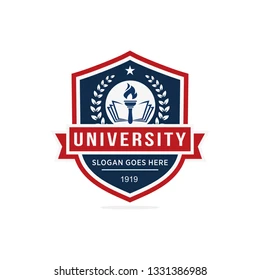
**Project Report**

**Android Malware Detection Project**

**First four page [front page thats include acknowledgment, support and logo of your college]**

## **PROJECT REPORT**

## **Android Malware Detection by Machine Learning Approach**



**Student Names**

**Year/Sem**

**Email ID**

**Android Malware Detection Using Machine Learning**

##### PROJECT REPORT

SUBMIT BY

##### **Student Name**

##### ROLL NUMBER

##### UNIVERSITY NAME

##### Under the guidance of MR. TEACHER NAME



**UNIVERSITY**

**ADDRESS**

Year 2023



BONAFIDE CERTIFICATE

Certified that this project titled **‘Android Malware Detection using Machine Learning’** is a bonafide work of  **STUDENT NAME** who carried out the research under my supervision.

HEAD OF DEPARTMENT FACULTY IN CHARGE

(SIGNATURE WITH DATE) (SIGNATURE WITH DATE)

## **Acknowledgement**

This document: I thank all my colleagues who provided a deep insight and expertise that greatly assisted in the document development.

I personally thank my Professor for the assistance with the initial drafts and the core structure of the proposal for the comments that greatly improved the manuscript.

I would also like to show my gratitude to the University for sharing its pearls of wisdom with me during the course of the development of this proposal. I am immensely grateful to everyone.



**ANDROID MALWARE DETECTION BY MACHINE LEARNING**

**ABSTRACT**

As the use of mobile devices continues to increase, so does the risk of malware attacks. Android malware has become a significant threat to mobile security, and detecting it has become a challenge for security researchers. In this study, we propose a malware detection framework using Machine Learning algorithms to classify Android applications as malware or benign. The proposed framework uses a combination of static and dynamic features extracted from the Android application's code and behaviour.

The study also conducted feature importance analysis, revealing the most significant features contributing to the detection of Android malware. The results show that dynamic features, such as API calls and system events, play a crucial role in detecting Android malware.

In conclusion, the proposed framework using SVM and Random Forest algorithm offers an effective approach to detecting Android malware. The study's results show the potential of the proposed approach in identifying malware threats in Android applications, which could help enhance mobile security and protect users from potential threats.

**SUMMARY**

Android Malware Detection is a significant part of endpoint security including workstations, servers, cloud instances, and mobile devices. Malware Detection is used to detect and identify malicious activities caused by malware. With the increase in the variety of malware activities on different files online and offline, It's Important for Data Security, Privacy and protection. So We will use Machine Learning and its algorithm to see the accuracy and prediction on Malware Datasets. In this Project we will use many different algorithms for analysing and studying the Malware in Dataset.

**INTRODUCTION**:

Idealistic hackers attacked computers in the early days because they were eager to prove themselves. Cracking machines, however, is an industry in today's world. Despite recent improvements in software and computer hardware security, both in frequency and sophistication, attacks on computer systems have increased. Regrettably, there are major drawbacks to current methods for detecting and analysing unknown code samples. The Internet is a critical part of our everyday lives today. On the internet, there are many services and they are rising daily as well. Numerous reports indicate that malware's effect is worsening at an alarming pace. Although malware diversity is growing, anti- virus scanners are unable to fulfil security needs, resulting in attacks on millions of hosts. Around 65,63,145 different hosts were targeted, according to Kaspersky Labs, and in 2015, 40,00,000 unique malware artefacts were found. Juniper Research (2016), in particular, projected that by 2019 the cost of data breaches will rise to $2.1 trillion globally. Current studies show that script-kiddies are generating more and more attacks or are automated. To date, attacks on commercial and government organisations, such as ransomware and malware, continue to pose a significant threat and challenge. Such attacks can come in various ways and sizes. An enormous challenge is the ability of the global security community to develop and provide expertise in cybersecurity. There is widespread awareness of the global scarcity of cybersecurity and talent. Cybercrimes, such as financial fraud, child exploitation online and payment fraud, are so common that they demand international 24-hour response and collaboration between multinational law enforcement agencies. For single users and

organisations, malware defence of computer systems is therefore one of the most critical cybersecurity activities, as even a single attack may result in compromised data and sufficient losses.

Mobile phones have become increasingly important tools in people’s daily life, such as mobile payment, instant messaging, online shopping, etc., but the security problem of mobile phones is becoming more and more serious. Due to the open source nature of the Android platform, it is very easy and profitable to write malware using the vulnerabilities and security defects of the Android system. This is the main reason for the rapid increase in the number of malware on the Android system. The malicious behaviors of Android malware generally include sending deduction SMS, consuming traffic, stealing user’s private information, downloading a large number of malicious applications, remote control, etc., threatening the privacy and property security of mobile phones users.

The number of Android malware is growing rapidly; particularly, more and more malicious software use obfuscation technology. Traditional detection methods of manual analysis and signature matching have exposed some problems, such as slow detection speed and low accuracy. In recent years, many researchers have solved the problems of Android malware detection using machine learning algorithms and had a lot of research results. With the rise of deep learning and the improvement of computer computing power, more and more researchers began to use deep learning models to detect Android malware. This paper proposes an Android malware detection model based on a hybrid deep learning model with deep belief network (DBN) and gate recurrent unit (GRU). The main contributions are as follows:

(i) In order to resist Android malware obfuscation technology, in addition to extracting static features, we also extracted the dynamic features of malware at runtime and constructed a comprehensive feature set to enhance the detection capability of malware.

(ii) A hybrid deep learning model was proposed. According to the characteristics of static features and dynamic features, two different deep learning algorithms of DBN and GRU are used.

(iii) The detection model was verified, and the detection result is better than traditional machine learning algorithms; it also can effectively detect malware samples using obfuscation technology.

**ANDROID APPLICATIONS**

Android applications, or simply "apps," are software programs designed to run on mobile devices powered by the Android operating system. Since its introduction in 2008, Android has become the most widely used mobile operating system, powering over 70% of smartphones and tablets worldwide. As a result, Android apps have become an essential component of modern life, used for everything from communication to entertainment, productivity, and health and fitness.

Android apps can be downloaded from the Google Play Store, which offers a wide range of free and paid apps. They can also be installed manually by downloading the APK (Android Package) file from a third-party source and installing it on the device. However, it is recommended to download apps from trusted sources only, such as the Google Play Store, to avoid malware or other security risks.

Android apps are built using the Java or Kotlin programming languages, with the Android SDK (Software Development Kit) providing the necessary tools and resources for developers to create apps that can run on Android devices. The SDK includes a range of libraries, APIs (Application Programming Interfaces), and tools that allow developers to create apps with features such as multimedia support, location-based services, push notifications, and more.

One of the key advantages of Android apps is their flexibility and customization. Developers can create apps for a variety of purposes and target specific audiences, from casual users to professionals. Apps can be designed to meet different needs, such as communication, entertainment, productivity, and education. This flexibility also allows developers to create apps that work across different devices, from smartphones and tablets to wearables and smart TVs.

Another advantage of Android apps is their availability. The Google Play Store offers a vast selection of apps, with over 3.5 million apps available as of 2021. This means users can find apps for almost anything they need, from social media and messaging to gaming, finance, and fitness. Many of these apps are also free or offer a free version with limited features, allowing users to try before they buy.

However, with so many apps available, it can be challenging to find the right app for a particular need. Users should research apps before downloading to ensure they are safe, reliable, and meet their needs. They should also be cautious when granting permissions to apps, as some apps may collect personal information or access device features without the user's knowledge.

In conclusion, Android apps have become an integral part of modern life, providing a range of features and services that can enhance productivity, entertainment, and communication. With the flexibility and customization offered by the Android SDK, developers can create apps for a wide range of purposes and target specific audiences. However, users should be cautious when downloading and using apps to ensure they are safe, reliable, and meet their needs.

**MALWARES AND THREATS IN ANDROID APPLICATIONS**

Malware and security threats are a growing concern for Android users as the popularity of Android devices continues to increase. Malware is malicious software that can infect an Android device and cause various issues such as data loss, unauthorized access, and financial loss. In this article, we will discuss some of the common types of malware and security threats that Android users may encounter while using apps.

**Trojan malware:**

1. Trojan malware is one of the most common types of malware that infects Android devices. It disguises itself as a legitimate app, and once it is installed, it can steal personal information, monitor the user's activities, and even control the device remotely. Trojan malware is often spread through social engineering techniques, such as phishing emails or fake app stores.

**Adware:**

1. Adware is a type of malware that displays unwanted ads on the device. These ads can be intrusive and annoying, and in some cases, they may also contain malicious content. Adware is often bundled with free apps or games, and it can be difficult to detect and remove.

**Spyware:**

1. Spyware is a type of malware that is used to spy on the user's activities. It can track the user's location, monitor their calls and messages, and even record their keystrokes. Spyware is often used for malicious purposes, such as stealing sensitive information or blackmailing the user.

**Ransomware:**

1. Ransomware is a type of malware that encrypts the user's files and demands payment in exchange for the decryption key. Ransomware attacks are becoming more common on Android devices, and they can cause significant financial loss and data loss.

**Malicious apps:**

1. Malicious apps are apps that are designed to infect the device with malware. They can be disguised as legitimate apps, such as games or productivity tools, and they may contain hidden malware that can harm the device or steal personal information.

**EVOLUTION OF MALWARE**  
In order to protect networks and computer systems from attacks, the diversity, sophistication and availability of malicious software present enormous challenges. Malware is continually changing and challenges security researchers and scientists to strengthen their cyber defences to keep pace. Owing to the use of polymorphic and metamorphic methods used to avoid detection and conceal its true intent, the prevalence of malware has increased. To mutate the code while keeping the original functionality intact, polymorphic malware uses a polymorphic engine. The two most common ways to conceal code are packaging and encryption . Through one or more layers of compression, packers cover a program's real code. Then the unpacking routines restore the original code and execute it in memory at runtime. To make it harder for researchers to analyse the software, crypters encrypt and manipulate malware or part of its code. A crypter includes a stub that is used for malicious code encryption and decryption. Whenever it's propagated, metamorphic malware rewrites the code to an equivalent. Multiple transformation techniques, including but not limited to, register renaming, code permutation, code expansion, code shrinking and insertion of garbage code, can be used by malware authors. The combination of the above techniques resulted in increasingly increasing quantities of malware, making time-consuming, expensive and more complicated forensic investigations of malware cases. There are some issues with conventional antivirus solutions that rely on signature-based and heuristic/behavioural methods. A signature is a unique feature or collection of features that like a fingerprint, uniquely differentiates an executable. Signature-based approaches are unable to identify unknown types of malware, however. Security researchers suggested behaviour-based detection to overcome these problems, which analyses the features and behaviour of the file to decide whether it is indeed malware, although it may take some time to search and evaluate. Researchers have begun implementing machine learning to supplement their solutions in order to solve the previous drawbacks of  
conventional antivirus engines and keep pace with new attacks and variants, as machine learning is well suited for processing large quantities of data.

1. **MALWARE DETECTION**  
   In such a way, hackers present malware aimed at persuading people to install it. As it seems legal, users also do not know what the programme is. Usually, we install it thinking that it is secure, but on the contrary, it's a major threat. That's how the malware gets into your system. When on the screen, it disperses and hides in numerous files, making it very difficult to identify. In order to access and record personal or useful information, it may connect directly to the operating system and start encrypting it

Detection of malware is defined as the search process for malware files and directories. There are several tools and methods available to detect malware that make it efficient and reliable. Some of the general strategies for malware detection are:

* + Signature-based
  + Heuristic Analysis
  + Anti-malware Software
  + Sandbox  
    Several classifiers have been implemented, such as linear classifiers (logistic regression, naive Bayes classifier), support for vector machinery, neural networks, random forests, etc.  
    Through both static and dynamic analysis, malware can be identified by:
  + Without Executing the code
  + Behavioural Analysis

1. **NEED FOR MACHINE LEARNING IN MALWARE  
   DETECTION**  
   Machine learning has created a drastic change in many industries, including cybersecurity, over the last decade. Among cybersecurity experts, there is a general belief that AI-powered anti-malware tools can help detect modern malware attacks and boost scanning engines. Proof of this belief is the number of studies on malware detection strategies that exploit machine learning reported in the last few years. The number of research papers released in 2018 is 7720, a 95 percent rise over 2015 and a 476 percent  
   increase over 2010, according to Google Scholar,1. This rise in the number of studies is the product of several factors, including but not limited to the increase in publicly labelled malware feeds, the increase in computing capacity at the same time as its price decrease, and the evolution of the field of machine learning, which has achieved ground-breaking success in a wide range of tasks such as computer vision and speech recognition. Depending on the type of analysis, conventional machine learning methods can be categorised into two main categories, static and dynamic approaches. The primary difference between them is that static methods extract features from the static malware analysis, while dynamic methods extract features from the dynamic analysis. A third category may be considered, known as hybrid approaches. Hybrid methods incorporate elements of both static and dynamic analysis. In addition, learning features from raw inputs in diverse fields have outshone neural networks. The performance of neural networks in the malware domain is mirrored by recent developments in machine learning for cybersecurity.

**Malware**, short for malicious software, consists of programming (code, scripts, active content, and other software) designed to disrupt or deny operation, gather information that leads to loss of privacy or exploitation, gain unauthorized access to system resources, and other abusive behaviour. It is a general term used to define a variety of forms of hostile, intrusive, or annoying software or program code. Software is considered to be malware based on the perceived intent of the creator rather than any particular features. Malware includes computer viruses, worms, Trojan horses, spyware, dishonest adware, crime-ware, most rootkits, and other malicious and unwanted software or programs .

In 2008, Symantec published a report that "the release rate of malicious code and other unwanted programs may be exceeding that of legitimate software applications.” According to F-Secure, "As much malware was produced in 2007 as in the previous 20 years altogether.”.

Since the rise of widespread Internet access, malicious software has been designed for a profit, for example forced advertising. For instance, since 2003, the majority of widespread viruses and worms have been designed to take control of users' computers for black-market exploitation. Another category of malware, spyware, - programs designed to monitor users' web browsing and steal private information. Spyware programs do not spread like viruses, instead are installed by exploiting security holes or are packaged with user-installed software, such as peer-to-peer applications.

Clearly, there is a very urgent need to find, not just a suitable method to detect infected files, but too build a smart engine that can detect new viruses by studying the structure of system calls made by malware.

**2. Current Antivirus Software**

Antivirus software is used to prevent, detect, and remove malware, including but not limited to computer viruses, computer worm, Trojan horses, spyware and adware. A variety of strategies are typically employed by the anti-virus engines. Signature-based detection involves searching for known patterns of data within executable code. However, it is possible for a computer to be infected with new virus for which no signatures exist. To counter such “zero-day” threats, heuristics can be used, that identify new viruses or variants of existing viruses by looking for known malicious code. Some antivirus can also make predictions by executing files in a sandbox and analysing results.

Often, antivirus software can impair a computer's performance. Any incorrect decision may lead to a security breach, since it runs at the highly trusted kernel level of the operating system. If the antivirus software employs heuristic detection, success depends on achieving the right balance between false positives and false negatives. Today, malware may no longer be executable files. Powerful macros in Microsoft Word could also present a security risk. Traditionally, antivirus software heavily relied upon signatures to identify malware. However, because of newer kinds of malware, signature-based approaches are no longer effective.

Although standard antivirus can effectively contain virus outbreaks, for large enterprises, any breach could be potentially fatal. Virus makes are employing "oligomorphic", "polymorphic" and, "metamorphic" viruses, which encrypt parts of themselves or modify themselves as a method of disguise, so as to not match virus signatures in the dictionary.

Studies in 2007 showed that the effectiveness of antivirus software had decreased drastically, particularly against unknown or zero day attacks. Detection rates have dropped from 40-50% in 2006 to 20-30% in 2007. The problem is magnified by the changing intent of virus makers. Independent testing on all the major virus scanners consistently shows that none provide 100% virus detection.

**Static and dynamic analyses**

Static and dynamic analyses are sources for security features, which the secu- rity investigator uses to decide the maliciousness of a given application. Manual inspection of these features is a tedious task and could be automated using ma- chine learning techniques. For this reason, the majority of the state-of-the-art malware detection solutions use machine learning techniques. Developers must carefully choose between the two analysis techniques depending upon the re- quirements of the model and the kind of applications the model is expected to deal with.

Static analysis is performed in a non-run-time environment. It basically in- volves looking at the application from inside out without executing the program, but rather by examining the source code, byte code or application binaries for signs of security vulnerabilities. In the static analysis, the application data and control paths are modeled and then analyzed for security weaknesses. Dynamic analysis adopts the opposite approach and is executed while a program is in op- eration. It looks at the application by examining it in its running state and trying to manipulate it in order to discover security vulnerabilities. The behaviour of the application in dynamic analysis is studied by emulating it on a sandbox, with a monkeyrunner giving random inputs to the application.

Static analysis, with its white-box visibility, is certainly the more thorough ap- proach and may also prove more time and cost efficient with the ability to detect malware. Static analysis can also unearth some properties that would not emerge in a dynamic test. Dynamic analysis, on the other hand, is capable of exposing a subtle vulnerability too complicated for static analysis alone to reveal, espe- cially in cases of code encryption and obfuscation. A dynamic test, however, will only find defects in the part of the code that is actually executed. The developer must weigh up these considerations with the complexities of their own situation in mind while choosing the best analysis technique. Application type, time, and availability of resources are some of the primary concerns.

**Work has been described as**

* Describing the details: The dataset is imported and the different columns are discussed in the dataset.
* Data cleaning: The required steps are taken after examining the dataset so that the dataset can be cleaned and all the null values and columns of not much significance are removed so that they will not be of any concern in the training part.
* Data Training and Testing: When the information is transparent and ready for training, we spilled the information as a training dataset and testing dataset in an 80:20 ratio so that the data was spilled in an 80:20 ratio.  
  In this paper, as we try to achieve the highest accuracy, we use two algorithms to see which will give us better precision.
* Applying Different Algorithms[ML Algorithms]

**MAIN ALGORITHMS APPLIED:**

1. RANDOM FOREST
2. SVM

1**. SVM Algorithm**

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

You can look at support vector machines and a few examples of their working here.

## How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”. Don’t worry, it’s not as hard as you think!

Let’s understand:

* Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.  
  You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.
* Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?  
  Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called Margin. Let’s look at the below snapshot:  
  Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.
* Identify the right hyper-plane (Scenario-3):Hint: Use the rules as discussed in previous section to identify the right hyper-plane

Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

* Can we classify two classes (Scenario-4)?: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.   
  As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.  
  
* Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can’t have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.  
  SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z:  
    
  In above plot, points to consider are:
  + All values for z would be positive always because z is the squared sum of both x and y
  + In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

2. Random Forest

Random Forest is a machine learning algorithm that is widely used for both classification and regression tasks. It is a type of ensemble learning algorithm that combines multiple decision trees to improve the accuracy and robustness of the model. In this article, we will discuss the principles of the Random Forest algorithm, its advantages, and its applications.

Principles of Random Forest Algorithm

The Random Forest algorithm creates a collection of decision trees, where each tree is built independently and with a random subset of the data. The algorithm uses a technique called bagging (Bootstrap Aggregating) to create a diverse set of decision trees. In the bagging process, multiple samples are drawn randomly with replacement from the original dataset. These samples are then used to train individual decision trees.

To build each decision tree in the Random Forest, the algorithm first randomly selects a subset of features from the input data. It then constructs the tree using these features and their corresponding data points. This process is repeated for every tree in the ensemble, resulting in a set of decision trees that are diverse and uncorrelated.

Advantages of Random Forest Algorithm

The Random Forest algorithm offers several advantages over other machine learning algorithms:

1. Robustness: Random Forest is less prone to overfitting than other algorithms, such as decision trees. The algorithm uses bagging and feature selection to create a diverse set of decision trees that reduce the risk of overfitting.
2. Accuracy: Random Forest produces highly accurate results, even when working with noisy or incomplete data. The algorithm is also able to handle large datasets efficiently.
3. Versatility: Random Forest can be used for both classification and regression tasks. It can also handle both categorical and continuous input data.
4. Interpretability: Random Forest provides information on the importance of each feature in the model, making it easier to interpret the results and make informed decisions.

Applications of Random Forest Algorithm

The Random Forest algorithm has a wide range of applications in various fields, including:

1. Healthcare: Random Forest can be used to analyze medical data, such as patient records and clinical trials, to predict disease outcomes and identify potential treatments.
2. Finance: Random Forest can be used to predict stock prices, identify fraudulent transactions, and assess credit risk.

Random Forest is a machine learning algorithm that uses ensemble learning to improve the accuracy and robustness of the model. It is a type of decision tree-based algorithm that combines multiple decision trees to make a prediction. The algorithm works by creating a collection of decision trees, where each tree is built independently and with a random subset of the data. The final prediction is made by averaging the predictions of all the trees in the forest.

The formula for Random Forest algorithm can be broken down into the following steps:

1. Selecting the features: Random Forest algorithm randomly selects a subset of features from the input data. This is done to reduce the correlation between the trees and improve the accuracy of the model.

2. Building decision trees: Random Forest algorithm builds multiple decision trees on the selected subset of features and data points. Each decision tree is constructed independently and randomly.

3. Training the model: The algorithm uses the training dataset to build the decision trees. The trees are built using the selected features and a subset of the training data. The algorithm repeats this process for each tree in the forest.

4. Making predictions: Once the decision trees are built, the algorithm uses them to make predictions on the test dataset. The final prediction is made by taking the average of the predictions of all the trees in the forest.

The Random Forest algorithm has several advantages over other machine learning algorithms. First, it is less prone to overfitting than other algorithms, such as decision trees. The algorithm uses bagging and feature selection to create a diverse set of decision trees that reduce the risk of overfitting. Second, Random Forest produces highly accurate results, even when working with noisy or incomplete data. Third, Random Forest can be used for both classification and regression tasks. Fourth, Random Forest provides information on the importance of each feature in the model, making it easier to interpret the results and make informed decisions.

The Random Forest algorithm is widely used in various fields, including healthcare, finance, marketing, and ecology. It can be used to predict disease outcomes, stock prices, customer behavior, and ecological patterns, among other things.

In summary, the Random Forest algorithm is a powerful machine learning algorithm that uses ensemble learning to improve the accuracy and robustness of the model. It works by creating a collection of decision trees, where each tree is built independently and with a random subset of the data. The final prediction is made by averaging the predictions of all the trees in the forest. The algorithm has several advantages over other machine learning algorithms and is widely used in various fields.

**Codes :**

**mode.py**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import pickle

import torch

from sklearn import svm

from sklearn import tree

import pandas as pd

from sklearn.externals import joblib

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.ensemble import RandomForestClassifier

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

data = pd.read\_csv("android\_traffic.csv", sep=";")

data = data.drop(['duracion','avg\_local\_pkt\_rate','avg\_remote\_pkt\_rate'], axis=1).copy()

data=data[data.tcp\_packets<20000].copy()

data=data[data.dist\_port\_tcp<1400].copy()

data=data[data.external\_ips<35].copy()

data=data[data.vulume\_bytes<2000000].copy()

data=data[data.udp\_packets<40].copy()

data=data[data.remote\_app\_packets<15000].copy()

data[data.duplicated()].sum()

data=data.drop('source\_app\_packets.1',axis=1).copy()

scaler = preprocessing.RobustScaler()

scaledData = scaler.fit\_transform(data.iloc[:,1:11])

scaledData = pd.DataFrame(scaledData, columns=['tcp\_packets','dist\_port\_tcp','external\_ips','vulume\_bytes','udp\_packets','source\_app\_packets','remote\_app\_packets',' source\_app\_bytes','remote\_app\_bytes','dns\_query\_times'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaledData.iloc[:,0:10], data.type.astype("str"), test\_size=0.25, random\_state=45)

rdF=RandomForestClassifier(n\_estimators=250, max\_depth=50,random\_state=45)

rdF.fit(X\_train,y\_train)

# Saving model to disk

pickle.dump(rdF, open('modelling.pkl','wb'))

# Loading model to compare the results

model = pickle.load(open('modelling.pkl','rb'))

# pred=rdF.predict(X\_test)

**App.py**

import numpy as np

from flask import Flask, request, jsonify, render\_template

import pickle

app = Flask(\_\_name\_\_)

model = pickle.load(open('modelling.pkl', 'rb'))

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict',methods=['POST'])

def predict():

'''

For rendering results on HTML GUI

'''

int\_features = [int(x) for x in request.form.values()]

final\_features = [np.array(int\_features)]

prediction = model.predict(final\_features)

output = prediction[0]

return render\_template('index.html', prediction\_text='This Android is of {} nature'.format(output))

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**Conclusion:**

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

**REFERENCES:**

[1]http://www.us-cert.gov/control\_systems/pdf/undirected\_attack0905.pdf

[2] "Defining Malware: FAQ". http://technet.microsoft.com. Retrieved 2009-09-10.

[3] F-Secure Corporation (December 4, 2007). "F-Secure Reports Amount of Malware Grew by 100% during 2007". Press release. Retrieved 2007-12-11.

[4] History of Viruses. http://csrc.nist.gov/publications/nistir/threats/subsubsection3\_3\_1\_1.html [5] Landesman, Mary (2009). "What is a Virus Signature?” Retrieved 2009-06-18.

[6] Christodorescu,M., Jha, S., 2003. Static analysis of executables to detect malicious patterns. In: Proceedings of the 12th USENIX Security Symposium. Washington .pp. 105-120.

[7] Filiol, E.,2005. Computer Viruses: from Theory to Applications. New York, Springer, ISBN 10: 2- 287-23939-1.

[8] Filiol, E., Jacob, G., Liard, M.L., 2007: Evaluation methodology and theoretical model for antiviral behavioral detection strategies. J. Comput. 3, pp 27–37.

[9] Cai, H.; Ryder, B.G. Understanding Android application programming and security: A dynamic study. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 364–375.

[10] Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* **2020**, *8*, 124579–124607.

[11] Gilski, P.; Stefanski, J. Android os: A review. *Tem J.* **2015**, *4*, 116.

[12] Garg, S.; Baliyan, N. Comparative analysis of Android and iOS from security viewpoint. *Comput. Sci. Rev.* **2021**, *40*, 100372.

[13] Odusami, M.; Abayomi-Alli, O.; Misra, S.; Shobayo, O.; Damasevicius, R.; Maskeliunas, R. Android malware detection: A survey. In *International Conference on Applied Informatics*; Springer: Cham, Switzerland, 2018; pp. 255–266.

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

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

[16] R. Surendran, T. Thomas, and S. Emmanuel, “A TAN based hybrid model for android malware detection,” *Journal of Information Security and Applications*, vol. 54, article 102483, 2020.

[17] W.-C. Kuo, T.-P. Liu, and C.-C. Wang, “Study on android hybrid malware detection based on machine learning,” in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp. 31–35, Singapore, 2019.

[18] H. Zhang, S. Luo, Y. Zhang, and L. Pan, “An efficient android malware detection system based on method-level behavioral semantic analysis,” *IEEE Access*, vol. 7, pp. 69246–69256, 2019.