Malicious application detection using Machine Learning

Safura Sana, 2Bushra Tabassum, 3mohd aamir khan, 4Mukaramuddin

1, 2, 3 UG Scholar, Department of Information Technology, ISL engineering college, Hyderabad, India.
HOD, Dept, Dept of Humanities and sciences, ISL engineering college, Hyderabad, India.

1 Introduction

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 that the 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 their applications in the past decade. The Android platform employs the permissions system to restrict applications’ access to secure the sensitive resources of the users. The developer is responsible for determining 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, potentially allowing an application to access sensitive/private information. Another law 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 still confirm the installation. The Android security model is based mainly on permissions. As a result, the
implementation of these permissions is of interest to us. An Android permission is a restriction limiting access to 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’s phonebook as well. To require a permission, the developer specifies them using the Manifest file in declaring a "<uses-permission>" attribute. The "android:name" field specifies the name of the permission in the code. A permission can be associated with one of the following four protection levels:

Normal: A low-risk permission which allows applications to access API calls (‘SET WALLPAPER’) causing no harm to users.

Dangerous: A high-risk permission which allows applications to access potential harmful API calls (‘READ CONTACTS’) such as leaking private user data or control over smartphone device. Dangerous permissions are explicitly shown to the user before an app is installed and the user must decide to grant the permissions or not, determining whether the installation continues or fails, respectively.

Signature: A permission which is granted if its requesting application is signed with the same certificate as the application which defines the permission is signed.

Signature-or-system: A permission which is granted only if its requesting application is in the same Android system image or is signed with the same certificate as the application which defines the permission is signed.
Module

1. Permission
2. Combination of Permission
3. Feature Extraction
4. Classification

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. According to this work, malicious apps clearly tend to request more frequently on the SMS-related permissions, such as ‘READ SMS’, ‘WRITE SMS’, ‘RECEIVE SMS’, and ‘SEND SMS’. They found that malicious apps tend to request more permissions than benign ones. They found no strong correlation between applications categories and requested permissions, and introduce a method to visualize permissions usage in different app categories. The aim of their work is to classify Android applications into several categories such as entertainment, society, tools, and productivity, multimedia and video, communication, puzzle and brain games. Mentions a method that analyses manifest files in Android application by extracting four types of keyword lists:(1) Permission, (2) Intent filter (action), (3) Intent filter (category), and (4) Process name. This approach determines the malignancy score by classifying individually permissions as malicious or benign.

Combination of Permission:
A high-level contextual analysis and an exploration of Android applications based on their implementation of permission-based security models by applying network visualization techniques and clustering algorithms. From that, they discovered new potentials in permission-based security models that may provide additional security to the users. 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. They characterized malicious apps and the risks they pose to enterprises. Finally, they have proposed several approaches for dealing with security risks for enterprise. From the analysis of third-party applications, Permission additions dominate the evolution of third-party apps, of which Dangerous permissions tend to account for most of the changes. a method for detecting malware based on three metrics, which evaluate: the occurrences of a specific subset of system calls, a weighted sum of a subset of permission that the application required, and a set of combinations of permissions.

**Feature Extraction:**
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.

**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 security and privacy of smartphone users. This system monitors various permission-based features and events obtained from the android applications, and analyses these features by using machine learning classifiers to
classify whether the application is benign or malware. Once, the Support Vector Machine trained offline on a dedicated system and only it is transferred the learned model to the smartphone for detecting malicious applications.

**OBJECTIVE:**

The ultimate aim of the project is to improve permission for detecting the malicious android mobile application using machine learning algorithms.

**SYNOPSIS:**

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 or hackers.

In this paper we came up with idea of using machine learning approaches for detecting the 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 comparsion with training dataset and trained dataset we can predict the malware android apps upto 93.2% unknown / New malware mobile application.

**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, wors, 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, behavioural.
DISADVANTAGES:

✓ Identification of newly updated or created malicious application is hard to find out.
✓ 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 occurs.

PROPOSED SYSTEM:

In proposed paper, we implements SIGPID, Significant Permission Identification (SIGPID). The goal of the sigid 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 dataset and trained dataset. Support vector machine algorithms act as a classifier which is used to classify malicious application and benign app.

ADVANTAGES:

✓ 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 runtime performance by 85.6%.

SYSTEM ARCHITECTURE:
**Results**

First and foremost, we were able to implement much of what was discussed in the experimental setup phase. We were able to create the decision tree classifier, implement the Android application, and implement the classifier on the Android application. Below is a listing of the decision tree created, showing the learning process, and the final classifier’s nodes and their relative importance in the tree structure. Along with that is an example of the function that was exported to the Android application. While the function was automatically generated using Python, at this time it still needs to be put into the application manually through Android Studio. Though this is mainly due to the difficulty of implementing a decision tree regressor in Java, future iterations of our application will possibly allow a decision tree model to be read in through a file or through other means.

We collected data comparing the true and false positive rate of our classifier, shown below.
Conclusion and Future Work:

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 help users 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 malicious behavior 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.

References


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

