International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 12 | Dec-2017 www.irjet.net p-ISSN: 2395-0072

PERMISSION DRIVEN MALWARE DETECTION USING MACHINE LEARNING

Sanchit Gupta

Department of Computer Science and Engineering, SRM University, Chennai 603203, India

---

Abstract – *Mobiles are ubiquitous in today's world of technology. Android is very popular platform among these. This pervasiveness of Android makes it vulnerable to a number of attacks. The primary reason for this is the negligence and lack of knowledge while installing them. A number of malwares root up from the various permissions used by applications. What makes Android even more prone to these malware attacks is their ability to download and install apps from various third party sources too. Same application with same name might be available on various App Stores, this is taken as an advantage by the attackers to pack these apps with viruses and to attack the users. This paper's focus will be to study the correlation between the various permissions of Android applications and to find out an appropriate Machine Learning algorithm which has higher accuracy rates. Various classification algorithms are analyzed for the best performance and is used for Malware detection to help users be cautious and informed beforehand. There are 398 applications data collected with each having 331 permissions data to be analyzed [1]. This huge dataset enable for a better prediction and decision making.*

Keywords: Android, permissions, classification, machine learning algorithms, Malware detection.

I. INTRODUCTION

The android market is booming platform in mobile market, the market growth rate is increasing gradually and it is now at 84.7% [2].

Android platform is available on mobile phones, tablets, Smart TVs, etc. While, it may be advantageous to have this wide spread availability of applications, it can lead to security issues. For example, iOS users are allowed to download and install only from the authorized official App Store of Apple. On the other hand, android permits and can run many applications downloaded from a gamut of source, like Direct link downloads, Unofficial App Stores, Torrents, Google play store, etc. These varied sources make it easy to distribute malignant applications. The original applications are repacked with the malicious payload. These infected applications run some malicious code. They may try to seek access to many features and applications such as contacts book, GPS coordinates, messages, personal information gallery, etc. Moreover, they may also have monetized advertisements.

Even though a number of security steps have been taken, it's still not possible to eradicate these problems until and unless users are vigilant and more knowledgeable. A significant number of these risks and threats can be minimized by judging the various permissions an application is seeking for. Any application which is seeking approval of permissions which a user thinks, might not at all be needed or if the application due to some reasons doesn't look like an authentic application must not be installed. Various machine learning algorithms are compared viz., Naïve Bayes, Ada Boost, Multi Class Classifier, Random Tree, Random Forest and J48. To perform this classification process, a regimented process is followed. Firstly, the android manifest file is extracted. Secondly, the information about permissions an application requires is drawn out. The various permissions of the entire set of applications is considered. Thirdly, a database of all the permissions and the respective category of the application i.e. whether the application is safe (0) or a malware (1) is entered. Then a number of machine learning algorithms are used to study the pattern and to identify the applications are malware or not based on the set of permissions it seeks.

II. FRAMEWORK

A number of applications are evaluated using various Machine Learning techniques. Also, results are observed for a split of 75% training data and 25% remaining as the test data. There are many permission attributed considered in the evaluation.

A) Malware – Mobile malware is a malicious coded software that is custom made to attack mobile phones or smartphones. They try to exploit the main operating framework i.e. the operating system (OS). These kind of attacks are increasing and is a major concern for the android users rather than the iOS users. The main motive of these coded malwares is to extract personal sensitive information and data of the users. This data might include login credentials, credit card information, etc.

Plenty of detection techniques have been proposed for android. They are broadly divided into two categories, static and dynamic according to the execution requirements.

B) Static Analysis – It's the conventional method for computers, most likely for smart phones. The static analysis put forth by Schmidt et al. [3] can be executed on mobiles. Static analysis's main focus is to analyse the application before the execution to judge if the application is harmful or not [4]. This way of analysis involves a lot of storage spaces for the opcodes, graphs, etc. Also, to use static analysis, executable file has to be unpacked and decoded accordingly.

C) Dynamic Analysis – This way of analysis involves the analysis of the application by its execution on a virtual machine or real device processor. This is used for the study of application's characteristics and its behaviour. The code is tracked in and foraged for during the execution. It is more convenient to apply the dynamic analysis than the static analysis. This particular paper does detection of malware in android using the manifest permissions file that aid in the dynamic detection of the malware during installation.

III. ANDROID APPLICATION PACKAGE

The basic format of an android application is .apk format. This apk file has within itself an Androidmanifest.xml file. We need to use this file for permissions extraction. The following is a snapshot of the planned process in brief.

The following figure is an instance of Androidmanifest.XML [5] file.

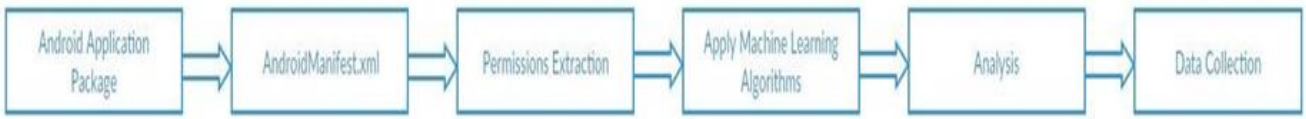
```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission android:name="com.android.vending.CHECK_LICENSE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.READ_LOGS" />
```

Figure Permissions Extraction

A. Feature extraction

For the analysis to be efficient, the features which are not useful or are same for all the applications can be remove to reduce the size of the dataset. So the first step is to extract the permission data set of all the apps under consideration. In the dataset sheet, an attribute "type" is used to declare if the application is malware-1 or not-0 [1].

The order for analysis to be done is, collect data of android apps both malware and genuine. Then prepare an excel sheet to record the data. If required, convert the Numeric type to Nominal type for analysis. Convert this file to Weka ARFF format or CSV format. After removing the unnecessary feature attributes, only 93 attributes for each application were to be considered. This increased the efficiency of the algorithms.



IV. VARIOUS TERMINOLOGIES AND EXPERIMENTAL OBSERVATION

After all unnecessary features are removed from the dataset, we start the analysis. Some terms to be familiar with are discussed here for better understanding. For analysis, a confusion matrix is used. Confusion matrix is also called as coincidence matrix. This confusion matrix is a table in which the predicted and the actual values are populated.

Accuracy (ACC): It is the ratio of number of correctly classified instances to the total number of examined instances.

True Positive Rate (TPR): Also called as recall. It is the ratio of positive instances correctly classified to the total number of actual positive instances.

False Positive Rate (FPR): It is the ratio of negative instances incorrectly classified as positive to the total number of negative instances.

True Negative Rate (TNR): It is the ratio of negative instances correctly classified to the total number of actual negative instances.

False Negative Rate (FNR): It is the ratio of positive instances incorrectly classified as negative to the total number of positive instances.

Precision (P): It is the ratio of predicted positive instances that were correctly predicted to the total number of false positive and true positive rate.

Table: Weighted average values of Type 0 and 1 (benign and malware) for various algorithms for class

Algorithm	TPR	FPR	Precision	Accuracy	Time Taken
Naïve Bayes	0.915	0.085	0.916	91.4573%	0.02 s
Ada Boost	0.917	0.083	0.917	91.7085%	0.12 s
Multi Class Classifier	0.887	0.113	0.887	88.6935%	0.15 s
Random Tree	0.920	0.080	0.920	91.9598%	**0.01 s**
Random Forest	0.932	0.068	0.932	**93.2161%**	0.3 s
J48	0.927	0.073	0.927	92.7136%	0.07 s

The respective confusion matrices for each of the algorithms is shown below.

Class 0 represents – Benign Application
Class 1 represents – Malware Application

Confusion Matrix by the number of applications on record are recorder and are as follows:

Naïve Bayes:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 187  12 |   a = 0
  22 177 |   b = 1
```

Ada Boost:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 176  23 |   a = 0
  10 189 |   b = 1
```

Multi Class Classifier:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 177  22 |   a = 0
  23 176 |   b = 1
```

Random Tree:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 183  16 |   a = 0
  16 183 |   b = 1
```

Random Forest:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 186  13 |   a = 0
  14 185 |   b = 1
```

J48:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 182  17 |   a = 0
  12 187 |   b = 1
```

V. CONCLUSION AND FUTURE SCOPE

In this paper permission-based analysis was done for various android applications with each application having many number of attributes. Different machine learning algorithms were performed such as.

In this paper different machine learning algorithms are used such as Naïve Bayes, Ada Boost,

Multi Class Classifier, Random Tree, Random Forest and J48. A number of applications are evaluated to detect whether the application is infected with malware or not. Totally 398 applications dataset was collected with each having 331 attributes. To clean the data, many attributes which weren't different or significant in the analysis were removed from the study. So based on this, it was found that 93 attributes are useful for the study. From the study, it was found that Random Forest has performed better than other classifiers. So it can be used to find out if an application has malware or not effectively. The performance of classifiers can be increased greatly by feature reduction.

New effective feature reduction techniques must be researched to increase efficiency. Also, some other better classifier algorithms must be researched.

REFERENCES

[1] Urcuqui, C., & Navarro, A. "Machine learning classifiers for android malware analysis", Communications and Computing (COLCOM), 2016 IEEE Colombian Conference (pp. 1-6).

[2] Madihah Mohd Saudi and Zul Hilmi Abdullah, "An Efficient Framework to Build Up Malware Dataset," International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 7, pp. 1104-1109, 0ct 7 2013.

[3]A.-D. Schmidt et al, "Static analysis of executables for collaborative malware detection on android," IEEE Conference on Communications, 2009.

[4] Prajakta D. Sawle and Prof. A. B. Gadicha, "Analysis of Malware Detection Techniques in Android," International Journal of Computer Science and Mobile Computing, vol. 3, pp. 176-182, March 2014.

[5] Mohsen Damshenas, Ali Dehghantanha, and Ramlan Mahmoud, "A survey on malware propagation, analysis and detection," International Journal of Cyber-Security and Digital Forensics (IJCSDF), pp. 10-29, 2013.