Journal of Engineering Sciences

Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

Malicious Application Detection Using Machine Learning

1G.Chandana ,2B. Anusha , 3S. Sri Devi , 4Mrs. V. Sri Suma

B.tech Student, Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

CMR TECHNICAL CAMPUS, Hyderabad

ABSTRACT

Android plays a vital role in the today's market. According to recent survey placed nearly 84.4% of people stick to android which explosively become popular for personal or business purposes. It is no doubt that the application is extremely familiar in the market for their amazing features and the wonderful benefits of android applications makes the users to fall for it. Android imparts significant responsibility to application developers for designing the application with understanding the risk of security issues. When concerned about security, malware protection is a major issue in which android has been a major target of malicious applications. In android based applications, permission control is one of the major security mechanisms. In this project, the permission induced risk in application, and the fundamentals of the android security architecture are explored, and it also focuses on the securit y ranking algorithms that are unique to specific applications. Hence, we propose the system providing the detection of malware analysis based on permission and steps to mitigate from accessing unwanted permission (limits the permission). It is also designed to reduce the probability of vulnerable attacks.

I.INTRODUCTION

1.1OBJECTIVE OF THE PROJECT

In recent years, the usages of smart phones are increasing steadily and also growth of Android application users are increasing. Due to growth of Android application user, some intruder are creating malicious android application as tool to steal the sensitive data and identity theft / fraud mobile bank, mobile wallets. There are so many malicious applications detection tools and software are available. But an effectively and efficiently malicious applications detection tools needed to tackle and handle new complex malicious apps created by intruder orhackers. In this paper we came up with idea of using machine learning approaches for detectingthe malicious android application. First we have to gather dataset of past malicious apps as training set and with the help of Support vector machine algorithm and decision tree algorithm make up comparision with training dataset and trained dataset we can predict the malware android apps. With an estimated market share of 70% to 80%, Android has become the most popular operating system for smartphones and tablets. Unsurprisingly, cyber-criminals have followed, expanding their malicious activities to mobile platforms. Mobile threat researchers have recognized an alarming increase of Android malware from 2012 to 2013 and estimate thatthe number of detected malicious applications is in the range of 120,000 to 718,000. To efficiently detect malware from applications available from official and third-party sources, many efforts have contributed to studying the nature of smartphone platforms and theirapplications in the past decade. The Android platform employs the permission system to restrictapplications privileges to secure the sensitive resources of the users. The developer is responsible for determining appropriately which permissions an application requires, but an application needs to get a user's approval of the requested permissions to access private or otherwise-restricted resources. Although the permission system can protect users from applications with invasive behaviors, its effectiveness highly depends on a user's comprehension of the consequences of granting a permission. According to recent studies, many users do not understand what each permission means and blindly grant them, potentiallyallowing an application to access sensitive/private information. Another laws that the user cannot decide to grant single permissions, while denying others. Many users, although an app might request a suspicious permission among much seemingly legitimate permission, will stillconfirm the installation. The Android security model is based mainly on permissions.

1.2PURPOSE OF THE PROJECT

The ultimate aim of the project is to improve permission for detecting the malicious android mobile application using machine learning algorithms. As a result, the implementation of these permissions is of interest to us. An Android permission is a restriction limiting accessto a part of the code or to data on the device. The limitation is imposed to protect critical data and code that could be misused to distort or damage a user's experience. Permissions are also used to allow or

restrict application access to restricted APIs and resources. For example, the Android 'INTERNET' permission is required by apps to perform network communications so, opening a network connection is restricted by the 'INTERNET' permission. Furthermore, an application must have the 'READ CONTACTS' permission in order to read entries in a user'sphonebook as well. To require a permission, the developer specifies them using the Manifest file in declaring a "" attribute. The "android : name" field specifies the name of the permissionin the code.

1.3 PROJECT FEATURES

A new method to detect malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself. Features used to classify are the presence of tags uses-permission and uses-feature into the manifest as well as the number of permissions of each application. These features are the permission requested individually and the «uses- feature» tag the possibility of detection malicious Android applications based on permissions and 20 features from Android application packages.

II.LITERATURE SURVEY

We studied the techniques that are proposed to identify Android malwares. In his work, Anshul et al. presented an idea to detect Android Malwares by Network traffic analysis. Their approach is used to identify malware on Android that is operated by a remote server. These malwares either accept orders from the server or leak sensitive data to it. First, they analyzed the network traffic of android malwares and then the traffic of normal applications. They discovered the characteristics that distinguish malware traffic from non-malware traffic.. And in the second phase, they built a classifier using these network traffic features which can detect the malwares. In another work, Anshul et al. proposed a technique called the PermPair method. They approached the goal by considering every pair of permissions as the possible input feature and finally decided on each pair, if that combination is vulnerable. Their method includes data sets from 3 different sources called Genome ,Debris and Koodoos. Their approach had 3 phases. In the first phase, they constructed 4 different graphs by extracting permission pairs from each application. Out of the 4 graphs, 3 graphs are for malwares and 1 graph is for benign applications. In the second phase, they dealt with merging 3 malicious graphs into a single malicious graph. At the end of this phase, they ended up with two graphs, one for malicious and one for benign. In the third and final phase of their method, they developed a model that calculates two scores called normal score and malicious score for every application and decides whether a particular application is malware or not. The most commonly used properties in static and dynamic Android malware detection are permissions and network traffic features respectively. Static permissions cannot identify sophisticated malware, which is capable of update attacks. And coming to dynamic network traffic, it cannot detect malware samples without a network connection. Therefore, a hybrid model integrating both of these properties is proposed. They extracted both permissions and network traffic features and made them into a single vector. Using the K-medoids method, they partitioned the vectors into K clusters. And they used the K-Nearest Neighbours method, to classify whether a particular application is malicious or not. They made sure that K is odd, just to make sure out of K nearest neighbours, the count of malicious and benign neighbours is not the same. In another work, Zhenlong Yuan et al. proposed a technique to associate static features with dynamic features and then classify the given android applications as malicious or safe. They got the features they used as input to their model in three stages: • Static Phase • Sensitive APIs • Dynamic Phase Static phase includes the permissions that are obtained by unzipping the apk file and parsing xml files obtained later. Another file classes.dex accounts for the sensitive api calls.

III.SYSTEM ANALYSIS

3.1PROBLEM STATEMENT

Smartphones have become the most used device in one's day to day life. They facilitateusers with a variety of applications that are enriched with powerful features. It is almost impossible for anyone these days to spend a day without their smartphones. Out of allsmartphones, Android smartphones are the ones that are widely used. This increasing popularity of Android smartphones has also attracted malicious attackers. This malicious activity can be done by either a single application or a group of applications working together.The objective of this project is to create a model that can detect such malicious applications.

3.2 EXISTING SYSTEM

Traditionally Numerous malware detection tools have been developed, but some tools are may not able to detect newly created malware application and unknown malware application infected by various Trojan, worms, spyware. Detecting of large number of

malicious application over millions of android application is still a challenging task using traditional way. In existing, Non machine learning way of detecting the malicious application based on characteristics, properties, behavioral.

DISADVANTAGES OF THE EXISTING SYSTEM

Identification of newly updated or created malicious application is hard to findout.

Non Machine learning approaches are not reliable and efficient.

In Existing approaches covers only 30 permissions out of 300 app permissions,due to this limited apps permissions different types of attacks can occur.

3.3PROPOSED SYSTEM

In proposed paper, we implement SIGPID, Significant Permission Identification (SIGPID). The goal of the sigpid is to improve the apps permissions effectively and efficiently. This SIGID system improves the accuracy and efficient detection of malware application. With help machine learning algorithms such as SVM and Decision Tree algorithms make a comparison between training and trained datasets. Support vector machine algorithms act as a classifier which is used to classify malicious application and benign app.

ADVANTAGES OF THE PROPOSED SYSTEM

Improves the percentages of detection malicious application

Machine learning is better efficient than Non machine learning algorithm.

Able to detect new malware android applications.

We only need to consider 22 out of 135 permissions to improve the runtimeperformance by85.6.

IV.ARCHITECTURE

4.1PROJECT ARCHITECTURE

Web applications are by nature distributed applications, meaning that they are programsthat run on more than one computer and communicate through network or server. Specifically,web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can beaccessed and used by millions of people.

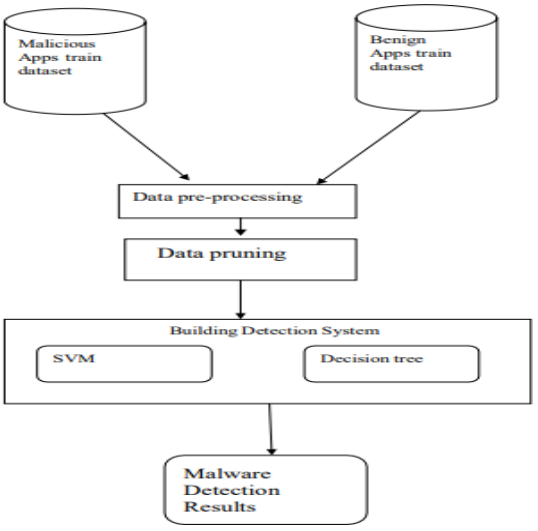


Figure no. 3.1 Project Architecture

4.2MODULE DESCRIPTION

1. Permission

Permission characterize existing Android malware from various aspects, including the permissions requested. They identified individually the permissions that are widely requested in both malicious and benign apps

2. Combination of Permission

This method on network classification helps to define irregular permission combinations requested by abnormal applications. The nature, sources and implications of sensitive data on Android devices in enterprise settings.

3. Feature Extraction

A new method to detect malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself.

4. Classification

According to them, by combining results from various classifiers, it can be a quick filter to identify more suspicious applications. And propose a framework that intends to develop a machine learning-based malware detection system on Android to

detect malware applications and to enhance

V.IMPLEMENTATION METHODOLOGY

To classify malicious application from benign application a decent dataset is required.The dataset can be downloaded from debrin dataset. We construct massive experiments, including 516 benign applications and 528 malicious applications. In this section the methodology followed is discussed in details.

Dataset

Any machine learning model needs a dataset over which it can be trained. So data collection is one of the most important steps. We've worked on 3 different datasets and compared their result against each other

1st dataset is collected from google comprised of 70 different application eachhaving a set of 17 permission

2nd is downloaded from Kaggle which has 184 different permissions or we can say features list for 29999 apps individually.

3rd one is downloaded from Kaggle. It has 138047 records with each record consisting 57 columns(permissions)

For training and testing purposes, we split the dataset into two parts. We used 80% of the dataset for training the machine learning model and the remaining 20% dataset was used for testing every machine learning model and calculating the performance of each model with metrics such as accuracy, f1-score, precision and recall.

Feature Engineering

The feature set used for training has a big impact on machine learning. Several research have found that certain features are helpful in training machine learning-based malware classifiers. That is the reason we have used feature engineering in ourimplementation. In supervised learning, we will use Feature engineering, which is the process of selecting, manipulating, and changing raw data into features. We use Feature Engineering for the following reasons:

To remove imputation

Handling outliers

To achieve Normalization

Null Value Handling

To remove invalid data

To achieve scaling

Data Preprocessing

The process of converting raw data into a comprehensible format is known as data preparation. We can't work with raw data, thus this is a key stage in machine learning. Before using machine learning or data mining methods, make sure the data isof good quality.The purpose of data preprocessing is to ensure that the data is of good quality. The following criteria can be used to assess quality accuracy, completeness, consistency, trustworthy, understandability. Data Preprocessing involves the followingsteps:

Data Cleaning: Correcting or deleting incorrect, corrupted, improperly formatted, duplicate, or incomplete data. We have removed those columns having missing values. We've removed any undesirable observations from our datasets, such as duplicates or irrelevant observations

Data Transformation: Changing data from one format to another. For string columns and decimal columns such as price, they're converted to binary. • Data integration: combining data from a variety of sources, including databases (bothrelational and non-relational), data cubes, files, and so on.

Data reduction: It is possible to reduce the amount of records, characteristics, ordimensions. It is carried out during feature selection using correlation matrix.

We have worked on last dataset in detail and for remaining we have tested accuracy and compared its result. We made sure that the dataset contained enough examples for both the malware and benign applications. There are 41323 examples for benign applications and 96724 applications for malicious applications. So, we can say that the dataset is not skewed. Each permission column is a binary column,indicating a permission is asked or not.

A SVM Linear classifier is built to fit the data you supply and provide a hyperplane that fits well and classify your data into different classes. Following that, you may input some attributes to your classifier to check what the projected class is once you've obtained the hyperplane. Support Vectors are the points which can be considered as edge cases. They are very nearer to the hyperplane. The two support vectors

corresponding to either classes benign and malicious respectively are equidistant to the hyperplane with maximum margin possible. In classification using Support vector machines, the model with polynomial kernel performs merely when the positive examples and negative examples in the data are overlapping. One way to deal with this overlapping data is to use a support vector machine with a radial kernel generally known as Radial Basis Function(RBF). When RBF kernel is used in SVM, radial kernel behaves like a weighted nearest neighbor model. In other words, the nearest observation has a lot of influence on how we classify the new example. The value obtained after substituting in radial kernel function is inversely proportional to the closeness. The radial kernel function of two data observations a,b is as below.

$$RBF(a, b) = e^{-\gamma(a-b)^2}$$

Decision Trees are a non-parametric supervised learning approach which can be used for classifying problems as well as regression problems. The sole objective is to construct a machine learning model that guesses the class of a given instance by learning basic decision rules from feature values. First, we will calculate the entropy. It is also known as measure of uncertainity. Then for each attribute A, we calculate information gain. The attribute with maximum value for information gain will be selected as the root node and this process continues. The formulas for entropy and Information gain are:

$$E(S) = -[p\log(p) + (1-p)\log(1-p)]$$

A collection of classification algorithms which are based on a theorem named after Bayes together forms a Naive Bayes classifier. It isn't a single algorithm. It is a group of algorithms sharing one common principle i.e., every pair of features which are used in classification are independent of each other. For two events Ea, Eb Bayes theorem tells that:

$$P(Ea|Eb) = P(Eb|Ea) * P(Ea)\ P(Eb)$$

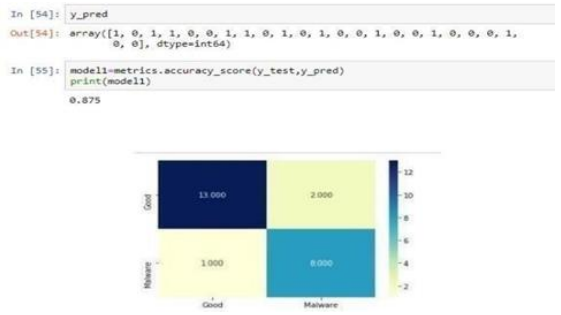
Using this principle for classification task, we can say that:

$$P\ r(class|attributes) = P\ r(attributes|class) * P\ r(class)\ P\ r(attributes)$$

As we have two classes malicious and safe, we will calculate the probabilities for the application to be in malicious class and to be in safe class. The class for which the probability value is higher, is the class to which the application belongs to.

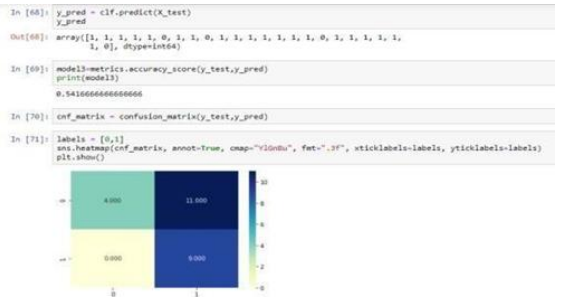
VI. SCREENSHOTS

After building the svm model, we have trained and tested the data using the built svmmodel and obtained a accuracy of 87.5

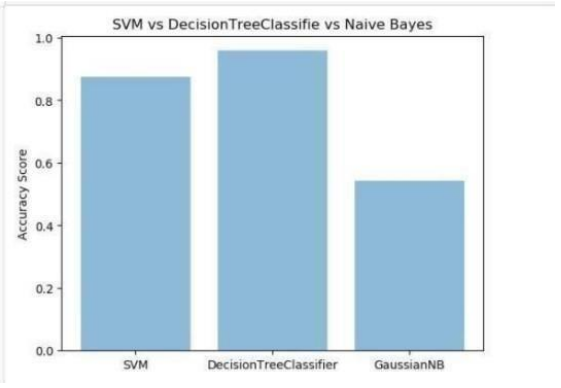


Screenshot no 6.1 Building SVM Model.

After building the Bayesian model, we have trained and tested the data using the built Bayesian model and obtained a accuracy of 54.1



Screenshot no 6.2 Building Naïve Bayes Classification Model.

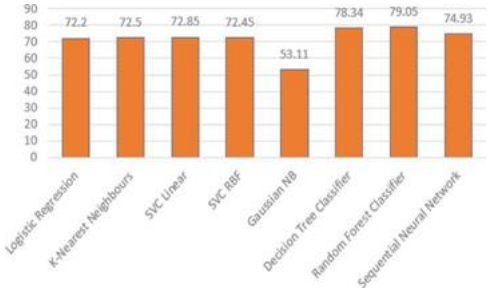


Screenshot no 6.3 SVM vs Decision Tree vs Naïve Bayes

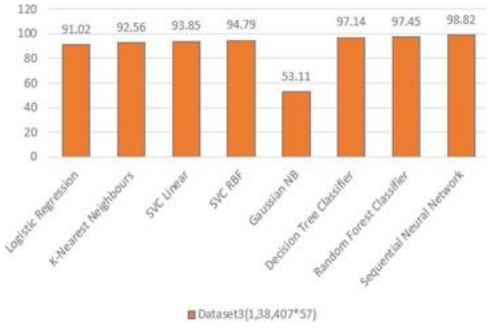
Screenshot no. 6.4 Learning curve

S.No	Model Name	Testing Accuracy	
		Dataset 1 (29999*184)	Dataset 2 (70*17)
1	Logistic Regression	72.2	75.13
2	K Nearest Neighbors	72.5	83.3
3	SVM Linear	72.85	83.3
4	SVM RBF	72.45	66.6
5	Gaussian NB	53.11	62.5
6	Decision Tree	78.34	91.6
7	Random Forest	79.05	98.21
8	Sequential Neural Network	74.93	70.83

Screenshot no. 6.5 Performance of different models on different datasets



Screenshot no. 6.6 Dataset 1



Screenshot no. 6.7 Dataset 2

VII.CONCLUSION

In conclusion, our project can identify, with moderate success, applications that pose a potential threat based on the permissions that they request. Our application can scan applications on a phone at any time, and alerts the user to do so when an installation or app update occurs. We believe that this is an important step in preventing Android malware, because this application brings to the user's attention all the possibly dangerous applications, allowing them to scrutinize the applications that they trust more carefully. This in turn will helpusers become more security-conscious overall. Even so, this is only a first step. Future work for this project will include increasing the accuracy of the classifier, migrating the Python portions of this project to Java, and integrating more advanced methods of detecting maliciousbehavior such as looking at API calls (this follows a "defense in depth" strategy). One benefit of the decision tree classifier is its speed. It can serve as a preliminary screen for more advanced but slower methods, to focus the applications they will inspect. Lastly, taking into account application categories such as being a game or email-client would also help detect suspicious permissions and behaviors. But, a set of android applications operating together can carry out a malicious activity. We call them colluding apps. In this, the malicious activity is carried out by more than one application. Each application participating in collusion does a small part of the malicious action. These applications communicate with each other through covert channels. Sometimes when a malicious activity cannot be performed by a single application, it might be possible that a group of applications coordinating with each other can perform that malicious activity. This phenomenon is called Application Collusion. It is an emerging threat. The reason behind why we are calling this as an emerging threat is because most of the android malware detectors scan the applications individually when determining whether it is a malware or not. But as the malicious activity here is being carried out by a group of applications, those traditional detectors cannot detect this. So, we need a model to detect these colluding applications. Till now, very little research has been done on this and there is scarcity for datasets. We are trying to create or obtain a few applications that can perform collusion, so that we can do some research on them which may eventually help in creating a model that can detect colluding applications. Firstly, we have to obtain a template for collusion. Then, we have to try to split a malicious task into various steps and make each application perform one of the steps. By doing this, we can achieve collusion.

VIII.FUTURESCOPE

We are trying to create or obtain a few applications that can perform collusion, so that we can do some research on them which may eventually help in creating a model that can detect colluding applications. Firstly, we have to obtain a template for collusion. Then, we have to try to split a malicious task into various steps and make each application perform oneof the steps. By doing this, we can achieve collusion.

REFERENCES

1. A. P. Felt, K. Greenwood, and D. Wagner, “The effectiveness of install-time permission systems forthird-partyapplications”,2010.

2. B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android permissions: aperspective combining risks and benefits,” 2012.

3.Y. Zhou and X. Jiang, “Dissecting android malware: Characterization andevolution,2012.

4. V. Rastogi, Y. Chen, and X. Jiang, “Droidchameleon: evaluating android antimalwareagainsttransformation attacks, 2013.

5. G. Canfora, F. Mercaldo, and C. A. Visaggio, “A classifier of malicious android applications,”2013.

6. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. A´ lvarez, “Puma:Permission usage to detect malware in android,”,2013.

7. C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance Evaluation on PermissionBased Detection for Android Malware,”2013.

8. Franklin Tchakount´, Computers & Security “Permission-based Malware Detection Mechanisms on Android: Analysis and Perspectives”,2014.

9.Z. Fang, W. Han, and Y. Li, “Permission-based Android security: Issues and counter measures,”

10. W. Enck, M. Ongtang, P. McDaniel, "Understanding Android Security"