INTRODUCTION

1.1 Context

Nowadays, Android is one of the mobile market leaders, giving more than a million applications on Google Play store. Current the year 2015, in the average, 135 millions of applications per day were installed by Android users. Android applications can be written by any developer and do not need any certification or validation before being made available on the store. Thus, abusive Android applications are available on Google Play [1, 2, 3, 4, 5]. These applications collect user's data and require access to sensitive services not related to their functionalities. To solve this security problem, Google uses a permission system. The permission system method design a list of permission required by each Android application for its installation. The problem is that the permission list is most often ignored by end-users. The fact that the permission list may provide information about the behavior of the application gives the opportunity to automatize the analysis of the applications. The exploitation of this issue may conducted to the classification or the categorization of the Android applications. For this purpose, strong key permission and expected permission requests may be well identified. This work use graph modeling to identify the normal behavior of application and the expected permission requests in order to classify the applications by category. So, each application is assigned to the group (category) which has the same graph pattern.

1.2 Background

One of the Android platform security mechanisms and a principle user warning system of Android is the permission system. Each application has very limited capabilities by default, and needs to require permissions to access sensitive data or services. Users are prompted with a list of the permissions required by an application just before the installation. This list is supposed to warn users about hazardous and abusive applications, but, unfortunately, permission lists have been shown to be ineffective for this purpose. First, users see permission lists as a repeated warning or a license agreement that must be accepted to obtain a service. Permission lists are only shown in the final step before installation when other criteria for the user's decision have been met, and therefore the permission list is considered an obligation rather than a decision factor [7, 6]. Second, users often do not have enough background to understand the meaning of permissions and their possible harm. Third, permissions are shown entirely out of the context, which prevents the user from understanding their purposes. Finally, some permissions are so frequently required that users do not pay any attention to them [8, 9].

It can be seen that there is at present no system that helps Android users to take a decision aimed at more privacy-respecting and secure Android applications. Users must either rely on the community with comments and ratings, which rarely refer to possible security problems, or manually verify permissions and rely on their personal knowledge and understanding. Authors of previous studies, such as [10], highlight the need for a new security and privacy indicators for mobile users

1.3 problem and research question

Studies was conducted to analyses permission usage of Android application but limit their works to the detection of correlation between permission requests and other application attributes, such as price and rating [11, 12, 4, 13]. Malware detection by defining malware-specific behavior [14, 15, 16, 17, 18, 19]. The detection approach is limited to the known malware. Therefore unknown malware and applications that abusively require permissions would stay undetected.

This work address the problem of the categorization of Android applications using the graph patterns detection. This problem can formulated as follows: given an application with an unknown category, can we predict the actual category of this application? The objective of this work is to identify whether the actual class corresponds to the predicted class. The hypothesis of the research are formulated as follows:

1. Application category contains similar applications that would use similar permissions. Therefore, an average or 'normal' category application could be represented by a permission pattern.
2. Different application categories contain different applications. Therefore, the patterns that characterize one category should differ from the pattern characterizing another category. In this case, patterns should permit to identify the category of an application by permissions this application requires.
3. A pattern characterizing 'normal' applications of a category should permit to measure the risk level of an application and to detect abnormal applications: applications abusively requiring permissions, bad-quality applications, applications from wrong categories and malware. By hypothesis, the more applications request permissions that are not normally observed in the category, the higher is its risk score.

From the hypothesis highlighted above, two main research questions were formulated: (1) Do Android applications of different categories require different permission patterns and can be distinguished by patterns? (2) Can a category pattern allow to measure an application risk/privacy level and permit malware detection?

2- Related works

Reference	Benign application analysis	result
13	Top 50 free Android applications of 2009	Application in the same category do not necessarily require the same permissions
4	most popular and new Android applications	Popular applications requested more permissions than news application
12	unsupervised learning	different permission requests for distinct categories
Reference	**Permission-based decision support systems**	**Usable fot Android applications**
22	crowd-sourcing system	No
23	searching for a justification of permission usage in application descriptions with NLP	Not verified for all
25	risk warning system based on the occurrence of 24 permissions (identified by author as dangerous	Yes
Reference	**Malware applications analysis**	**Tools used**
15	supervised machine learning techniques to categorize 820 applications into 7 categories	Term Frequency (TF) and Inverse Document Frequency (IDF); Bayesian Tree Augmented Naive and Random Forest Algorithms, K-Nearest Neighbor (KNN) methods
16	classified applications into two main categories : games and tools	Chi-Square (CS), Fisher Score (FS) and Information Gain (IG) methods
17	analysed the use of the permissions in a set of 1,227 clean and 49 families of malicious Android applications	Andrubis
18	Extracted permissions and API calls from benign and malicious applications	SVM, RaJ48 Decision Tree and Bagging algorithms
Reference	Permission verification tools	
27	tool informing a developer about unnecessary permissions	different behaviours between benign and malicious applications in permissions usage
5	tool called Permlyzer	
2 3 28 29 30 31,32 33	TaintDroid AppFence SCAndroid AndroidLeaks ScanDal Blue Seal FlowDroid	monitor and fully control data flow in application but not clear if the user is able and willing to adopt those technologies

3- Methodology employed

To highlight their methodology, the authors of the present work make further assumptions. First the risk of an application would increase with the number of permissions used. As consequence, many malicious applications use very few permissions and some very popular and functional applications may request many permissions. Second the final score of one application depends on all other applications in the dataset and all scores must be recalculated when one application is added or deleted from the dataset. The work is now focus on the evaluation of the risk of a given Android application and detects abnormal applications. The authors calculate the risk of an application based on the proximity of its permission request with a pre-calculated normal behavior. Therefore, even if an application requests few permissions but they deviate from the expected request in the given category, it would be considered abnormal.

This methodology can be summarized into three main steps:

1) The construction of a category pattern to identify the permissions expected for a given application. We assume that applications grouped into categories provide similar functionalities and would therefore require similar sets of permissions.
2) Application classification to highlight the most relevant patterns.
3) The risk metrics that aim to detect abusive applications. The overall objective is to provide a warning system that remains within the proposed patterns and risk metrics.

Figure 1 show the five steps of pattern construction methodology.

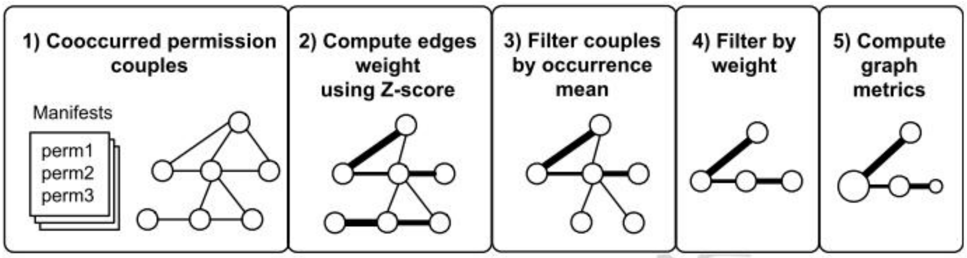

Figure 1: Five steps of pattern construction methodology

The performances of the proposed methodology was tested using a 10-fold cross-validation. The dataset is randomly divided into 10 equal subsets. The training is performed on 9 folds and the classification is performed on the last fold. This process is repeated for several rounds, and helps to avoid data overfitting problems. Our dataset contains different numbers of applications for different categories and a misbalance in the number of class members can bias the classification. To avoid this bias, we applied a distribution-based balancer to our dataset as a pre-processing step [40]. This procedure allowed us to obtain 300 members for each category of applications. We tested several classification methods such as Random Tree, Support Vector Machine, Naive Bayesian, etc. For clarity in this paper, we will only present the Naive Bayesian method, which provides the best classification results.

4- Results founds and perspectives

Figures and tables below describe the main results obtained by the proposed approach.

Table 3 illustrates the number of permissions and permission pairs left for each pattern after the execution of the methodology presented in Figure 1

Category	Nodes	Edges	Category	Nodes	Edges
communication	59	835	weather	23	53
app_widgets	61	602	casual	17	52
productivity	61	475	photography	22	40
tools	58	436	finance	22	40
social	44	260	medical	26	40
personalization	37	221	media_and_video	22	39
app_wallpaper	29	183	health_and_fitness	21	35
entertainment	30	143	lifestyle	28	32
travel_and_local	37	140	game-wallpaper	19	29
business	49	138	transportation	19	24
music_and_audio	29	97	books_and_references	19	22
libraries_and_demos	21	79	cards	13	12
shopping	22	56	sports	12	9
comics	24	55	news_and_magazines	9	7
arcade	19	53	sports_game	7	7
game_widgets	20	53	racing	5	3

Table 3: Number of permissions and co-required permissions by pattern.

The table 4 provide the results of the Naive Bayesian algorithm for classifying the applications into categories using pattern-related features

Metric	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area	Correctly classified (%)
All 7 metrics	0.809	0.006	0.818	0.809	0.809	0.995	80.86
Betweenness & Weighted & Degree & PageRank & HubAuth & PermCount	0.787	0.007	0.794	0.787	0.786	0.994	78.66
Betweenness & Weighted & Degree & PageRank & HubAuth	0.769	0.007	0.78	0.769	0.77	0.993	76.87
Betweenness & Weighted & Degree & PageRank	0.712	0.009	0.731	0.712	0.715	0.99	71.21
Betweenness & Weighted & Degree	0.653	0.011	0.681	0.653	0.657	0.985	65.34
Betweenness & Weighted	0.559	0.014	0.599	0.559	0.565	0.973	55.85
Betweenness	0.412	0.019	0.46	0.412	**0.417**	**0.941**	41.23
Weighted Degree	0.338	0.021	0.329	0.338	0.331	0.918	33.75
Degree	0.322	0.022	0.312	0.322	0.315	0.914	32.20
PageRank	0.315	0.022	0.306	0.315	0.308	0.911	31.54
Authority/Hub	0.308	0.022	0.299	0.308	0.301	0.908	30.81
Closeness	0.27	0.024	0.26	0.27	0.263	0.89	27.04
PermCount	0.265	0.024	0.256	0.265	**0.258**	**0.891**	26.54
Binary	0.15	0.03	0.15	0.15	**0.14**	**0.72**	15.24

Table 4: Classification results for each metric and combinations of metrics

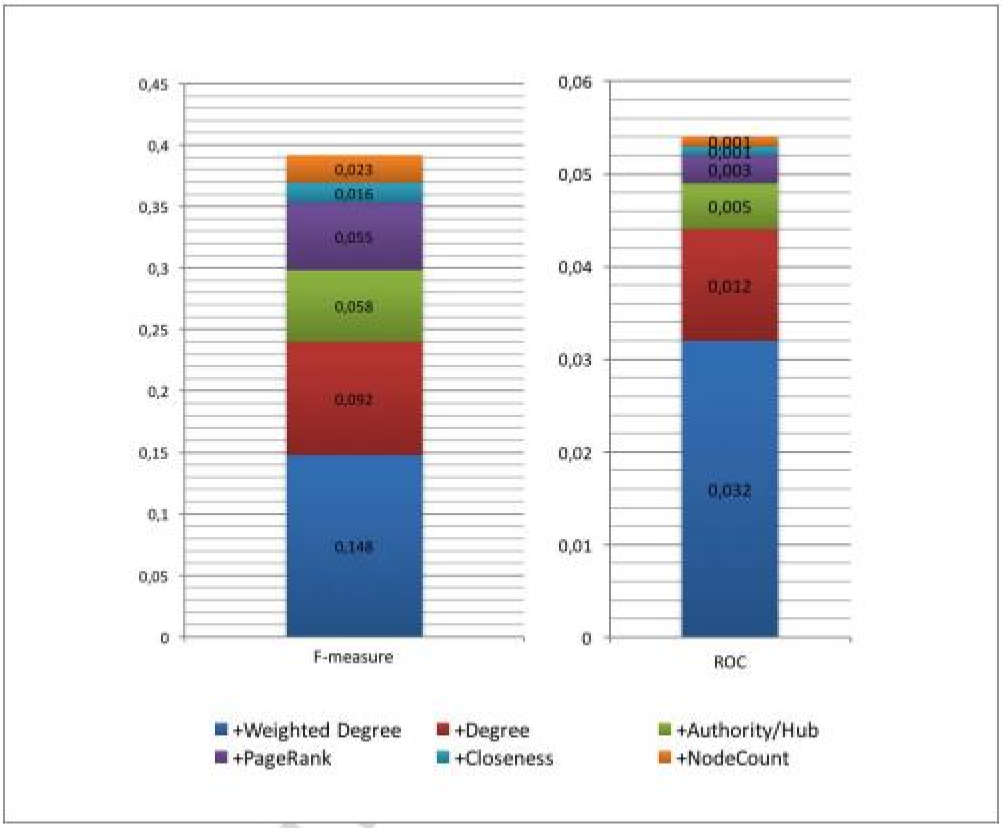

Figure 5: Performance gain brought while adding all metrics one by one to betweenness centrality

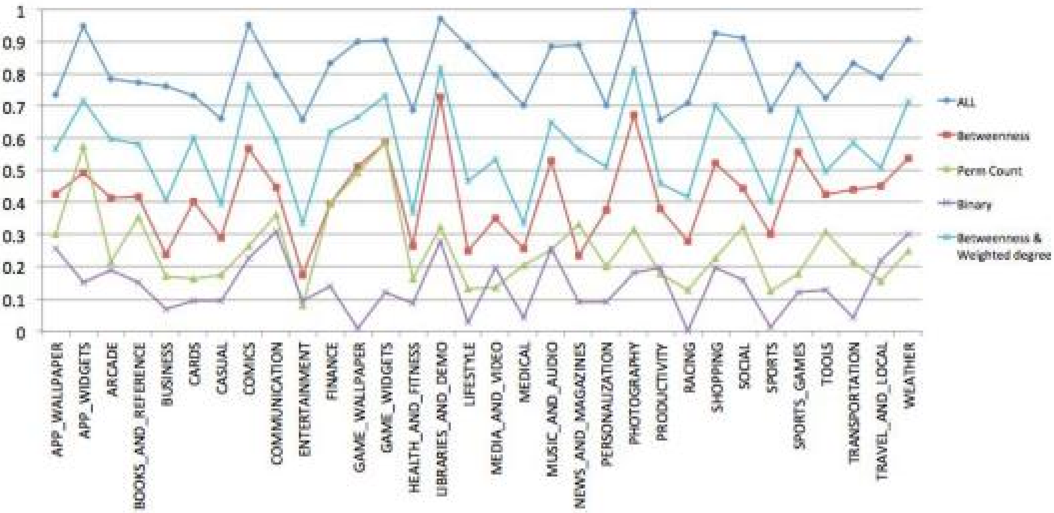

Figure 6: Binary vector and pattern-related features comparison regarding F-measure and all categories

Beta	F-measure	F_2-measure	$F_{0.5}$-measure	Min Likeness LN_0
1	0.738	0.640	0.862	0
2	0.761	*0.568*	0.860	10
3	**0.798**	0.658	0.872	10
4	0.772	0.692	**0.874**	10
5	0.770	0.690	**0.874**	30
0	*0.661*	0.679	*0.595*	150
Perm	0.670	**0.701**	0.780	N/A

Table 5: Best results for F-measure, F_2-measure and $F_{0.5}$-measure for different β

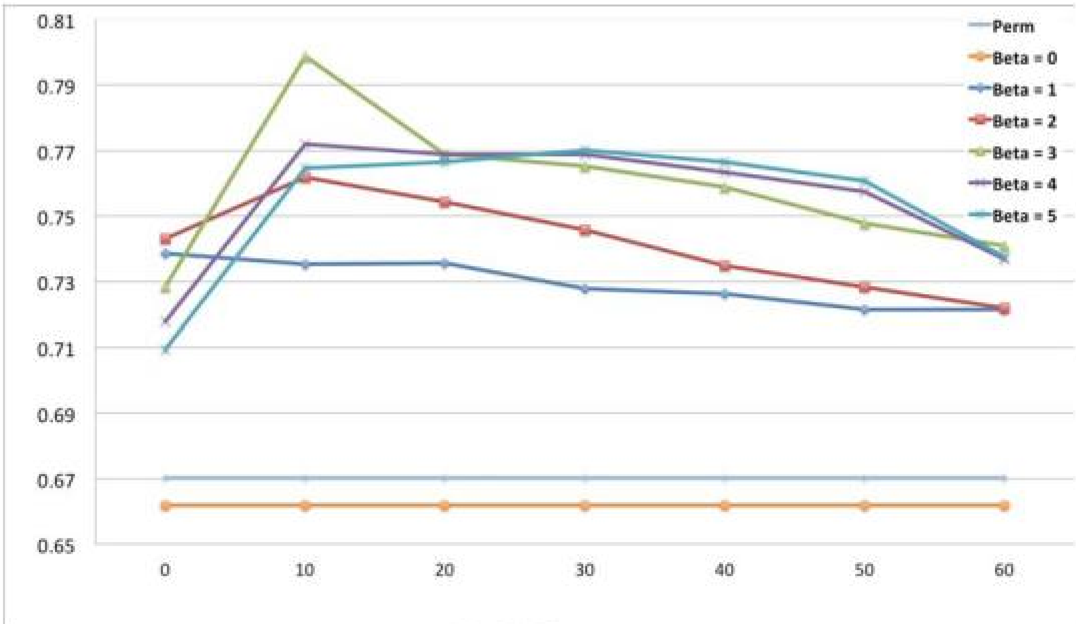

Figure 7: Graph represents how the minimal likeness and beta affects risky application detection.

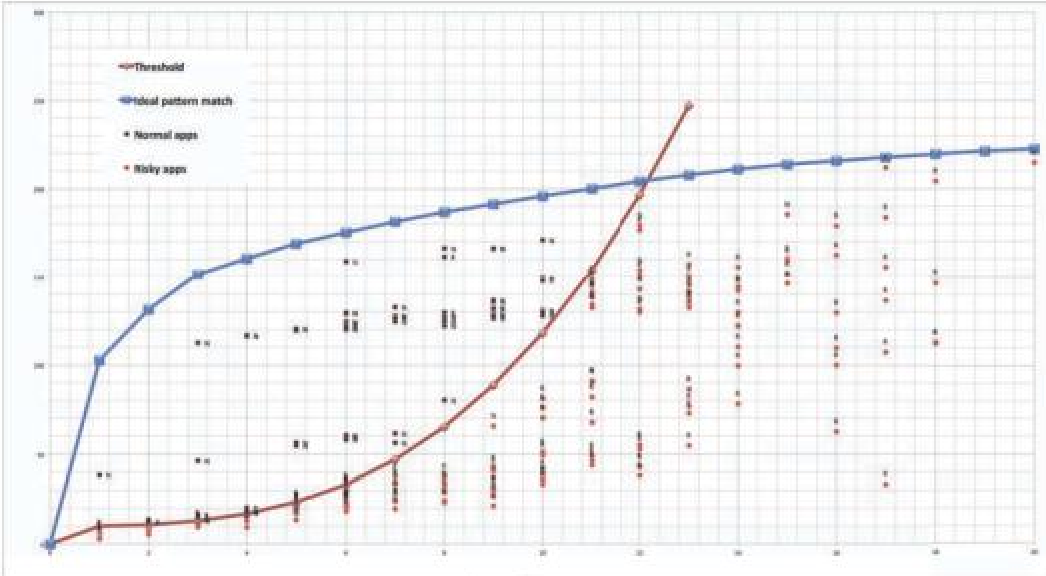

Figure 8: Optimal threshold for detecting risky applications $\beta = 3$, $threshold = 0.108$, $LN_0 = 10$

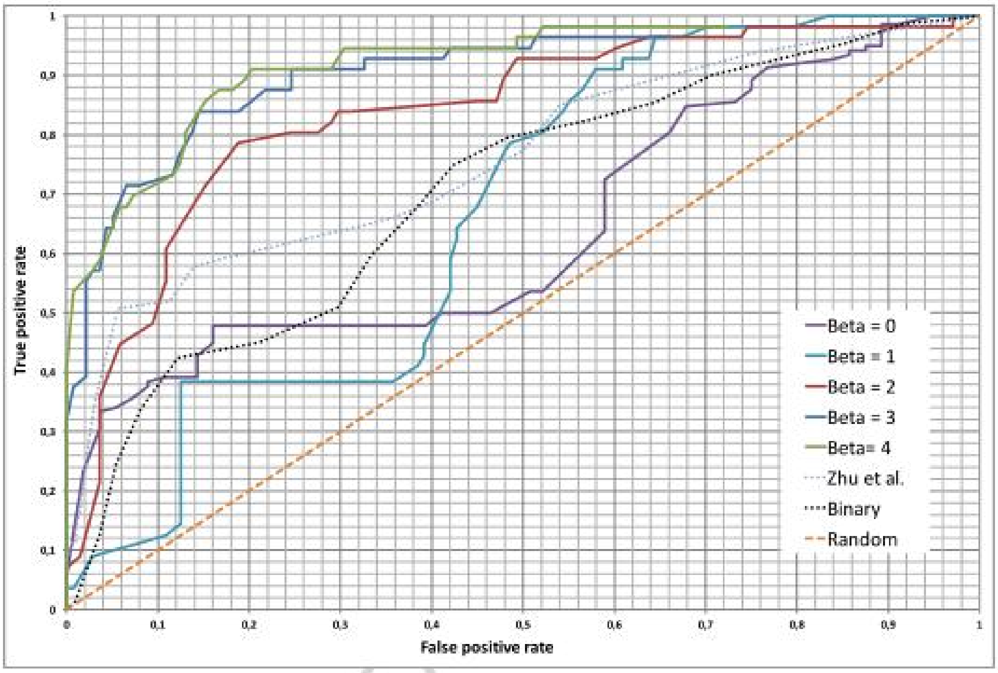

Figure 9: Performance for risky application detection in `Photography' category. ROC curves for four different β (denoted Beta on the graph) and thresholds for the equation 7.

This work can be continued by introducing an application recommendation system that not only takes into account application ratings, but also privacy, using a normal permissions request pattern. The proposed methodology allows us to measure risk of an application using two axes: number of permissions and pattern likeness. Thereby, the future recommendation system could propose applications that are similar to the one chosen by the user, but having less risk. This can be rather an application requiring less permissions for similar likeness, an application having a maximum likeness for the same number of permissions or a balanced recommendation.

5- Most important publication for this work