SECURITY RISK ANALYSIS OF GOOGLE PLAY APPLICATIONS

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Jasmeet Kour

August 2013

SECURITY RISK ANALYSIS OF GOOGLE PLAY APPLICATIONS

Jasmeet Kour

APPROVED:

Dr. Weidong Shi, Chairman
Dept. of Computer Science

Dr. Ricardo Vilalta
Dept. of Computer Science

Dr. Bogdan Carbunar
School of Computing and Information Sciences
Florida International University

Dean, College of Natural Sciences and Mathematics

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Weidong Shi for providing an excellent environment for conducting research work. It was an absolute privilege to work under his tutelage, inspiring me to explore ideas for prospective research topics, obtaining constructive feedback on my performance, providing me the opportunity and the flexibility to explore the research topic, and also allowing me to interact, gather, and share ideas with fellow team members.

I would also like to thank Dr. Ricardo Vilalta. Without his support and guidance, I would not have made much progress in my research work. I would like to extend a special word of thanks to Dainis Boumber and Yifei Jiang for their help in collecting necessary data to test the ideas presented in this thesis and also providing suggestions in choosing the right tools for testing framework.

A word of thanks to my family and friends, especially Tejinder Singh and Harsheen Kaur, for inspiring confidence and determination in me to never give up and strive to achieve the best in my endeavors.

Lastly, I would like to thank the Department of Computer Science for providing me the opportunity to pursue research in this distinguished university.

SECURITY RISK ANALYSIS OF GOOGLE PLAY APPLICATIONS

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Jasmeet Kour

August 2013

Abstract

In this work we determine the security risk posed by Google Play applications to mobile device users by using statistical and machine learning techniques. A Google Play application is characterized by its description, a category, and a set of security permissions required to perform its described functions. Typically mobile users install applications based on their descriptions and don't necessarily analyze the permission settings. Therefore, an application is defined as unsafe or risky when some permissions are accessed which provide potentially sensitive and confidential information and not justified by its description. We use Stanford Topic Modeling Toolbox (TMT) to perform topic model learning on a given dataset and then perform inferencing on a new corpus. The testing from the training data set shows that the results obtained from using clustering are better than the classification approach results. The results also indicate that it is possible to predict with high confidence the security risk of an application based on its permission settings and description.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid growth of smart phones has lead to a renaissance for mobile and tablet services. Application markets such as Apple's Application Store and Google's Google Play Store provide point-and-click access to hundreds of thousands of paid and free applications. The fluidity of the markets also presents enormous security challenges. Therefore, the ability to predict the security risk of an application is very critical before safely using it. The security risk of an application is mainly a function of its categorization and permission settings. A very promising tool to attain this objective is the use of machine learning techniques. Machine learning algorithms are techniques that automatically build models describing the structure at the heart of a set of data. Ideally, such models can be used to predict properties of future data points and people can use them to analyze the domain from which the data originates. In fact, one of the most useful machine learning techniques in predictive learning is classification. Classification is a predictive machine learning technique, makes predictions about

the values of data using known results found from different data [1]. We also use the machine learning technique of clustering in our analysis. The objective of clustering is to divide a set of objects into clusters so that the objects in the same cluster are similar to each other, and/or objects in different clusters are dissimilar. Stanford Topic Modeling Toolbox (TMT) is used to cluster different applications that belong to the same category based on their text descriptions. After TMT is trained on the sample data, inferencing is performed to find the safety of an application.

The main contribution of this work is the methodology to determine whether an application is safe or unsafe. We define applications safe or unsafe when they ask for permissions not essential to their advertised functions. The classification approach is not very successful in determining the safety risk of an application due to our inability to define whether an application as safe or unsafe correctly as an input. We also make certain assumptions in our analysis. We perform the classification or clustering techniques assuming that the applications belong to the same category and the process is repeated for different categories. The challenge in the analysis is to train the model for unsafe behavior, as determining unsafe risks is the output of the analysis. For our analysis, a data set of more than 3000 applications is obtained from Google Play Store belonging to different categories. We train and validate the model using cross-validation and percentage split techniques for classification techniques. For clustering technique, we use a reference data set for test results. Our testing from the training data set shows that the results obtained from using clustering are better than the classification approach results.

The thesis is organized as follows. Section 2 gives background information on

the reasons behind launching this study and the state of the art in the security of applications. It also provides information on tools like WEKA and TMT used in the analysis. Section 3 develops the methodology used in this work and describes the procedure to find the probability whether an application is safe or unsafe. Section 4 details the results obtained from the methodology described in section 4 and finally, section 5 concludes the paper and outlines directions for future work.

Chapter 2

Background

It is estimated that 1.2 billion people worldwide were using mobile applications at the end of 2012 [1]. This is forecasted to grow by 29.8 percent each year, to reach 4.4 billion users by the end of 2017 [1]. Also, analysts estimate that 56 billion smart phone applications will be downloaded in 2013 and by operating system; 58 percent of such downloaded applications will be on Google Android [1]. People who create viruses and other malicious software, or malware, for mobile devices have targeted Android because it has become the dominant mobile operating system worldwide. Unlike the approach of Apple's Application Store, Android software development and the Google Play Market are relatively open and unrestricted. This offers both developers and users more flexibility and freedom, but also creates significant security challenges. Although Google cracks down on malware found in the Play store, other websites and stores are less likely to scan for malicious software. It has also been reported by some studies that 25 percent of Google Applications pose a security

threat [2]. The security risk for applications is defined as being able to make unauthorized payments, steal data, modify user settings, etc. The current Android development and usage model allows developers to upload arbitrary applications to the Google Play Store and involves the end-user in granting permissions to applications at install-time. This, however, opens attack opportunities for malicious applications to be installed on users' devices (see DroidDream Trojan [3]). Since its introduction, a variety of attacks have been reported on Android showing the deficiencies of its security framework. In the literature, Enck et al. introduced TaintDroid [4] to track privacy-related information flows to discover such malicious applications. TaintDroid provides real-time analysis by leveraging Android's virtualized execution environment, compared to the static analysis of our work based on the configuration settings of the applications. Furthermore, Enck et al. analyzed 1,100 Android applications for malicious activity and detected widespread use of privacy-related information for tracking. However, no other malicious activities were found, and in particular, no exploitable vulnerabilities that could have lead to malicious control of a smart phone were observed [5]. Bugiel et al. [6] investigate the problem of designing and implementing a practical security framework for Android to protect against confused deputy and collusion attacks. Burns [7] discusses developing secure mobile application for Android. In this work, instead of focusing on the real-time tracking of security issues and the security framework itself, we investigate the security of the applications based on its intended functions and security permissions. To the best of our knowledge, there is no in-depth study of the security of Google Play Store applications to date based on its permission settings, though some analysis on the

riskiness of applications has been done by a few security firms. We provide some background on the tools TMT and WEKA that have been used in this work.

2.1 WEKA

WEKA is abbreviation of Waikato Environment for Knowledge Analysis. is a popular suite of machine learning software written in Java, developed at the University of Waikato. WEKA is free software available under the GNU General Public License [8]. The WEKA workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. The original non-Java version of WEKA was a TCL/TK front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and a Makefile-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (WEKA 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. The main strengths of WEKA are that it is freely available under GNU (General Public License), very portable because it uses Java Programming Language that can run in almost all modern platforms, contains a large number of data preprocessing and modeling technique, and is easy to use by beginners with its easy to use graphical user interface. WEKA supports several standard machine learning tasks, more specifically, data preprocessing, clustering,

classification, regression, visualization, and feature selection. We use WEKA mainly for pre-processing and classification of the applications. Predictive models have the specific aim of allowing us to predict the unknown values of variables of interest given known values of other variables. Predictive modeling can be thought of as learning a mapping from an input set of vector measurements to a scalar output [9]. Classification is a predictive machine learning technique. Prediction models that include description, category, user rating, and security permissions are necessitated for the effective prediction of the safety risk of an application. The prediction of safety risk of an application with high accuracy is beneficial to identify the applications that can access confidential information without the knowledge of the end user. We use WEKA tool to perform the classification of applications. Different classification techniques like logistic regression, decision trees, neural networks, etc. are used in this work. All classification techniques develop relationship between a categorical dependent variable and one or more independent variables. Therefore, the main challenge in using classification techniques for finding the nature of an application is the pre-determination of the outcome variable for the training data set. WEKA provides different testing strategies to validate the model. We can use cross-validation, percentage split, or separate data sets other than the training data set for validating the model.

2.1.1 Logistic Regression

Logistic regression is a form of regression analysis used for predicting when the output variable is discrete rather than continuous. The theoretical foundation of the

method is attractive. It is in line with the generalized regression. Thus the logistic regression is a well identified variant which one can implement according the kind of the dependent variable. Their performance in prediction is comparable to the other approaches. Furthermore, it puts forward some indicators for the interpretation of the results. Among them, the famous odds ratio enables one to identify precisely the contribution of each predictor. WEKA also provides the coefficients of the predictor variables and the confusion matrix as part of the results.

2.1.2 J48 Decision Tree

In WEKA, decision trees are given the intuitive name "j48"; you can find this algorithm by opening the trees subfolder under the classifiers button. A decision tree is a decision modeling tool that graphically displays the classification process of a given input for given output class labels. Decision trees are potentially powerful predictors and embody an explicit representation of the structure in a dataset. Their accuracy and comprehensibility depends on how concisely the learning algorithm can summarize this structure. To use J48 in WEKA, the following steps are done. In Preprocess: load a dataset and look at it, use the Data Set Editor, and apply a filter (to remove attributes and instances). In Classify: load a dataset and classify it with the J48 decision tree learner (test on training set), examine the tree in the Classifier output panel, interpret classification accuracy and confusion matrix, test the classifier on a supplied test set, and visualize classifier errors.

2.1.3 Multilayer Perceptron

Artificial Neural Networks (ANNs) denote a set of connectionist models inspired in the behavior of the human brain. In particular, the Multilayer Perceptron (MLP) is the most popular ANN architecture, where neurons are grouped in layers and only forward connections exist [10]. This provides a powerful base-learner, with advantages such as non-linear mapping and noise tolerance, increasingly used in machine learning due to its good behavior in terms of predictive knowledge [11]. Human brain is a densely interconnected network of approximately 1011 neurons, each connected to, on average, 104 others. Neuron activity is excited or inhibited through connections to other neurons. The fastest neuron switching times are known to be on the order of 10^{-3} sec. It is clear that human brain is beyond amazing about how fast each neuron connected with each other with their speed, 10^{-3} sec. So what is the connection between human brain and ANN? A multilayer perceptron is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output. It is a modification of the standard linear perceptron in that it uses three or more layers of neurons (nodes) with nonlinear activation functions, and is more powerful than the perceptron in that it can distinguish data that are not linearly separable, or separable by a hyper-plane.

2.2 Stanford TMT

The Stanford Topic Modeling Toolbox (TMT) brings topic modeling tools to social scientists and others who wish to perform analysis on datasets that have a substantial

textual component. TMT was written at the Stanford NLP group, first released in September 2009. The toolbox features the ability to:

- Import and manipulate text from cells in Excel and other spreadsheets.
- Train topic models to create summaries of the text.
- Select parameters (such as the number of topics) via a data-driven process.
- Generate rich Excel-compatible outputs for tracking word usage across topics.

Topic models can be useful for extracting patterns in meaningful word use, but they are not good at determining which words are meaningful. It is often the case that the use of very common words like 'the' do not indicate the type of similarity between documents in which one is interested. To lead Latent Dirichlet Allocation (LDA) towards extracting patterns among meaningful words, TMT implements a collection of standard heuristics. In this work, we use TMT to group applications belonging to a given category by using LDA model. The test scripts are written in Scala language. Scala is a general purpose programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It smoothly integrates features of object-oriented and functional languages, enabling Java and other programmers to be more productive. Code sizes are typically reduced by a factor of two to three when compared to an equivalent Java application.

Chapter 3

Methodology

In this section, we define two solution approaches, one based on classification and other on clustering. Classification analysis is performed using WEKA, whereas TMT toolbox is used for clustering method. We also provide a formal definition of the application security problem.

3.1 Problem Definition

A category defines the area of function to which an applications belongs to e.g. education, communication, productivity, etc. A is the set of applications (training set) such that each application $x \in A$ has a category $c_x \in C$ where C is the set of categories. P is the set of the permissions any application is allowed to have where $|P| = L$. P_x is the set of values of permissions of application x such that the value $p_{x,i} \in P_x$ corresponds to the permission $p_i \in P$ where $1 \leq i \leq L$. Each permission

value $p_{x,i}$ is a binary variable; $p_{x,i} = 1$ means the permission p_i is allowed for the application x otherwise not when $p_{x,i} = 0$. D_x is the set of keywords describing the functionality of the application x. S_x is the probability that application x is safe. Therefore, $S_x = 1$ means the application x is definitely safe otherwise it is unsafe to install and use when $S_x = 0$. For values of S_x between 0 and 1, we define a criterion in section 3.3 whether the application x is safe or not.

3.2 Classification

The objective of classification techniques is to group objects into predetermined classes. It is an example of supervised learning where mapping of inputs to desired outputs is required. In our case, the two classes are safe and unsafe applications. Therefore, the main challenge in using classification techniques is to determine unsafe applications in the training set. In our analysis, we first assume that all the applications that belong to the same category of the application for which we are determining the security risk are safe and all the applications that belong to other categories are unsafe in the training set. Let y be the app for which security risk needs to be determined and $c_y \in C$ is its category. Therefore, for each application x that belongs to training set A, S_x is defined below:

$$S_x = \begin{cases} 1 & \text{if } c_x = c_y \\ 0 & \text{otherwise} \end{cases} \quad x \in A \tag{3.1}$$

This approach of defining the safety of an application for the training set has its limitations. It may define two different applications with same permission settings as

safe and unsafe and the classification technique won't be effective for the training set. Therefore, it is important to prune the training data set A based on the category C_y of application y. We define a procedure to develop a training set A_y for application y to determine its safety decision variable S_y. The procedure to find safety risk of an application is based on the permission settings. The applications that have permission settings mostly different from the permissions settings of all applications belong to a given category are defined as unsafe applications in the procedure with respect to the given category. Such applications obtained from the procedure constitute the training set A_y for the category C_y.

3.3 Grouping

Grouping of application belonging to the same category is done using Stanford TMT toolbox. It is an example of unsupervised learning model, as no mapping of inputs to output labels is required. It is possible that applications within the same category may have completely different permission settings. Therefore, for more accurate results, the grouping is not only done based on the category of an application, but also using the description of an application. The description defines the intended functions of an application and its permission settings reflect that. In other words, we assume that description is the only independent variable that influences the grouping of an application with other applications in the same category. Based on these assumptions, we define a probability model to determine the safety risk of an application. Let G_c defines a set of groups belonging to the category $c \in C$. Each

group $g \in G_c$ has two attributes, a topic T_g and a set of applications S_g of apps that belong to the group. Topic T_g of group $g \in G_c$ is nothing but a set of keywords representing all the applications that belong to the group. Since all the applications that belong to a single group have similar functionalities due to matching keywords, they should have similar permission settings. Let U_g be a set of permissions obtained from the union of the permission settings of all the applications S_g belonging to the group g, such that $u_{g,i} \in U_g$ is the value of permission setting for permission $p_i \in P$, $1 \leq i \leq L$. In the model, we also define two different weights for each permission $p \in P$. Weights $w_{c,p,1} > 0$ and $w_{c,p,2} > 0$ are assigned to each permission $p \in P$ for category $c \in C$. Weight $w_{c,p,1}$ represents how widespread is the use of the permission p in the applications belonging to the category c. The lower the value of weight $w_{c,p,1}$, the higher is the frequency of the applications using the permission p. Weight $w_{c,p,2}$ provides the level of risk associated with using the permission p independent of its popularity among applications belonging to category c. The larger the value of $w_{c,p,2}$, the greater the security risk posed by the application that is using the permission p. The output of Stanford TMT toolbox from the training data set is a probability distribution for each application across the pre-determined number of groups. The output also provides keywords associated with each group. Since the output is a probability distribution, it is possible that an application may not clearly belong to one group of applications. Let n is the number of groups, a priori, to be formed from the training data set. We train the model separately for a given category $c \in C$. Let $A_c \subseteq A$ be the training data set and $B_c \subseteq A$ be the inferencing dataset of applications for which we needs to determine their safety risk for the category $c \in C$.

Assume that $q_{x,i}$ is the probability that application $x \in A_c$ belongs to group g_i where $1 \leq i \leq n$. We define a cut-off probability $\alpha > 0$ such that when $q_{x,i} < \alpha$, application $x \in A_c$ is not considered for the group g_i. $S_{y,g}$ is the probability that application $y \in B_c$ is safe when it belongs to group g as per the TMT output and is given below:

$$S_{y,g} = \frac{\sum\limits_{\substack{i=1 \\ p_{y,i}=1, u_{g,i}=1}}^{L} w_{c,i,1}}{\sum\limits_{\substack{i=1 \\ p_{y,i}=1, u_{g,i}=1}}^{L} w_{c,i,1} + \sum\limits_{\substack{i=1 \\ p_{y,i}=1, u_{g,i}=0}}^{L} w_{c,i,2}} \tag{3.2}$$

From the empirical results, it rarely happens that an application belongs to only one group. Therefore, it is important to consider the safety of an application based on all the groups that an application may belong to. Let $M_y \subseteq G_c$ is the set of groups to which application y is part of based on the inferencing results for the data set B_c such that $|M_y| = m$. We define $d_{y,k}$ as the probability that application y belongs to group $g_{y,k} \in M_y$ where $1 \leq k \leq m$. The probability S_y that application y is safe is determined as:

$$S_y = \sum_{k=1}^{m} \left(\frac{d_{y,k}}{\sum\limits_{i=1}^{m} d_{y,i}} \right) S_{y,g_{y,k}} \tag{3.3}$$

S_y is calculated as an expected value of a random variable for a discrete probability distribution. The random variable is the probability that an application is safe when it belongs to a given group. The final decision variable of the model is whether an application is safe or risky to use. We again define a cut-off probability $\beta > 0$ such

that when $S_y \geq \beta$, the application y is considered safe ($S_y = 1$) otherwise risky ($S_y = 0$). This parameter β can be optimized from the training data set A_c for category $c \in C$. We assume that we know beforehand whether an application x that belongs to training data set A_c is safe or not such that $W_x = 1$ means application x is safe otherwise not when $W_x = 0$. β can be found by optimizing the following problem:

$$\begin{aligned} &\underset{\beta}{\text{minimize}} \sum_{x=1}^{|A_c|} |R_x - W_x| \\ &\text{subject to } R_x = \begin{cases} 1 & \text{if } S_x \geq \beta, \;\; x = 1, \ldots, |A_c| \\ 0 & \text{otherwise} \end{cases} \\ &0 < \beta \leq 1 \end{aligned} \tag{3.4}$$

Note that the parameter β is calculated from the training data set A_c and not the inferencing data set B_c. It is obvious that before we start inferencing, β should be known as the final decision variable whether the application $y \in B_c$ is safe or not is dependent on it.

Chapter 4

Results

In this section we provide the results of the methodologies discussed in section 3 to find out the safety risk of an application. The results are obtained by using a training data set of around 3200 applications downloaded from the Google Play Store. Each Google Play application has 6 attributes which are "Name", "Category", "Description", "Rating", "NumRating" and "Permission settings". Each permission setting has a unique name and the maximum number of such permissions for an application can be 121. Each permission setting is a binary variable and can take a value of either 0 or 1. A value of 1 means that the permission setting is used by the application and 0 implies the permission is not used. The average number of permissions used by applications in the training data set is 3.5, i.e. on an average 2.9% of permissions are accessed by applications. The top 3 permissions in the training data set "full internet access ", "view network state", and "modify delete usb storage contents modify delete sd card contents" are accessed by 70%, 47%, and 31% of the applications respectively.

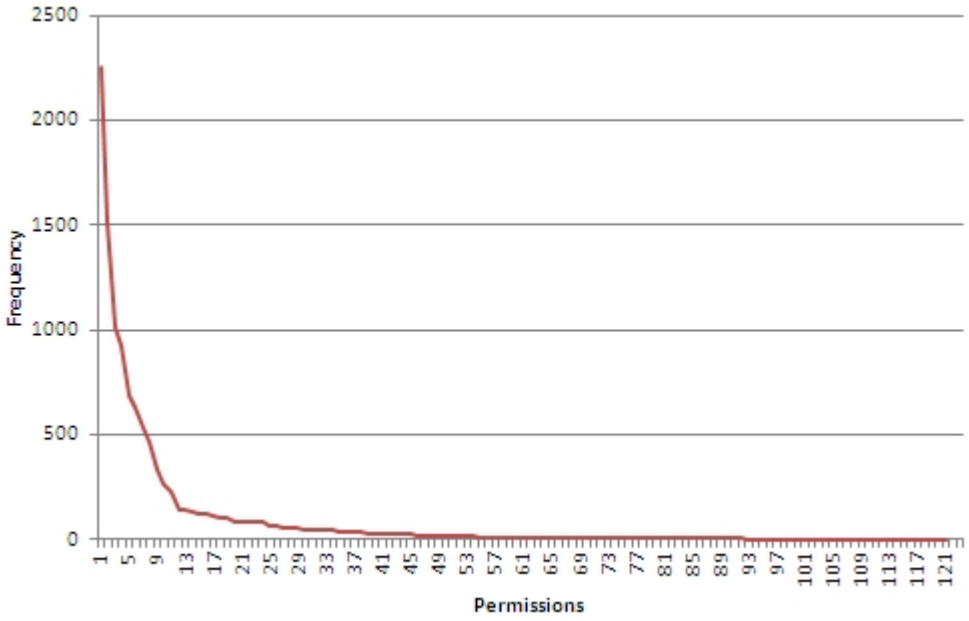

Figure 4.1: Permissions Frequency

The average number of permissions accessed by each application is 3.5 permissions. Figure 4.1 shows that the frequency of each permission used is exponentially decreasing; only 30 permissions out of 121 are accessed at least 50 times by the applications. "Rating" attribute takes a value between 0 and 5 and represents the popularity of the application. The higher the value of the "Rating" attribute, the higher is the popularity of the application. "UserRating" is the number of times the application is downloaded by the users and also an indicator of the popularity of the application. We don't use these two attributes in deciding the safety of an application, as we don't have a correlation between these two attributes to the safety of an

application in the training data set. Many mobile users provide a higher rating if they like the application without finding whether the application is harmful. "Category", "Description", and "Permission Settings" are the critical attributes in determining the application safety. In this research work, we define that application as safe which has permission settings representing its intended functions. The intended functions of an application are defined by its category and description. Therefore, if the permissions are not reflecting the intended functions, we categorize that application as unsafe. The training set consists of 30 different categories e.g. photography, finance, sports, etc., and an average of 108 applications per category. Results obtained from classification and grouping techniques are very encouraging and can be helpful in determining the safety risk of an application.

4.1 Classification Results

WEKA is used to perform different classification methods for determining the safety risk of Google Play applications. The different techniques used are logistic regression, decision-trees, and neural networks. Each classification technique is tested on two subsets of training data. The first subset is randomly obtained from the parent data set for a given category. For the second data sub-set, we use the filtering process defined in section 3.2 to train the model for unsafe applications. Also, for each classification method used, we use two validation approaches, cross-validation and percentage split. For cross-validation, we use 10 folds testing and for percentage-split, 66% of the training data set is used for model learning and rest for testing

the model. Cross-validation gives better results than split percentage testing mainly due to more data-points used for testing the model. In our case, we have a two-class prediction problem (binary classification), in which the outcomes are labeled either as safe (p) or unsafe (n). The total safe and unsafe instances are P and N. There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p. The measures we use to determine the success rate of the classification techniques are Accuracy, True Positive Rate (TPR), and False Discovery Rate (FDR). Accuracy is determined as the percentage of instances correctly predicted. TPR is calculated as percentage of positives (P) correctly predicted (TP) and FDR as percentage of all predicted positives (TP + FP) that are falsely predicted positive (FP).

4.1.1 Logistic Regression Results

The average Accuracy, TPR and FDR for cross-validation and percentage split testing approaches together are 71.2%, 48.0%, and 14.2% respectively for the random data set. FDR results suggest that we can say with a confidence of approximately 86% that an application in a testing data set is safe to use.

WEKA allows the user to generate a threshold curve where the points in a threshold curve record various statistics such as true positives, false positives, etc. The curves

	Cross-Validation (10 Folds)			Percentage Split (66%)		
Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate(%)	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	73	48.5	8.1	66.3	51	20
2	74.1	49.2	8.4	64.5	0.5	19.1
3	78.2	50.1	8.9	68.7	49.7	18.4
4	75.8	52.1	9.9	68.9	51.9	21.8
5	73.7	48.4	9.2	67.9	54.5	18.9
6	70.2	51.8	8.7	65.6	53.2	17.5
7	77.5	49.8	9.5	69.3	51.9	19.9
8	78.2	46.7	8.3	68.6	50.6	20.6
9	74.6	53.2	11.1	70.2	48.7	23
10	71.9	48.9	9.3	67.7	49.5	22.8

Table 4.1: Logistic Regression - Random Data Set

are generated by sorting the predictions produced by the classifier in descending order of the probability it assigns to the positive class. Since we know the actual class labels in our historical data, we can step through this list and compute the number of true positives, etc. at each point in the list (each step in the list becomes one point on the curve). Each point in the list also corresponds to setting a threshold on the probability assigned to the positive class (hence the name "threshold curve"). The accuracy reported in the Explorer in WEKA corresponds to one point on the Receiver Operating Curve (ROC) - namely where the default threshold of 0.5 is set in a binary class problem. Figure 4.2 shows the variability of the TPR along Y-axis with respect to changes in the threshold value for category id 1. It is easy to notice the two extreme points on the curve where threshold value is set > 1, $TPR = 0$ and threshold value is 0, $TPR = 1$.

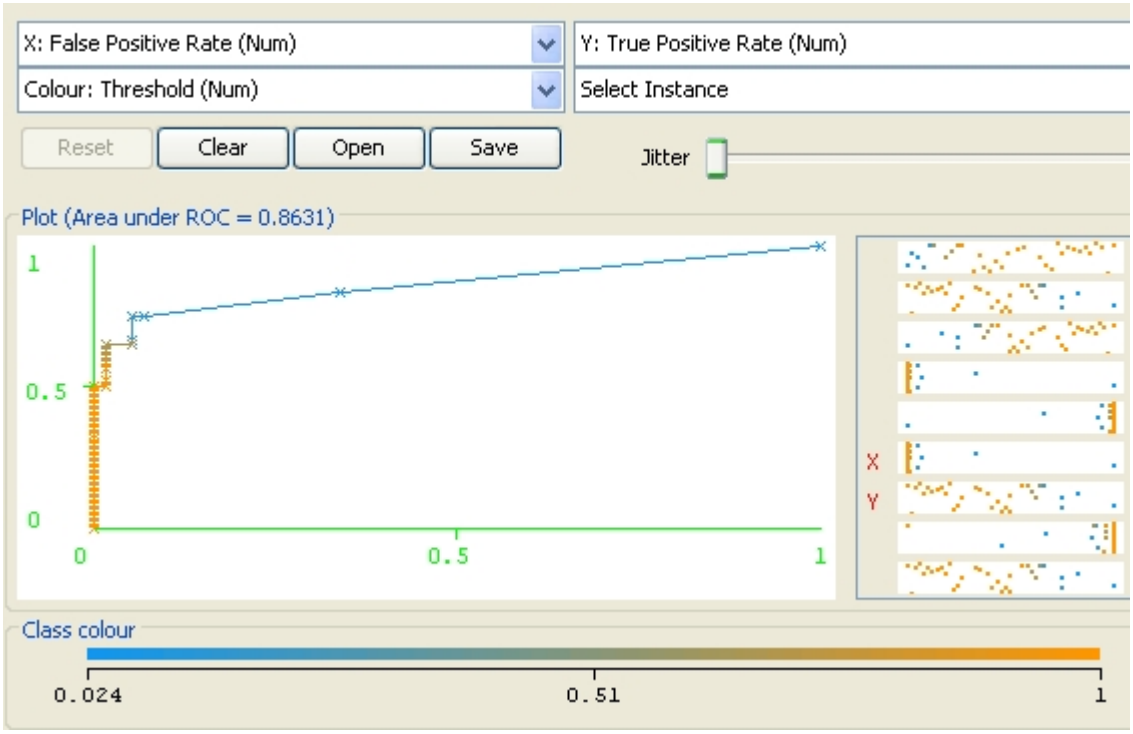

Figure 4.2: Threshold Curve

The issue with the random data set that there may be two applications that may have similar permission settings but different class type, which makes it harder for the model to find a good regression model. Therefore, the results with the filtered data set are better than with the random data set of applications. The filtered data set is based on the applications that have different permission settings compared to the safe applications. The results from the filtered data set are shown in Table 4.2. The average Accuracy, TPR, and FDR for cross-validation and percentage split testing approaches together are 86.1%, 70.8%, and 7.2% respectively for the random data set. FDR results suggest that we can say with a confidence of approximately 92% that an application in a testing data set is safe to use, a gain of 6 percentage

points over random data set.

	Cross-Validation (10 Folds)			Percentage Split (66%)		
Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate(%)	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	88.3	72.8	4.8	80.8	64.6	9.6
2	89.1	70.3	4.3	82.4	68.3	9.8
3	86.5	76.3	4.6	84.2	66.4	10.2
4	90.2	74.7	5	85.5	67.3	9.1
5	89.3	76.8	5.6	84.3	69.2	9.5
6	85.1	71.1	5.4	83.2	71.2	10.4
7	92.2	73.5	4.5	87.6	67.5	9.7
8	87.6	70.2	5.1	81.3	68.1	9
9	88.9	77.4	4.9	86.2	68.6	9.4
10	84.6	76.9	4.8	84.6	64.6	9.2

Table 4.2: Logistic Regression - Filtered Data Set

Logistic regression results are sensitive to differences between permissions of safe and unsafe applications given that there are 121 permission settings for each application. The results are also sensitive to ratio of safe to unsafe applications in the training data set. As the number of unsafe applications in the training set increase

compared to safe applications, Figure 4.2 shows that TPR has a downward trend, whereas accuracy increases because of improvements in better prediction of unsafe applications.

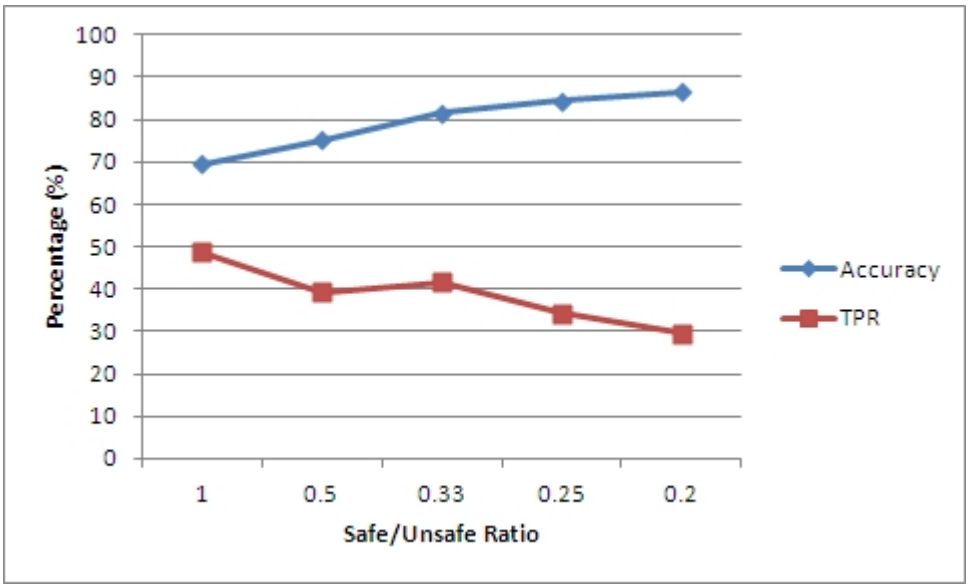

Figure 4.3: Class Ratio Impact

4.1.2 Decision Tree Results

We use j48 decision tree module in WEKA to determine the safety risk of an application. The average Accuracy, TPR, and FDR for cross-validation and percentage split testing approaches together are 66%, 40.5%, and 33.5% respectively for the random data set. FDR suggests that we can say that an application predicted by the model to be safe is safe with a confidence of approximately 96%. The results from the random data set are given in Table 4.3. Decision tree results with random data

	Cross-Validation (10 Folds)			Percentage Split (66%)		
Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate(%)	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	69.7	38.1	16.4	67.5	40	36.1
2	66	46.7	38.5	63.4	45.3	40
3	65.5	44.3	34.5	62.3	38.4	41.2
4	68.3	45.3	30.8	61.9	41.7	36.8
5	69.4	39.4	28.4	64.3	36.6	39.1
6	70.1	41.8	19.3	67.8	35.8	41.2
7	67.4	42.9	24.6	63	37.1	44.3
8	68.2	41.8	28.1	64.2	39.3	42.1
9	64.5	43.9	27.3	60.3	33.8	36.9
10	69.8	40.4	18.7	67.3	38.3	47.2

Table 4.3: Decision Tree - Random Data Set

set are not very promising due to the reason that it is hard to find a clear distinction between permission settings of safe and unsafe applications. We also test the decision tree model with the filtered data set where it is easy to find a pattern of permission settings among safe and unsafe applications. The j48 model does a better job in predicting safe and unsafe applications using both cross-validation and percentage split approaches. The average Accuracy, TPR, and FDR for cross-validation and percentage split testing approaches together are 86.3%, 70.4%, and 4.0% respectively for the random data set. FDR values are very low compared to the results with random data and that means it would be easy to predict an application as safe when it really is safe. The results are shown in Table 4.4. When we visualize the decision tree, the classification is mainly based on the permission settings and the description of an application doesn't play any role in deciding the application safety risk. The j48 classifier displays additional information, including a text representation of the tree

	Cross-Validation (10 Folds)			Percentage Split (66%)		
Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate(%)	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	88.3	73.6	3.1	84.6	68.8	2.9
2	89.6	70.3	3	81.3	65.6	3.1
3	91.3	71.2	3.5	82.7	69.3	3.8
4	87.9	75.4	4.1	83.9	71.1	5.4
5	90.2	73.2	4.5	80.1	67.3	5.2
6	91.2	70.9	3.4	85.7	67.3	4.8
7	90.3	75.1	3.8	87.4	68.2	4.1
8	89.7	76.2	3.3	82.8	71.9	3.9
9	87.6	71.2	4.2	83.5	63.4	4.8
10	88.1	69.8	4.8	80.9	69	5

Table 4.4: Decision Tree - Filtered Data Set

it uses to perform evaluations as shown in Figure 4.3. One of the main advantages of the j48 classifier is that is relatively quick to train, and should finish almost immediately on a small data set, such as the one we are using. Once a classifier has been found that performs suitably well on the data, it can be saved to be used later. One of the weaknesses of using a decision tree is that it is too simplified a classification. For example, in figure 4.3, the decision tree is generated for "entertainment" category training data set, 52 applications are safe just because they are using "read phone state and identity" permission setting. A random application using this permission, j48 classifier would predict the application to be safe ignoring its description.

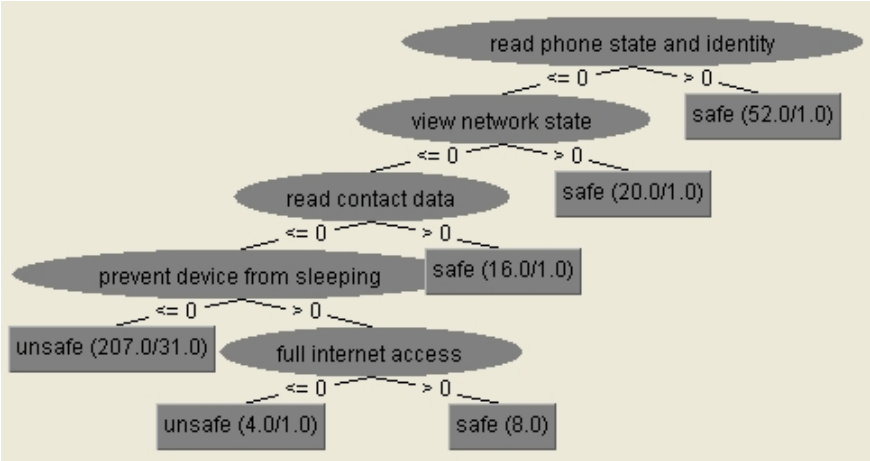

Figure 4.4: Tree View

When we analyze the threshold curve for j48, TPR increases very fast with the decrease in the threshold value from 1 as shown in Figure 4.5.

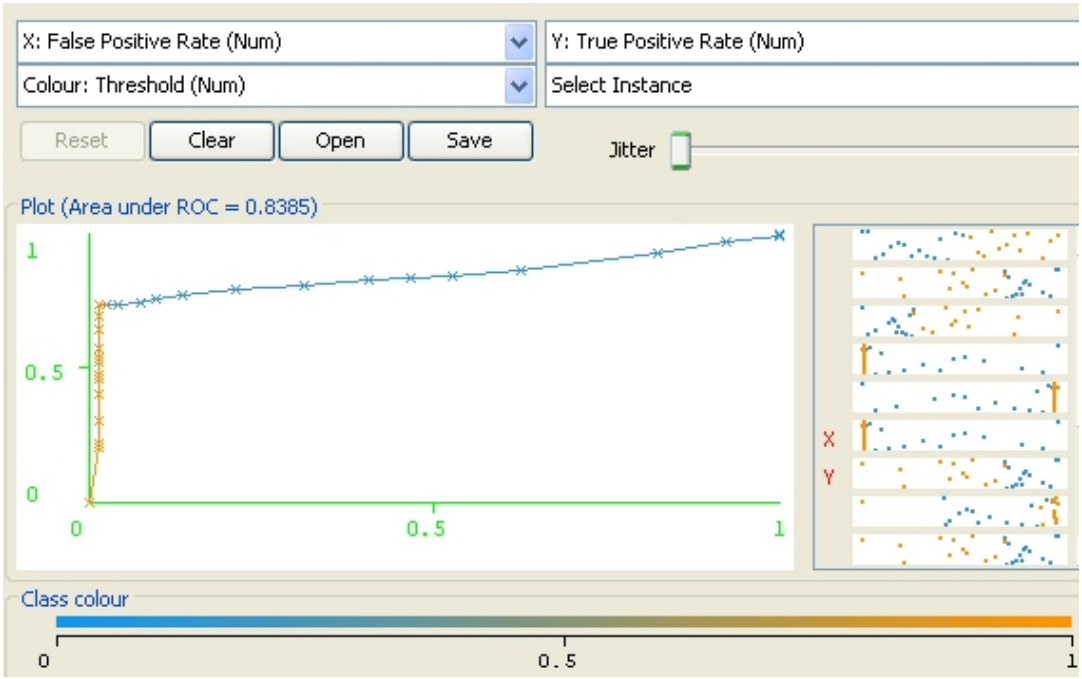

Figure 4.5: Threshold curve - j48

4.1.3 Multilayer Perceptron Results

Multilayer Perceptron (MLP) does not make any assumption regarding the underlying probability density functions or other probabilistic information about the pattern classes under consideration in comparison to other probability-based models. The goal of the training process for MLP is to find the set of weight values that will cause the output from the neural network to match the actual target values as closely as possible. As we know the results on random data set are not better than with filtered data set, we are only providing results for the filtered data set with MLP model training. The average Accuracy, TPR, and FDR for cross-validation and percentage split testing approaches together are 70.3%, 77.4%, and 24.0% respectively for the filtered data set.

	Cross-Validation (10 Folds)			Percentage Split (66%)		
Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate(%)	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	75.2	77.2	20.2	67.5	75.7	39
2	75.2	73.2	16.9	70.3	67.6	23.7
3	70	78.2	21.1	65.5	73.4	35.2
4	71.9	82.4	15.4	69.2	75.6	36.7
5	69.2	85.8	14.9	68.3	78.3	30.2
6	76.3	81.3	18.8	64.8	74.8	28.4
7	77.4	83.4	17.3	69.1	76.2	25.3
8	72.6	80.5	16.3	70.1	76.8	28.4
9	71.5	86.1	21.2	68.1	79.1	25.9
10	69.1	77.5	17.4	65.3	69.3	28.4

Table 4.5: MLP - Filtered Data Set

4.1.4 Classification Results Summary

It is important to compare the three classification techniques to find the best classification technique for determining the safety risk of an application. Figure 4.6 shows the analysis of accuracy performance indicator and it is clear that decision tree and logistic regression are comparable.

Figure 4.6: Accuracy Comparison

Figure 4.7 shows the comparison for TPR among all three techniques where MLP performs better than rest of the two approaches. MLP is able to predict most of the applications that are actually safe as safe applications. It is also important to compare FDR to make a conclusive result. Figure 4.8 shows the comparison for FDR; decision tree performs much better than Logistic Regression and MLR. There

is no single classification technique that outperforms others in all three performance indicators.

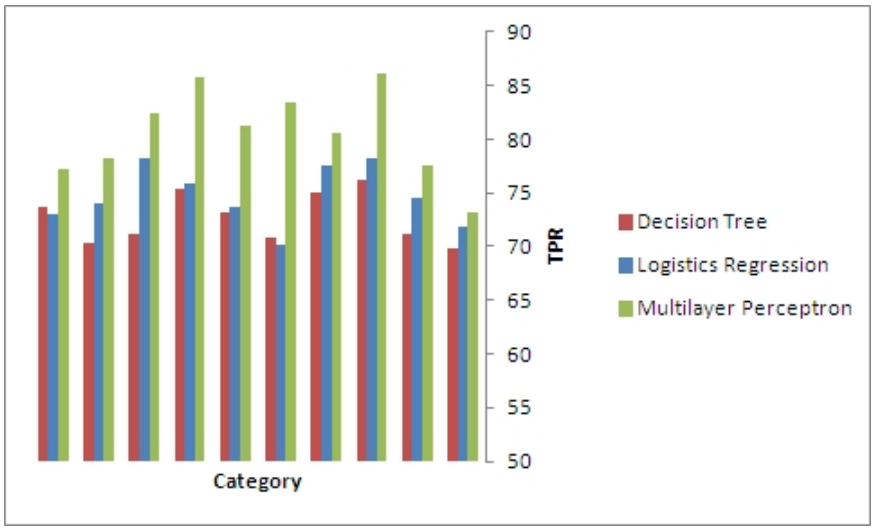

Figure 4.7: TPR Comparison

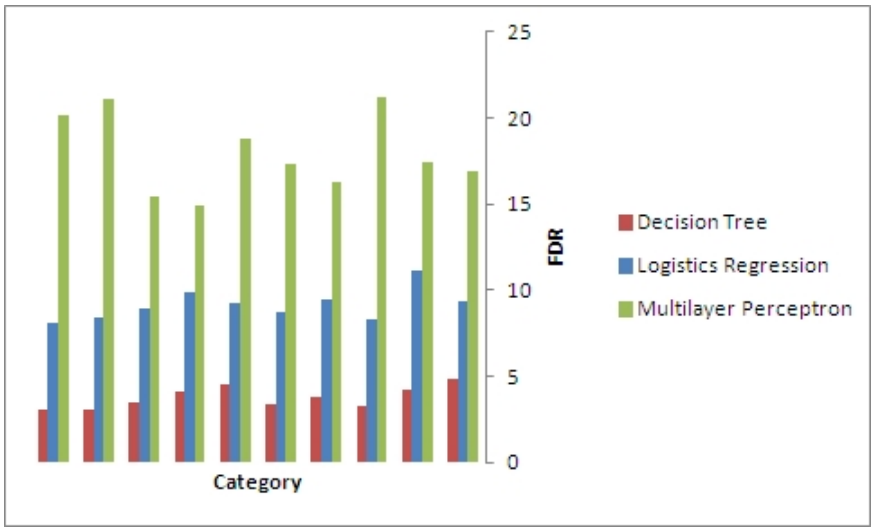

Figure 4.8: FDR Comparison

Decision tree has a very low False Detection Rate, which makes it the most suitable technique, as TPR and Accuracy indicators are also manageable. The weakness in the classification techniques is that they don't use the description of an application in the output model and solely predict the applications based on the permission settings. The results from the random data set are not good enough for safety risk detection of an application. There is always some overlap among permissions for safe and unsafe applications that complicates the output model. The next subsection documents the results from the clustering technique based on the TMT model.

4.2 TMT Results

We cluster the applications that belong to a single category into different groups. Each group has a topic associated with it which is essentially a set of keywords covering the entire set of applications that belong to it. We use the description of applications for grouping and don't consider the permission settings at all. The reasoning is that the permissions are a consequence of an application's functions and therefore are not independent variables. The independent variables of an application for our purpose are its category and permissions. We use empirical data to determine the number of groups to be created from the training data set. We have found that at least 4-5 applications should belong to a group. Therefore, the total number of groups to be created by the TMT model is equal to the total number of applications divided by the expected number of the applications in a group e.g. 5. TMT provides a probability distribution for each application across the different groups. In our

testing, we use a cut-off probability of $\alpha = 0.3$ to determine whether an application belongs to a group. Also, we use the following values of weights $w_{c,p,1}$ and $w_{c,p,2}$ in our testing.

Weight	Low Risk Permission	Medium Risk Permission	High Risk Permission
$w_{c,p,1}$	1	1	1
$w_{c,p,2}$	1	5	10

Table 4.6: Weights

The advantage of using weights is that an application may access a highly risky permission but still be safe if most applications of the group to which the given application belong access the same risky permission. In other words, if the description of an application is such that it needs the highly risk permission for its intended functions, the application won't be classified unsafe. Also, if an application accesses permissions outside the group permissions to which it belongs, the application may be safe or unsafe depending upon the weights of those permissions accessed outside the group permissions. Solving equation 3.4 empirically by using the training data set, we find the value of $\beta = 0.55$ which is needed before we determine to be safe or unsafe. Another advantage of using TMT model is that we don't have to define an application to be safe or unsafe as part of the input in the training data set compared to classification techniques. This means that there is no user-defined input in the training data set and hence, no user bias in the output model. In other words,

the classification output model is a function of the type of applications defined as safe and unsafe, whereas the TMT model always provide the same output model irrespective of what types of applications are actually safe or unsafe. Also, for the TMT model, the data set doesn't need to be filtered based on the permissions, as we are not considering the permissions as input. The TMT model is run separately on each category and the results are provided in Table 4.7 below based on the confusion matrix generated for the testing applications in the inference data set. The average Accuracy, TPR, and FDR are 90.0%, 90.4%, and 3.5% respectively for the inference data set. The testing for TMT is performed on 20 categories to show robustness in the results.

The TMT model is quite sensitive for applications that access only a few permissions, like 2 or 3. In such cases, if an application accesses few permissions outside the group's permissions to which it belongs, the application is highly likely to be classified as unsafe. Applications that access no permissions are obviously classified as safe but an application accessing a single permission outside the group's permissions is classified as unsafe. It is helpful if an application belongs to more than one group to reduce the sensitivity to such cases.

Category	Accuracy (%)	True Positive Rate (%)	False Discovery Rate (%)
1	88.5	80.2	2.5
2	90.3	90.9	2.4
3	92.6	92.8	3.3
4	92.1	93.7	4.1
5	92.6	96.1	4.4
6	92.8	93.3	2.6
7	92.2	95.6	3.1
8	89.8	91.7	2.7
9	88.8	93	4
10	89	89.3	3.9
11	91.6	87	4.5
12	94.7	89.9	3.5
13	86.7	86.6	3.9
14	93.5	86	3.8
15	85.8	91.7	3.5
16	86.6	90.8	4.1
17	86.9	91.5	3
18	85.9	90.5	4.1
19	90.3	90.6	3
20	88.9	86.2	4.4

Table 4.7: TMT Results

Chapter 5

Conclusion

It has been reported by some security firms that almost 25 percent of Android applications feature code that can access application permissions and cause security vulnerabilities. It is practically impossible for users to distinguish "high risk" applications from the safe ones on their own. Therefore, it is important for end users to know that an application is safe to install especially in Android-based platforms. We conclude from this work that it is possible to objectively determine the security risk of an application with a high confidence factor. The security risk calculation is based on the assumption that an application's permission settings should represent its intended functions based on its description. It is also clear from the results that clustering techniques are better than classification techniques to predict the security risk of an application.

Since we only looked at an application's description and its permissions for security risk determination, it is possible that an application has permission settings

matching its description but it is still possible that an application is unsafe due to a faulty code in the application. Faulty code in an application can allow hackers to track user phone numbers, modify user's bookmarks, and push unwanted ads onto user's devices. Therefore, for the future work, it needs to be explored how such "high risk" applications can be predicted effectively. The research can be further extended to include the user behavior regarding "high risk" applications i.e. how users react when they are notified about the risk posed by an application from its permission settings.

Bibliography

[1] Global mobile statistics 2013 Section E: Mobile apps, app stores, pricing and failure rates. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/e, May, 2013.

[2] 25% of Google Play apps pose a security risk. http://www.net-security.org/secworld.php?id=13891,November, 2012.

[3] T. Bradley. DroidDream becomes Android market nightmare. http://www.pcworld.com/businesscenter/article/221247/droiddream_becomes_android_market_nightmare.html, 2011.

[4] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. *In Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation,* pages 393-407, 2010.

[5] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A Study of Android Application Security. *In Proceedings of the 20th USENIX Conference on Security,* 2011.

[6] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, and B. Shastry. Towards Taming Privilege-Escalation Attacks on Android. *In Proceedings of the 19th Network and Distributed System Security Symposium,* 2012.

[7] J. Burns. Developing Secure Mobile Applications for Android. *iSEC Partners,* October, 2008. http://www.isecpartners.com/files/iSEC_Securing_Android_Apps.pdf.

[8] WEKA (Machine Learning) http://en.wikipedia.org/wiki/Weka_(machine_learning).

[9] D. Hand, Heikki, M. P. Smyth. Principles of Data Mining. *PHI Learning.* http://www.phindia.com.

[10] S. Haykin. Neural Networks—A Comprehensive Foundation, second ed., Prentice-Hall, New Jersey, 1999.

[11] S. Mitra, S. Pal, P. Mitra. Data Mining in Soft Computing Framework: A Survey, *IEEE Trans. Neural Networks,* 13 (1) 3-14, 2002.

[12] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. *In Proceedings of the IEEE International Conference on Security & Privacy,* pages 50-57, 2009.

[13] P. McDaniel and W. Enck. Not So Great Expectations: Why Application Markets Haven't Failed Security. *IEEE Security & Privacy,* 8(5): 76-78, 2010.

[14] M. Costa, J. Crowcroft, A. Rowstron, L. Zhou, L. Zhang and P. V. Barham. End-to-End Containment of Internet Worms. *In Proceedings of the ACM Symposium on Operating Systems Principles,* 2005.

[15] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry. Practical and Lightweight Domain Isolation on Android. *In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM),* 2011.

[16] P. Gilbert, B.-G. Chun, L. Cox, and J. Jung. Automating Privacy Testing of Smartphone Applications. *Technical Report CS-2011-02, Duke University,* 2011.

[17] C. Marforio, F. Aurelien, and S. Capkun. Application Collusion Attack on the Permission-based Security Model and its Implications for Modern Smartphone Systems. *Technical Report 724, ETH Zurich,* 2011.

[18] M. Ongtang, K. Butler, and P. McDaniel. Porscha: Policy Oriented Secure Content Handling in Android. *In 26th Annual Computer Security Applications Conference (ACSAC),* 2010.

[19] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. *IEEE Security and Privacy,* 2009.

[20] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici,S. Dolev, and C. Glezer. Google Android: A Comprehensive Security Assessment. *Security & Privacy, IEEE,* 8(2):35-44, 2010.

[21] A. Porter Felt, E. Chin, S. Hanna, D. Song, andD. Wagner. Android Permissions Demystified. *In Proceedings of the 18th ACM Conference on Computer and Communications Security,* pages 627-638, 2011.

Appendix A

TMT Prediction Examples

This appendix provides the examples where the TMT model is applied to test its prediction accuracy on the applications from the Google Play Store. We use the value of $\beta = 0.55$ for these examples.

A.1 Hacker's Home

Hacker's Home is an application that belongs to the "Education" category. Hacker Home is the Android application which includes information realted to hacking computer and other hacking tricks. After inference is done, the application belongs to group "Group 9" with probability 37%. Hacker's Home is predicted to be safe based on the TMT model even though the description of the application may bring some suspicion into its activites. The prediction details are given as:

Group	Probability
Group 9	33%

Hackers Home Permissions	Group 9 Permissions
Full Internet Access	Full Internet Access
View Network State	View Network State
	View Wi Fi State
	Modify delete usb storage contents, modify delete sd card contents

Based on the above Permissions comparison, the **Hackers Home app** doesn't look suspicious.
P(App is safe) = 1 + 1 / (1 + 1) = 100% > ***beta*** , APP is **SAFE** to use.

Figure A.1: Hacker's Home

A.2 Mathway App

This application provides answers to math problems in the "Education" category. The TMT model predicts that it belongs to two different groups "Group 1" and "Group 9" with probabilities 47% and 33%. Given the description of the application, we have defined it as "safe" for use and the model also predicts to be "safe".

Group	Probability
Group 1	47%
Group 9	33%

Mathway Permissions	Topic 1 Permissions	Topic 9 Permissions
Full Internet Access	Full Internet Access	Full Internet Access
View Network State	View Network State	View Network State
View Wi Fi State	read phone state and identity	View Wi Fi State
	Modify delete usb storage contents, modify delete sd card contents	Modify delete usb storage contents, modify delete sd card contents
	prevent device from sleeping	

Figure A.2: Mathway App

A.3 GrooVe IP

GrooVe IP is an Android application that connects to Google Voice using Voice over IP. This application belongs to the "Communication" category. We want to test this application assuming as if it belongs to the "Education" category making it suspicious. TMT model is able to detect the "GrooVe IP" as unsafe for the "Education" category apps. The model inferences this application to belong to only one group "Group 4".

Group	Probability
Group 4	47%

GrooVe IP Permissions	Group 4 Permissions
Full Internet Access	Full Internet Access
View Network State	View Network State
Create Bluetooth connections	Create Bluetooth connections
Prevent device from sleeping	Prevent device from sleeping
Control vibrator	Control vibrator

There are **15** other permissions that GrooVe IP App accesses that are **NOT** under **Group 4 allowed permissions**, making it highly **suspicious** app under **Education** Category.

GrooVe IP vs. Group 4	P(App is SAFE) = 5/20 = 25% < *beta* UNSAFE

Figure A.3: GrooVe IP

When we set the category to its actual category "Communication" and then use the training model learnt on the applications belonging to "Communication" category for inference, we get a different result as the model suggests it to be "safe" to use. We give both the weight types a value of "1" for this example.

Group	Probability
Group 4	43.2%

Groove IP vs. Group 4	**17 out of 20 permissions MATCH**

GrooVe IP vs. Group 4	P(App is Safe) = 17/20 = 85% > beta, **SAFE**

Figure A.4: GrooVe IP - Real Category