INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad-500 043

Project Based Learning

(Prototype / Design Building)

External Evaluation Report

Title of your Idea : **HOW ANTIVIRUS SOFTWARE IS WRITTEN ?**
Thrust Area / Sector : **MACHINE LEARNING**
Branch : **COMPUTER SCIENCE AND ENGINEERING**
Year / Semester : **IIIrd year**

Team Members (Max. 4) :

S. No	Name of the Student	Roll Number	Mobile Number	Signature
1	**KUKKADAPU LOHITH**	19951A0581	7680860954	
2	**SHINGADI LOKESH**	19951A0582	8247754115	
3	**THOTA NANDITHA**	19951A0584	9059912703	
4	**GALIPELLI MEGHANA**	19951A0591	9390448637	

1. Background of the Idea :

The evolution of mobile devices and mobile applications has drastically changed our ways of engaging in our everyday lives. However, despite bringing convenience to users, the mobile device is also facing malware invasion and attacks due to online social networks and online services. Mobile malware can trick as a standard code, then modify any intended application to corrupt and interfere with the system's functionality.

Machine learning predicts the data through computational learning theories and learns from experience or historical data. It has supervised and unsupervised classifiers that are used to analyze the features and trace the model. The approach provided by machine learning is to assist in the validation of normal and malicious activities.

In addition, malware classification is also significant in the determination of machine learning detection. Classification of malware is a process of categorizing a collection of malwares into target items based on categories, families, and classes. The new and favored classifier in machine learning is the Naïve Bayes classification model, which is used in static analysis methods. It uses the Bayes' Theorem and is a supervised machine-learning algorithm. Naïve Bayes calculates the probability of each event and predicts the class label to problem instances and presents the values of the features. It is a simple technique and produces a high degree of accuracy in detection.

This paper proposes the Bayesian probability method on permission-based features to accomplish the examination of signature-based malware. The permission-based features were extracted using static analysis that was classified using the Bayesian classifier to yield the results of the prediction of the malware. Specifically, this study applies malware detection to generate an outcome that allows users to recognize the threat in Android apps, thus preventing their mobile phones from being infected by the malware threat.

2. Problem Statement :

Since discovery of first Android malware in August 2010, new families have evolved in sophistication and are becoming increasingly difficult to detect by traditional signature-based antivirus. More recent families have been observed to exhibit polymorphic behavior, increased code obfuscation, encryption of malicious payloads, as well as stealthy command and control communication with remote servers. In fact, some Android malware like Anserver Bot is known to have the capability to fetch and execute payload at run time thus rendering its detection quite challenging. Security experts believe that difficulties in spotting malicious mobile applications results in most Android malware remaining unnoticed for up to 3 months before discovery. Furthermore, it is observed that it took on average 48 days for a signature-based antivirus engine to become capable of detecting new threats.

Limited Detection Techniques: There's more than one way to detect a virus, but one big disadvantage to some antivirus programs is that they may not employ all detection techniques. virus scanners are the most common method of detection. Scanning means searching your computer for known virus code patterns. There's an inherent weakness here. A scanner can either give a false alarm if a pattern code happens to match part of a normal file's code. Worse, it might not detect the code of a new virus at all. Installing and running antivirus software can use up a lot of computer memory and hard disk space, slowing down your PC. Premium antivirus software works absolutely fine when they are updated regularly this is because they update the virus definition database for more viruses to scan and detect. Premium software is costly.

In this paper we present a new approach - based on Bayesian classification - which utilizes certain characteristic features frequently observed with malware samples to classify an app as ‘suspicious’ or ‘benign’. Our approach is developed as a proactive method aimed at uncovering known families as well as unknown malware so as to reduce incidents of malware in marketplaces from evading detection. The Bayesian classification-based approach can complement signature-based scanning thus enabling harmless apps obtained from Android marketplaces to be verified whilst isolating suspicious samples for further scrutiny.

3. Proposed Solution :

Bayesian probability is used for malware detection, focusing on permission-based features. The application characteristic was retrieved from static analysis using extensive malware samples that contain a variety of benign and malware. The proposed system considers the most dominant permission that contributes to malware detection. Bayesian probability was selected as a method due to its sturdy mathematical base and stability of classification competency. Moreover, Bayes’ rule is one of the fundamental pillars of probability and the probability theory implemented in the computerized system. The equation of Bayes’ rule is as follows:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

According to Eq., A is the event for the probability, and B is the new evidence that is related to A in some way. The event to be estimated is called posterior P(A|B), which is the probability of getting malware when the phone gets the virus. When the probability is observed, the new evidence is P(B|A), which is also called likelihood, becomes the initial hypothesis. The prior is P(A); that is, the event that is not required is the additional information of prior. The theory is similar to the example that probability has malware. The marginal likelihood is P(B), which contains the total probability of observing the evidence. The Bayesian Theorem is used in various fields of classification studies. The simple technique in Naive Bayes that produces the detection, has the lowest error rate, assigns the class label.

4. Technology concept formulation:

A. The Classifier model

The Bayesian based classifier consists of learning and detection stages. The learning stage uses a training set in form of known malicious samples in the wild and benign Android applications, collectively called the app corpus. The Java-based package analyzer uses the detectors to extract the desired features from each app in the corpus. The feature set is subsequently reduced by a feature reduction function while the training function calculates the likely probability of each selected feature occurring in the malicious and benign applications. The training function also calculates the prior probability of each class i.e., suspicious and benign. The feature sets along with their likelihood probabilities are stored for use in the subsequent classification stage. New applications are assessed in the classification stage with respect to the features in the selected feature set.

B. Feature ranking and selection

Let an application characteristic ri obtained from the APK analyzer detectors be defined by a random variable:

$$R_i = \begin{cases} 1, \textit{if discovered by the detectors} \\ 0, \textit{otherwise} \end{cases}$$

Let C be a random variable representing the application class, suspicious or benign:

$$C \in \{suspicious, benign\}$$

Every application is assigned a vector defined by $\vec{r} = (r_1, r_2, \ldots r_n)$ with ri being the result of the i-th random variable Ri .
As the goal is to select the most relevant features, the feature reduction function calculates the Mutual Information of each random variable, thus:

$$MI(R_i, C) = \sum_{r \in \{0,1\}} \sum_{c \in \{sus, ben\}} P(Ri{=}r; C{=}c) . \frac{P(Ri{=}r; C{=}c)}{P(Ri{=}r) . P(C{=}c)}$$

After calculating the MI for each feature, the feature set is then ranked from largest to smallest in order to select those that maximize the MI between R and C thus enabling optimum classifier performance.

C. Bayesian classification

According to Bayes theorem, the probability of an application with the feature vector

$$\vec{r} = (r_1, r_2, \ldots r_n)$$

belonging in class C is defined by:

$$P(C=c \mid \bar{R}=\bar{r}) = \frac{P(C=c)\prod_{i=1}^{n} P(R_i=r_i \mid C=c)}{\sum_{j\in\{0,1\}} P(C=c_j)\prod_{i=1}^{n} P(R_i=r_i \mid C=c_j)}$$

D. Evaluation Measures

To evaluate the predictive accuracy of classifiers, several measures have been proposed in the literature. In the context of our problem the relevant measures utilized in our experiments are given below.

Let $n_{ben \rightarrow ben}$ be the number of benign applications correctly classified as benign, $n_{ben \rightarrow sus}$ the number of misclassified benign applications, $n_{sus \rightarrow sus}$ the number of suspicious applications correctly identified as suspicious while $n_{sus \rightarrow ben}$ represents the number of misclassified suspicious applications. Accuracy and Error Rate are respectively given by:

$$Acc = \frac{n_{ben \rightarrow ben} + n_{sus \rightarrow sus}}{n_{ben \rightarrow ben} + n_{ben \rightarrow sus} + n_{sus \rightarrow ben} + n_{sus \rightarrow sus}}$$

$$Err = \frac{n_{ben \rightarrow sus} + n_{sus \rightarrow ben}}{n_{ben \rightarrow ben} + n_{ben \rightarrow sus} + n_{sus \rightarrow ben} + n_{sus \rightarrow sus}}$$

We also define the false positive rate (FPR), false negative rate (FNR), true positive rate (TPR), true negative rate (TNR) and precision (p) as follows:

$$FPR = \frac{n_{ben \rightarrow sus}}{n_{ben \rightarrow sus} + n_{ben \rightarrow ben}}$$

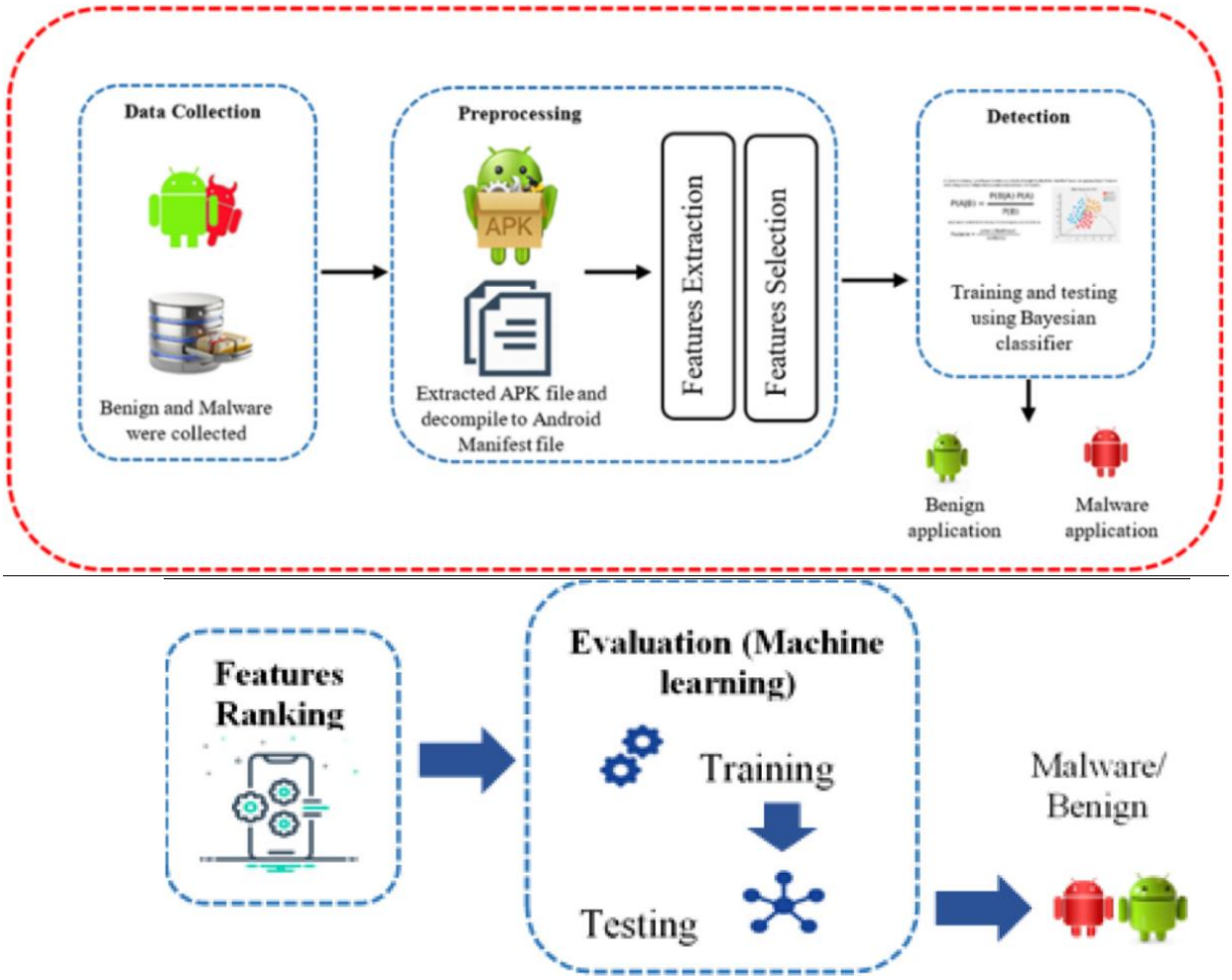
$$FNR = \frac{n_{sus \rightarrow ben}}{n_{sus \rightarrow sus} + n_{sus \rightarrow ben}}$$

$$TPR = \frac{n_{sus \rightarrow sus}}{n_{sus \rightarrow ben} + n_{sus \rightarrow sus}}$$

$$TNR = \frac{n_{ben \rightarrow ben}}{n_{ben \rightarrow sus} + n_{ben \rightarrow ben}}$$

$$p = \frac{n_{sus \rightarrow sus}}{n_{ben \rightarrow sus} + n_{sus \rightarrow sus}}$$

5. **Prototype of proposed system (UI screens / block diagrams / circuits / designs):**



6. **Detailed description of prototype / product / project :**

1. **Dataset collection phase**

This process includes decompiling the .apk file, extracting, and filtering permissions. The permissions are collected and saved as a .arff file in a readable format. The .arff file holds all the attributes of the functionality used to optimize functionality so that noise and irrelevance are excluded. Then, the downloaded datasets are extracted and compiled in a .csv file format. The samples were predefined manually, then labelled, such as benign or malware. The inspection of the Android application status in VirusTotal is completed with the labelling process of malware and benign. It is widely used by researchers in their fields. VirusTotal is an online website provided for checking viruses through URLs and uploaded files. The samples of malware were run through VirusTotal as the VirusTotal completed the validation. Additionally, benign datasets are screened in VirusTotal to determine if they are malicious or not. The dataset sample was then used for feature selection.

2. Pre-processing phase

APK is decompiled to parse AndroidManifest.xml containing permissions and compiles classes.dex with all Dalvik bytecodes, such as API calls through static analysis. The feature extraction needs different tools, such as Androguard and ApkTool, to gain AndroidManifest.xml and classes.dex files that comprise permission, system call, intent, and native code. The AndroidManifest file stores all the information of the Android application like permission. The tag of classifies the types of permission requested in the application. The APK file is decompressed and converted from the binary manifest file to readable XML

format. The Androguard tool is used to obtain permission tags. The sample of a dataset that contains permission features is labelled as benign or malware in the last column to be stored as .arff file in the database. These features served as attribute values and were termed as 'feature sets'. The binary value representing permission requested or not by that application was stated as '1' for permission requested by a specific application, while '0' for permission not requested by the specific application.

3.Detection phase

This study applied Bayesian probability in a machine learning approach to detect malicious codes in Android applications. Bayesian probability has excellent statistical properties that classify some malicious code as benign and some benign code as malicious, and can potentially detect unknown malicious code.

Implemented Bayesian models from different approaches

1.Permission-based Bayesian classifier

Permissions are the most recognizable security feature in Android. A user must accept them in order to install an application. Permissions have also been used in several of the Android tools mentioned in section 2, to provide app profiling information. Thus, their efficacy for machine learning based malware detection using trained models from large malware sample sets will be investigated. A permission is declared using the tag in the Manifest file. For example, in order for an application to read phone contacts it must declare the standard Android permission as follows:

<**uses-permission andriod:name="android.permission.READ_CONTACTS"**>
<**/uses-permission**>

The results indicated that permissions attributes would provide discriminative capabilities for training the classifier to distinguish between malware and benign applications. In order to evaluate the permissions-based model, we carried out experiments designed to determine: (a) How effective the permissions-based features extracted from analysis of our malware and benign sample sets are in detecting unknown malware. (b) How well the permission-based model performs compared to the other viable models e.g. trained models derived from code properties extracted as features.

2.Code-based properties Bayesian classifier

Unlike the permission-based model described above, the code-based model utilizes features extracted from code-based properties. A number of code-based properties were specified as matching criteria for a set of property detectors implemented within the apk analyzer. The detectors parse .smali files obtained from disassembled .dex files. In addition, external libraries, files within assets folders and resources folders are also scrutinized, if present within a decompressed APK.

3.Classifier based on combined ranked permissions and code-based properties

The third data mining approach that was implemented in the analyzer utilized a combination of permissions and code properties. The feature selection function was used to simultaneously rank the permissions and properties obtained from the code, using our 1000 benign and 1000 malware samples. The highest ranked from both were subsequently selected as input feature vectors for the Bayesian classifier model.

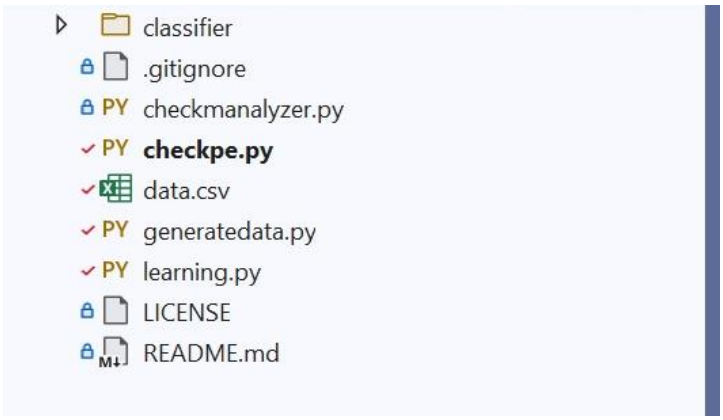
Feature extraction times comparison

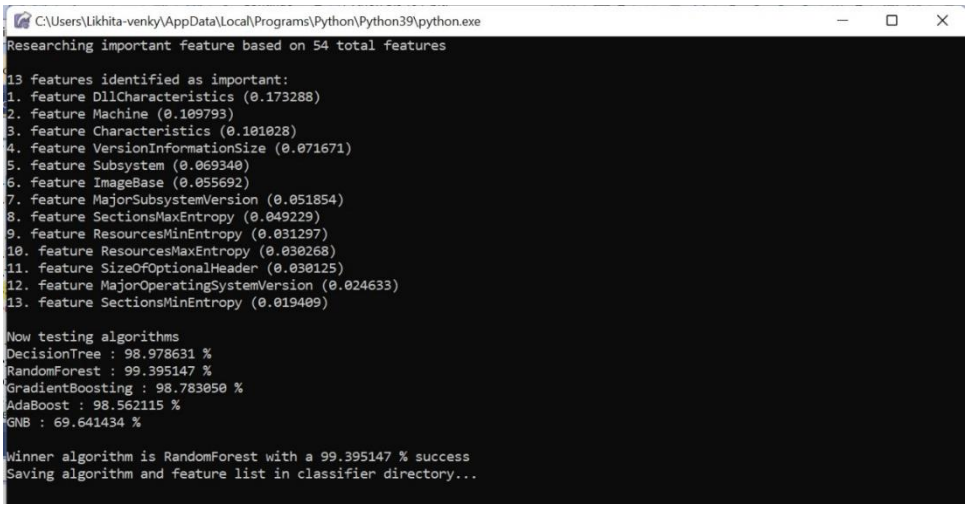
The ranking and selection of top relevant features for training the models will significantly reduce computational overhead during the classification of applications, since the lower ranked 'redundant features' will not be utilized. This can be deduced from the time taken by our APK analyzer to extract the properties and construct feature vectors for training each of the models.

7. Final version of prototype / product :

learning.py | checkmanalyzer.py | checkpe.py | generatedata.py

```
import pandas as pd
import numpy as np
import pickle
import sklearn.ensemble as ske
from joblib import dump
from pickle import dumps
from sklearn import tree, linear_model
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
from sklearn.externals import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```





8. Any other information:

Conclusion:

In this paper we proposed and evaluated a machine learning-based approach for detecting Android malware. In particular, a novel application of Bayesian classification is applied to this problem.

The results presented in the paper showed significantly better detection rates than were achieved by popular signature-based antivirus software tested previously on the same set of malware samples used in our experiments. The malware samples used in the experiments were from the largest publicly available collection at the time of writing. Future work could investigate the classifier performance with larger sample sets as more malware samples are discovered in the wild. Further studies could also investigate performance improvement via prior incorporation of expert knowledge within the Bayesian classification model.

Dean-CLET **Dean -TIIC**