Android Permission Usage: a First Step towards Detecting Abusive Applications

Karina Sokolova*, Marc Lemercier*

*University of Technology of Troyes
Troyes, France
email:{karina.sokolova, marc.lemercier}@utt.fr

Charles Perez†

†PSB Paris School of Business
Paris, France
email:cperez@faculty-esgms.fr

***Abstract*—Thousands of mobile applications are available on mobile markets and actively used everyday. One of the mobile market leaders – Android – does not verify the security of applications published on its market and assumes that users will carefully judge the applications themselves using the information available on the marketplace. A common assumption is that the list of permissions associated with each application provides users with security and privacy indications, but previous works have shown that users are barely able to understand and analyse those permission lists. Very few works propose solutions that could help users in deciding whether or not to install an application. Despite Android permissions' lack of user-friendliness, they are an important source of information. In this work, we analyse permissions used by a large set of applications for different Android market categories and define the core permission patterns characterising each one. The patterns obtained are a first step towards building an indicator for detecting normal and possibly over-privileged applications on the market.**

Keywords–patterns, usage, mobile applications, Android, Google Play, permissions, Network science, graph analysis, data mining, category

I. INTRODUCTION

Today the mobile market is constantly growing; an increasing number of mobile applications are made available to users every day. Application distribution and security methods differ from one company to another, but the permission system – an explicative additional privilege for each sensitive service or piece of data – is often a common factor. Two mobile market leaders – Android and iOS – use a different approach to permission systems: iOS proposes a very limited number of permissions and gives users control over granting or revoking a single permission; Android proposes very large number of permissions, and users have to accept all of them at once before installing an application. iOS applications are checked for malicious or abusive codes before they are made available on the market. Android applications are uploaded directly on the market and it is up to users to judge each application using data available on the market, such as permission lists, comments and ratings provided by other users.

Some research has shown that users are often incapable of judging a mobile application's legitimacy simply by looking at the permissions requested by it prior to installation [1][2]. Android permissions are often very specific and contain technical terms. Some permissions are so widely used that users do not even pay attention to them when viewing the list. The result is that users often have not grasped the meaning and accept the list regardless of its permissions. This leads to important privacy and security issues that can be exploited by applications abusively requiring permissions. It is important to note that mobile applications can be intrusive without necessarily being malicious; some may want to collect users' information to 'unfairly' improve their customer relationship management (CRM).

In this paper, we aim to identify normal application permission usage patterns by application category. The Android market groups similar applications with similar ends into categories. Different functionalities require different data and services, which in turn imply different permissions. Our hypothesis is that categories on the market containing applications with similar functionalities will also require similar groups of permissions. We identify central and core permissions for categories and discuss related functionalities. We believe that such permission patterns can help create a measurement that allows users to compare easily more and less intrusive applications.

The remainder of the paper is organized as follows: Section 2 presents background on Android; Section 3 presents related works; the methodology is presented in Section 4; and Section 5 presents the results. The paper ends with a discussion about future works and a conclusion.

II. BACKGROUND

Android is an open-source operating system owned by Google. Since 2010, it has been a leader on the mobile market and used widely on smartphones, tablets and, more recently, on smart objects.

Android applications are available to users via the market store – GooglePlay. Google does not verify Android applications when they arrive on the market, and users should carefully check all available information to judge if an application is trustworthy and can be installed.

To help users evaluate applications, Android embeds a permission system security mechanism. Applications have very limited rights when accessing system services, sensitive data or sensors; therefore, developers must explicitly add permissions for each protected interface into a compulsory file called 'AndroidManifest'. By doing so, each application is associated with a list of permissions.

Android has a predefined list of permissions that developers can use. According to our analyses, Android 4.4 currently (November 2014) contains 229 permissions: 30 normal, 48 dangerous, 11 development, 70 signature and 70 signature or

 ISBN: 978-1-61208-393-3

system permissions from which third-party applications may only use 89 Android permissions.

Native Android permissions have a similar prefix based on a predefined hierarchy: 'android.permissions.*'. For example, the permission for Internet access is defined as 'android.permission.INTERNET'. Only five Android permissions are prefixed with 'com.android.*': browser, alarm, launcher and voicemail-related permissions.

Android allows communication and data/functionalities to be shared between applications. Developers can define custom permissions to protect new interfaces in order to share services or data. Custom permissions can be named freely: Google does not impose any naming rules.

The Android permission system and permission usage are a valuable source of information. The next section presents the works related to permission analyses and their limitations.

III. Related works

Very few works propose solutions to help users judge whether a given application abusively uses permissions and represents a potential threat.

Some authors propose monitoring the data flow of Android applications and reporting permission usage to users [3][4][5]. These solutions are not proactive, as the application must already be installed, and it is not clear that users could easily adopt this solution.

The authors of [6] proposed searching for a justification for permission usage in an application description using natural language processing (NLP) techniques, warning the user if this is not found. A proof of concept was carried out on three Android permissions. Further work improved the detection and number of supported permissions [7].

In [8], the authors created a crowd-sourcing system that collects and analyses iOS application configurations to provide users with privacy recommendations. This approach could be applied to Android applications; however, currently, the revocation of an individual permission is not possible with Android – users do not have any control over the permission list of an installed application. Moreover, Android's permission list is much longer and more technical than that of iOS, which makes it very laborious for a user to configure. Furthermore, this would be an application control solution, not an application choice or judgment solution.

Several works have been done on Android permission analyses.

In 2009, the authors of [9] analysed the permissions of the top 50 free Android applications using a self-organizing map (SOM). The authors provide some statistics on permission usage, identifying a series of pairs of correlated permissions and providing correlations between permissions and categories.

In [10], the authors analysed the permissions of the top 100 Android applications and found that most permissions were used occasionally, in response to the action made on the graphical user interface (GUI). The paper highlighted that only 5% of applications legitimately required some of the permissions granted permanently.

The authors of [11] analysed the permissions of Android's most popular and novel applications from both official and non-official markets. The authors analysed the interdependency of the number of permissions and the application popularity, price, availability of the developer's website and availability of privacy policies. They also analysed the number of permissions for applications with similar names.

None of the previous works have focused on analysing patterns in permission usage. Moreover, current analysis has only been performed on a very limited number of applications.

Few tools have analysed Android permissions for security purposes, one of them being the *Kirin* tool, which analysed AndroidManifest files to identify dangerous permission combinations and flag potential malware before installation [12]. The related paper identified two dangerous permissions and seven combinations of permissions, which were added to *Kirin*'s installation privacy policy. *SCAndroid* went even further, using source code analysis to identify if permissions were really used together by an application [13].

These latter works focused on the technical challenges related to embedding a permission-pattern-based tool in the Android system, but they did not discuss permission patterns directly. It is important to note that dangerous permission combinations were defined manually by the authors.

In [14], the authors used probabilistic methods to identify patterns for high- and low-ranked applications. The authors noted that pattern identifications by category would improve results.

Recent research [15] used statistical methods to identify the top 40 risky permissions and performed clustering techniques to identify patterns and detect malicious applications.

The authors of [16] analysed permission usage over a set of 1,227 clean and 49 malicious application families. The authors generated a list of permission patterns unique to Android malware but did not process the patterns of 'clean' or non-threatening applications. Due to calculation costs, this research only obtained patterns with a maximum of 4 permissions.

The authors of [17] used 999 Android applications to build a graph based on the co-occurrence of permissions in different application categories. The authors focused on determining in which cases approaches such as [12] could be applied to malware detection. The most frequent groups of permissions for each category were identified by a modularity-optimizing classification algorithm and were considered to be a normal request for an application from a given category. The authors compared these groups with dangerous permission combinations from [12] and found that some of the presumably risky combinations in groups in fact are legitimate. The authors noted that there was a bias in the analysis, namely that very popular permissions form important clusters in many different categories.

IV. Methodology

This section presents the methodology applied to obtain categories permission patterns.

 ISBN: 978-1-61208-393-3

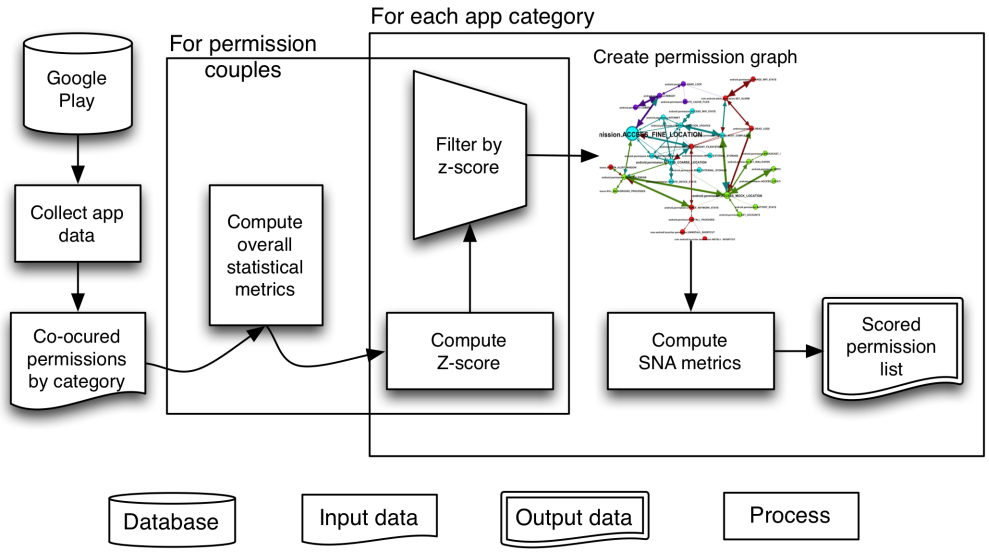

Figure 1. Methodology representation.

A. Overview

In this document, we aim to address some of the issues raised by previous works. Our objective was to analyse the permission usage of a large set of applications (not explicitly malicious) available on the official Android market. First, we identified significant patterns for the official Android market application categories. In order to do so, we performed a statistical and graph analysis that allowed patterns to be identified without limiting the number of items involved in a pattern. We built a methodology to avoid over-connecting the most popular permissions, only keeping track of the most significant patterns for each category. Finally, we assessed the graph obtained for further avenues of analysis.

A diagram of the methodology is shown in Figure 1. First, we compiled applications and related data from Google Play and prepared the data for processing. We evaluated pairs of permissions that co-occurred in each category. Then, we performed a statistical analysis and $\mathcal{Z} - score$ to determine the significance of each pair in each category. We filtered our dataset, keeping only the most significant permission pairs. Finally, we created a graph of the most significant permissions by category and used graph analysis to determine the importance of each individual permission. As an output, we obtained a scored list of the most significant permissions by category.

The following sections explained the dataset and the methodology in detail.

B. Dataset and initial observations

We compiled application data from the Google Play store using a publicly available non-official application programming interface (API) and a script written in PHP language published under the GNU General Public License [18]. We modified the script to match it to our objectives and stored the harvested data within a MySQL relational database.

We collected multiple types of information about applications available on Google Play: name, description, package name, version, users' note, number of downloads, price, category, number of screenshots, author and the list of permissions as defined in the manifest. For each category, we obtained the category's name and related description.

After launching a script to collect data, we obtained a sample of 9,512 applications related to 35 categories containing between 190 and 590 applications each. In our sample, we observed a set of 2,133 unique permissions, with 292 permissions identified as Android native permissions (263 matched the prefix 'android.permissions.*' and 29 matched the prefix 'com.android.*'). The other permissions are assumed to be custom permissions.

We compared the list obtained with a list of permissions extracted from Android 4.4 and found 157 permissions that did not match currently available Android permissions. These permissions were instead third-party application permissions, such as those for mobile device management (e.g., 'android.permission.sec.*'), old permissions from previous Android versions (e.g., 'com.android.launcher.permission.READ_SETTINGS'), permissions for Android in-app payment and licence libraries, and many misspelled permissions.

To carry out further analysis, we filtered our dataset to only keep permissions available in the Android 4.4 system.

Table I shows the top 10 permissions and the percentage of applications requiring these permissions. The INTERNET permission is the most required, as observed in previous works.

In the next section, we present the methodology we applied to obtain relevant patterns from our dataset.

C. Analyses

Our proposal was to analyse significant co-occurrences of permission pairs for each application category. To do so, for each category $\mathcal{C}$ we created a graph denoted $G_{\mathcal{C}}(N_C, E_C)$, where the set of nodes $N_{\mathcal{C}}$ corresponded to the permissions, and the set of edges E_C represented two commonly used

permissions in the category. This common usage was identified if both permissions were observed together in the Android-Manifest file of at least one application in the category.

It is important to note that although some permission pairs may have been used jointly in many categories, they were not necessarily relevant to it [9]. For example, INTERNET and ACCESS_NETWORK_STATE were used commonly in many categories but were not relevant when trying to create a permission fingerprint for a category. For this reason, the significance of the usage of a permission pair in a category had to be moderated by (1) the average use of this pair across categories and (2) weighted with respect to how regularly it appeared across multiple categories.

To quantify these observations, which have also been pointed out in previous works [17], we proposed scoring the weight between a permission pair $\mathcal{A} : (perm_i, perm_j)$ in a category $\mathcal{C}$ with the standard score or $\mathcal{Z} - score$ defined in equation 1.

$$\mathcal{Z}_{\mathcal{A}}^{\mathcal{C}} = \frac{\mathcal{A}_{\mathcal{C}} - \mu_{\mathcal{A}}}{\sigma_{\mathcal{A}}} \tag{1}$$

Where :

$\mathcal{A}$ is a permission pair $(perm_i, perm_j)$

$\mu_{\mathcal{A}}$ and $\sigma_{\mathcal{A}}$ are the mean and standard deviation of the usage of pair $\mathcal{A}$ across all categories.

$\mathcal{A}_{\mathcal{C}}$ is the observed usage of the pair in the category $\mathcal{C}$.

From equation 1, we propose defining the weight between two permissions $e_{\mathcal{C}}(perm_i, perm_j)$ for each $E_{\mathcal{C}}$ in graph $G_{\mathcal{C}}(N_{\mathcal{C}}, E_{\mathcal{C}})$ corresponding to a specific category $\mathcal{C}$ as follows:

$$e_{\mathcal{C}}(perm_i, perm_j) = \mathcal{Z}^{\mathcal{C}}{}_{(perm_i, perm_j)} \tag{2}$$

We obtained a graph of permissions with weighted relations for each category. We then filtered the edges of each graph by weight to highlight only the most significant patterns. We have removed edges whose $\mathcal{Z}-score$ stands below threshold 2. This threshold, for a normally distributed population, allows only 2,3% of the most relevant edges to be kept track of. Finally, we filtered the nodes, keeping only nodes with a non-null degree.

TABLE I
TOP 10 OF PERMISSIONS USAGE. EACH PERMISSION IS PREFIXED ORIGINALLY WITH 'ANDROID.PERMISSION".

Permission	Applications (%)
INTERNET	91,88
ACCESS_NETWORK_STATE	83,31
WRITE_EXTERNAL_STORAGE	60,39
READ_EXTERNAL_STORAGE	60,29
READ_PHONE_STATE	49,92
WAKE_LOCK	33,01
ACCESS_WIFI_STATE	31,47
VIBRATE	30,07
ACCESS_COARSE_LOCATION	27,73
ACCESS_FINE_LOCATION	27,42

Afterwards, we computed several graph metrics and algorithms in order to highlight the patterns for each category graph.

The first step was to compute a weighted modularity-based clustering algorithm to highlight potential functionalities represented by a common permission usage [19][20]. The modularity regrouped the graph's elements into communities. This score increased as the number of edges within communities increased and the number of edges between these communities decreased. The clustering algorithm used greedy optimisation to build communities in a way that maximized the modularity score.

Secondly, the betweenness centrality was computed on the permission graph in order to detect the most crucial permissions for an application in a category. In the domain of social network analysis, the betweenness centrality of a node v is measured as the ratio of the number of shortest paths between any node pairs (s, t) that pass throughout v by the number of shortest paths between these pairs. Mathematically, it is defined as stated in equation 3 below [21].

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3}$$

Where :

σ_{st} is the number of shortest path between two nodes s and t

$\sigma_{st}(v)$ is the number of shortest paths between two nodes s and t that pass through v.

The betweenness centrality measured the capacity of a permission to belong to many of the shortest paths. A high betweenness centrality indicated that this permission was required to perform multiple tasks or for the main functionality of applications in the category. More social network analysis measures (degree, closeness centrality, PageRank, etc.) were tested, but they are not discussed in this paper.

In the next section, we present the results and patterns obtained from this analysis.

V. RESULTS

We present below a set of results obtained from the analysis of permissions by categories using our dataset.

A. Number of relevant pairs by category

Relevant patterns formed by relevant pairs exist for each application category. Table II displays the number of permissions pairs $(perm_i, perm_j)$ named as relevant pairs that were statistically significant (w.r.t. $\mathcal{Z}-score > 2$) for each category.

We could observe that six categories covered a very large set of relevant pairs (up to 1,000). This showed that these categories were very broad and covered many different types of functionalities. An abusive application belonging to these categories could be harder to detect using the permission list than abusive applications belonging to a category that exhibited a more reasonable set of relevant pairs. We noted that two

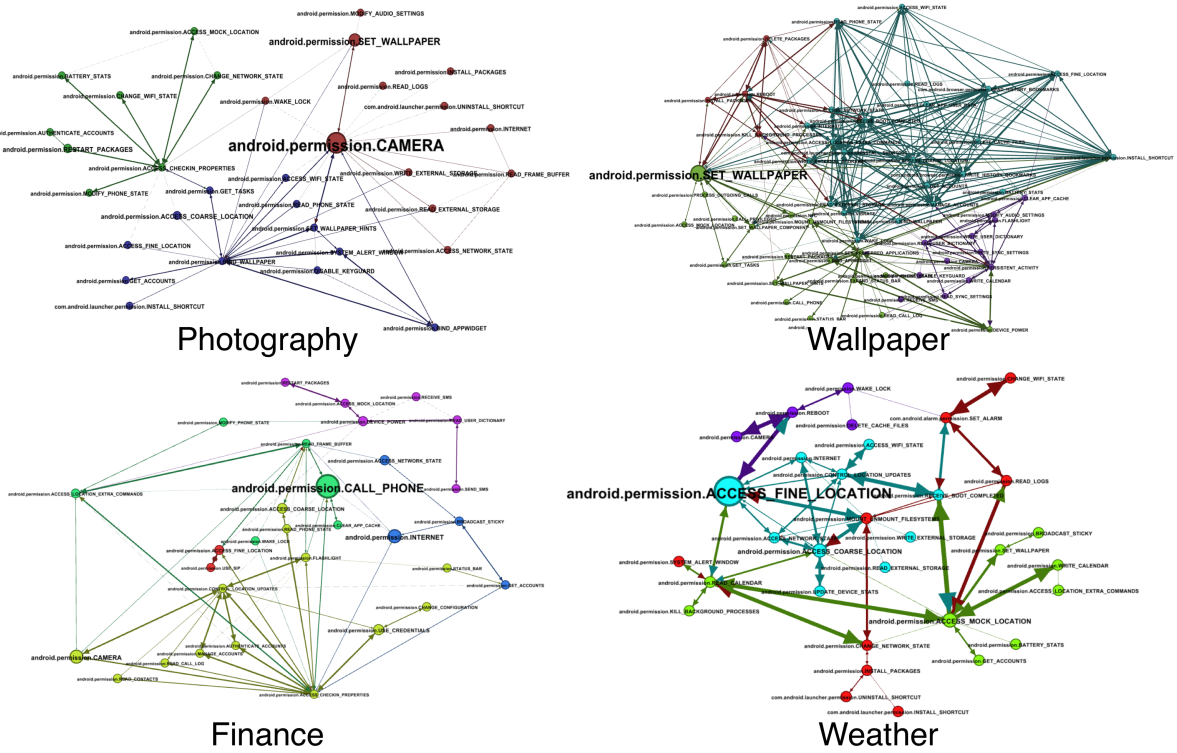

Figure 2. Permissions' graphs obtained for the categories Photography, Wallpaper, Finance and Weather.

categories possessed less than 20 significant pairs; these were very specific.

B. Significant patterns and centrality results for a category samples

Due to page number restrictions, we present the results related to only four of the categories – Photography, Wallpaper, Finance and Weather. We have chosen categories with an average number of relevant pairs. Corresponding patterns are highlighted in Figure 2. On each graph, the colour of the nodes is defined by the result of the modularity-based clustering algorithm. The weight of the link corresponds to the $\mathcal{Z}-score$, and the size of the nodes is proportional to the betweenness centrality. Graphs and data for all categories are available on [22].

Photography: As one would assume, 'CAMERA' is a central permission within the photography category. It is used with the 'READ_EXTERNAL_STORAGE' and 'WRITE_EXTERNAL_STORAGE' permissions, which allow photos taken to be saved and modified. The 'ACCESS_NETWORK_STATE', 'ACCESS_WIFI_STATE' and 'INTERNET' permissions enable photo sharing.

'SET_WALLPAPER' is the second most important permission, which allows the photo to be added as wallpaper on the main screen. We can distinguish a pattern grouping together wallpaper management permissions: 'SET_WALLPAPER_HINTS' and 'BING_WALLPAPER'.

'WAKE LOCK' prevents the screen from locking when the application is in use. This functionality seems relevant in camera-related applications.

Many applications in this category allow screenshots to be taken as well as photos. Shortcut management permissions allowing the creation of shortcuts can be used to take photos as well as for screenshots.

The presence of location-related permissions indicates that this information will be attached to the picture taken. The 'GET_ACCOUNTS' permission corresponds to a server-based user-specific service which probably backs up the photos taken on the server or shares photos with different services, such as social networks.

We noted an increased presence of system permissions that are not available for third-party applications. This indicates that many photography applications are built-in.

Wallpaper: The results for the Wallpaper category (APP_WALLPAPER) give a high number of significant permissions due to the diversity of animated wallpapers and the functionalities accessed and provided by animated wallpapers.

'SET_WALLPAPER'is the most central permission; we also find the wallpaper-related permissions in the pattern. File system and package management permissions can be observed, due to the different personalisation options proposed by a single wallpaper application, as well as shortcut and widget management permissions. We find many functionality-related permissions due to the different built-in functionalities: phone calls, SMS, calendar, settings, application list, contacts, bookmarks, cache – those functionalities are often included as a widget or fast access to wallpaper. External storage permissions allow personalisation images to be stored locally, and network-related permissions allow additional information such as weather to be obtained or new images downloaded. We also

TABLE II
NUMBER OF RELEVANT COUPLES OF PERMISSIONS ($\mathcal{Z} - score > 2$) FOR EACH CATEGORY OF APPLICATION

Category	# of relevant couples	Category	# of relevant couples
COMMUNICATION	3,620	WEATHER	124
TOOLS	2,826	PHOTOGRAPHY	116
APP_WIDGETS	2,318	ARCADE	106
PRODUCTIVITY	2,306	MEDIA_AND_VIDEO	92
BUSINESS	1,028	CASUAL	84
PERSONALIZATION	1,024	RACING	84
LIFESTYLE	738	SPORTS_GAMES	80
SOCIAL	738	SPORTS	78
APP_WALLPAPER	548	TRANSPORTATION	74
TRAVEL_AND_LOCAL	322	SHOPPING	66
ENTERTAINMENT	242	COMICS	64
MEDICAL	160	BOOKS_AND_REFERENCE	60
HEALTH_AND_FITNESS	158	CARDS	48
LIBRARIES_AND_DEMO	152	GAME_WIDGETS	46
MUSIC_AND_AUDIO	144	NEWS_AND_MAGAZINES	20
FINANCE	126	GAME_WALLPAPER	18

identify the 'WAKE_LOCK', 'ACCESS_FINE_LOCATION' and 'VIBRATE' permissions in this category.

Finance: The most central permissions for the Finance category are 'CALL_PHONE' and 'INTERNET'. Permissions used for calls, including Voice over IP (VoIP) calls and SMS, available to contact a bank or service manager. The 'INTERNET' permission would appear to be necessary in order to access up-to-date banking information. We can distinguish many account- and authentication-linked permissions due to the sensitivity of the financial information and the need for secure usage. Localisation permissions also appear in the pattern, probably to apply different location-dependent billing criteria or to identify the nearest offline office. The camera is often used for QR codes and making deposits in finance applications.

Weather: The central weather permissions is 'ACCESS_FINE_LOCATION', which gives the longitude and latitude so that the weather in the user's location can be obtained. All location- and network-related permissions are included in the pattern. 'ACCESS_MOCK_LOCATION' could indicate developer testing or be for locations given by the user. One can also see background process, shortcut and wallpaper permissions, which indicate that the weather application can be wallpaper-embedded. Permissions related to external storage are needed for heavy image storage. Weather applications are often system applications, and some system permissions are observed in the pattern.

VI. DISCUSSION AND FUTURE WORK

The state of the art's most commonly used permission indicator is the simple occurrence of permissions. To underline how our methodology has improved this, we proposed comparing our results to the top 5 most frequent permissions obtained for the same category.

We present the 'Finance' category as an example. Table III presents the top 5 permissions according to occurrence, and Table IV presents the top 5 permissions according to betweenness centrality. One can see that the top 5 'Finance' permissions of Table III correspond to the top 5 permissions for all categories presented in the Table I. This shows that even if those permissions are highly used in the 'Finance' category, they are not specific to it.

TABLE III
TOP 5 FREQUENT PERMISSIONS FOR FINANCE CATEGORY

	Occurrence (%)	Betweenness
INTERNET	91.19 (Rank 1)	361 (Rank 2)
ACCESS_NETWORK _STATE	75.15 (Rank 2)	202 (Rank 6)
WRITE_EXTERNAL _STORAGE	49.32 (Rank 3)	98 (Rank 16)
READ_EXTERNAL _STORAGE	49.12 (Rank 4)	98 (Rank 17)
READ_PHONE _STATE	32.88 (Rank 5)	69 (Rank 19)

TABLE IV
TOP 5 PERMISSIONS ACCORDING TO BETWEENNESS CENTRALITY FOR FINANCE CATEGORY

	Occurrence (%)	Betweenness
CALL_PHONE	11.74 (Rank 12)	470 (Rank 1)
INTERNET	91.19 (Rank 1)	361 (Rank 2)
CAMERA	10.96 (Rank 14)	360 (Rank 3)
USE_CREDENTIALS	3.52 (Rank 21)	257 (Rank 4)
ACCESS_COARSE _LOCATION	19.18 (Rank 8)	231 (Rank 5)

We noted that our pattern contains these permissions, but not as highly ranked; the top five permissions from our results (Table IV) show that 'Finance' permissions are often online (Internet) services and need secure authentication (use credentials). Banking applications tend to include direct bank-application contact (call phone), deposits (camera) and lists of office or cash withdrawal locations (access coarse location).

Our pattern is more accurate than simple frequency analysis

 ISBN: 978-1-61208-393-3

in defining a particular category and allows category-related functionalities to be detected. The use of the $\mathcal{Z} - score$ is particularly well-adapted to this purpose, since it allows how relevant a permission pair is to a category to be measured with respect to overall usage in units of standard deviation.

We observed many wrong, misspelled or old permissions in the applications. We feel that the system would benefit from automatic permission validity verification based on the list of valid Android permissions; rules for defining custom permissions could simplify the verification. Documentation for the Android permission system is incomplete, as many Android 4.4 extracted permissions were not found on the official website.

When we observed permission patterns by category, they often represented a particular functionality. This could permit the purpose of permission usage and the functionalities of an application to be determined automatically.

Some categories obtained a very large number of significant permissions, which means they may have been too broad. The division of these categories into subcategories would provide a more precise view of the applications.

Our patterns could permit an automatic classification of applications into categories and could also be used to measure how an application rates with regard to normal permission usage in a particular category. Applications using non-core or rare permissions can be penalised. Such indicators could be included in mobile markets to label abusive or non-threatening applications, comparing them to expected patterns.

VII. Conclusion

We analysed Android permission usage for each application category belonging to the Google Play store. We proposed a graph-based solution to characterise each category using patterns of the most significant permissions, taking into account the category and the overall usage of each permission combination. We scored permissions using betweenness centrality in order to obtain the most- and least- central permissions for each category. The identified patterns and permission scores could be used in the mobile market to detect abusive or non-threatening applications.

References

[1] P. Kelley et al., "A conundrum of permissions: Installing applications on an android smartphone," in Financial Cryptography and Data Security, ser. Lecture Notes in Computer Science, J. Blyth, S. Dietrich, and L. Camp, Eds. Springer Berlin Heidelberg, 2012, vol. 7398, pp. 68–79.

[2] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in Proceedings of the Eighth Symposium on Usable Privacy and Security, ser. SOUPS '12. New York, NY, USA: ACM, 2012, pp. 3:1–3:14.

[3] 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, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[4] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications," in CCS '11: Proceedings of the 18th ACM conference on Computer and communications security. ACM Request Permissions, Oct. 2011.

[5] P. Berthomé and J.-F. Lalande, "Comment ajouter de la privacy after design pour les applications Android? (How to add privacy after design to Android applications?)," Jun. 2012.

[6] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "Whyper: Towards automating risk assessment of mobile applications," in Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). Washington, D.C.: USENIX, 2013, pp. 527–542.

[7] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, "Autocog: Measuring the description-to-permission fidelity in android applications," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 1354–1365.

[8] Y. Agarwal and M. Hall, "Protectmyprivacy: Detecting and mitigating privacy leaks on ios devices using crowdsourcing," in Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 97–110.

[9] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in Proceedings of the 17th ACM Conference on Computer and Communications Security, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 73–84.

[10] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan, "User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems," Security and Privacy (SP), 2012 IEEE Symposium on, 2012, pp. 224–238.

[11] P. H. Chia, Y. Yamamoto, and N. Asokan, "Is this app safe?: A large scale study on application permissions and risk signals," in Proceedings of the 21st International Conference on World Wide Web, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 311–320.

[12] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in Proceedings of the 16th ACM Conference on Computer and Communications Security, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 235–245.

[13] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "SCanDroid: Automated Security Certification of Android Applications," Department of Computer Science, University of Maryland, College Park, Tech. Rep. CS-TR-4991, November 2009.

[14] M. Frank, B. Dong, A. P. Felt, and D. Song, "Mining Permission Request Patterns from Android and Facebook Applications," in Data Mining (ICDM), 2012 IEEE 12th International Conference on, 2012, pp. 870–875.

[15] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection," Information Forensics and Security, IEEE Transactions on, vol. 9, no. 11, 2014, pp. 1869–1882.

[16] V. Moonsamy, J. Rong, S. Liu, G. Li, and L. Batten, "Contrasting Permission Patterns between Clean and Malicious Android Applications," in Future Generation Computer Systems. Cham: Springer International Publishing, 2013, pp. 69–85.

[17] I. Rassameeroj and Y. Tanahashi, "Various approaches in analyzing Android applications with its permission-based security models," in Electro/Information Technology (EIT), 2011 IEEE International Conference on, 2011, pp. 1–6.

[18] O. V. Koc, "android-market-api-php." [Online]. Available: https://github.com/splitfeed/android-market-api-php[accessed:13-01-2015]

[19] H. Shiokawa, Y. Fujiwara, and M. Onizuka, "Fast algorithm for modularity-based graph clustering." in AAAI, M. desJardins and M. L. Littman, Eds. AAAI Press, 2013.

[20] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," Physical Review E, vol. 69, no. 6, 2004, pp. 066 133–066 133.

[21] L. C. Freeman, "Centrality in social networks conceptual clarification," Social Networks, vol. 1, no. 3, 1978, pp. 215–239.

[22] C. Perez and K. Sokolova, "Android permissions usage data." [Online]. Available: https://sites.google.com/site/androidpermissionsanalysis/[accessed:14-01-2015]

 ISBN: 978-1-61208-393-3