission-based-Dataset

REFERENCES

[1] B. STEGNER, "What Is an APK File and What Does It Do?," 12 December 2017. [Online]. Available: https://www.makeuseof.com/tag/what-is-apk-file/. [Accessed 09 May 2021].

[2] B. M. Mehtre, "Advances In Malware Detection-An Overview," *arXiv e-prints,* 2021.

[3] I. C. Education, "IBM Cloud Learn Hub," IBM, 19 August 2020. [Online]. Available: https://www.ibm.com/cloud/learn/supervised-learning. [Accessed 01 June 2021].

[4] B. Blazej, Osinski and Konard, "What is reinforcement learning? The complete guide," 5 July 2018. [Online]. Available: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/. [Accessed 01 June 2021].

[5] S. Mahdavifar, A. K. Fitriah, Andi, F. Rasool, A. Dima and A. G. Ali, "Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning," in *The 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC)*, 2020.

[6] D. Arp, M. Spreitzenbarth, M. Huebner, H. Gascon and K. Rieck, ""Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket"," in *21th Annual Network and Distributed System Security Symposium (NDSS),* February 2014.

[7] M. Spreitzenbarth, F. Echtler, T. Schreck, F. C. Freling and J. Hoffmann, ""MobileSandbox: Looking Deeper into Android Applications"," in *28th International ACM Symposium on Applied Computing (SAC)*, 2013.

[8] X. Liu, X. Du, X. Zhang, Q. Zhu, Z. H. Wang and M. Guizani, ""Adversarial Samples on Android Malware Detection Systems for IoT Systems"," *Sensors (Basel, Switzerland),* vol. 19, February 2019.

[9] Q. Wu, X. Zhu and B. Liu, "A Survey of Android Malware Static Detection Technology Based on Machine Learning," *Mobile Information Systems,* vol. vol. 2021, 2021.

[10] K. Allix, T. F. Bissyande, J. Klein and Y. L. Traon, "AndroZoo: collecting millions of Android apps for the research community," in *13th International Conference on Mining Software Repositories*, 2016.

[11] A. Kumar, V. Agarwal, S. K. Shandilya, A. Shalaginov, S. Upadhyay and B. Yadav, "Platform for Android Malware Classification and Performance Evaluation," in *IEEE,* Los Angeles, CA, USA, 2019.

[12] harrypro02, "GitHub," Wednesday June 2021. [Online]. Available: https://github.com/harrypro02/Android-Malware-Permission-Based-Dataset. [Accessed Wednesday June 2021].

[13] P. Joshi and G. Ciaburro, Python Machine Learning Cookbook - Second Edition, Packt Publishing, 2019.

[14] V. Kouliaridis, G. Kambourakis and T. Peng, "Feature Importance in Android Malware Detection," in *The 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2020)*, 2020.