Permission-Based Android Malware Detection

1.0 Introduction.

For each Android application, several selected features are retrieved from the corresponding application package (APK) file. In addition, the real permissions required by the applications ar identified.The values of selected features are stored as a binary number (0 or 1), which is represented as a sequence of comma separated values.All selected features are enumerated in following items. Each item includes the name of a feature, the data type of the feature, and data of the feature.

1.1 Getting Started.

Feature Selection:

In Machine Learning applications, a large number of extracted features, some of which redundant or irrelevant, present several problems such as-misleading the learning algorithm, over-fitting, reducing generality, and increasing model complexity and run-time.

The amount of features should be reduced while preserving a high level of accuracy. In this section we select the k best features from the extracted features of android application package by using feature selection method: Information Gain.

Gain (S, A)= Entropy (S) - Σ |SV| Entropy(SV) V€Values(A) |S|

1.2 Pre-requisites:

The list of softwares that have been used in the Module-02 as prerequisites are,
1. PyCharm - For executing python codes.
2. Notepad++
3. Windows Command Prompt
4. Scikit-Learn

The list of Folder and Sources in the zipped folder,

1. Python Source Files
2. Screenshots-Test Cases
3. Readme.pdf
4. Extracted features - datasetandroidpermissions.zip
5. Permission-based-Android-Malware-Detection.pdf

1.2 Machine Learning and Malware Detection.

The selected features are collected into the signature database and divided into training data and test data and used by standard machine learning techniques to detect the android malware applications. K-means clustering is chosen,
(i) it is data driven method relatively few assumptions on the distributions of the underlying data
(ii) it guarantees at least a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets.

1.3 K-Means Algorithm

1. Select k centroids arbitrarily (called as seed) for each cluster Ci , i ε [1, k]
2. Assign each data point to the cluster whose centroid is closest to the data point.
3. Calculate the centroid Ci of cluster Ci, i ε [1, k].
4. Repeat steps 2 and 3 until no points change between clusters.