PUREDroid: Permission Usage and Risk Estimation for Android Applications

Ali Alshehri
Oakland University
Rochester, MI 48309 USA
aaalshehri@oakland.edu

Pawel Marcinek
Oakland University
Rochester, MI 48309 USA
pbmarcin@oakland.edu

Abdulrahman Alzahrani
Oakland University
Rochester, MI 48309 USA
aalzahrani@oakland.edu

Hani Alshahrani
Najran University
Najran, Saudi Arabia
hmalshahrani@nu.edu.sa

Huirong Fu
Oakland University
Rochester, MI 48309 USA
fu@oakland.edu

ABSTRACT
Android applications pose many security risks that affect the security and privacy of their users. Adversaries construct different types' Android applications pose many security risks that affect the security and privacy of their users. Adversaries construct different types of malicious applications and use different social engineering approaches to attract users to download and trust these applications. Malicious applications usually request permissions that are not related to their main functionality in order to access sensitive information or resources. Most of users attempt to grant the requested permissions without understanding the potential harm of those applications and how the requested permissions can be misused to disclose their privacy. Therefore, there is a need for a risk assessment model which can intimate the users about the risk level of permissions requested by an application in order to assist users to make the right decision whether to grant or deny a requested permission. This paper proposes **P**ermission **U**sage and **R**isk **E**stimation for An**droid** (PUREDroid) to measure the security risk of Android applications' permissions and the magnitude of harm resulting from granting extraneous permissions requests. In an evaluation with more than 25000 applications, including 5773 malware applications and 19242 benign applications, we demonstrate the usefulness and the effectiveness of our proposed scoring method.

CCS Concepts
• **Security and privacy** → Malware and its mitigation.

Keywords
Android, permission model, analysis, privacy, risk score

1. INTRODUCTION
Among different smartphone platforms, Android has become the most prevalent and fastest growing platform [16]. Android devices, of which millions are in use today, are attractive target for adversaries. Therefore, a worth considerable effort is ensuring the Android operating system security, which has imposed a significant interest among researchers to alleviate its threats. Built-in security mechanisms in Android operating system (OS) provide various levels of protection for data and applications. For instance, application security is governed by a list of permissions used to permit an application to communicate with public resources (e.g. camera and Bluetooth) or highly privileged APIs (e.g. read or send SMS).

It is essential for an application to access sensitive information in order to operate properly. Thus, during the application development stage, required permissions must be declared. However, Android permission model does not have specific gauges that developers, who even are not fully knowledgeable about permissions, have to follow when they declare their applications' permissions [17]. Hence, most of Android applications developers requesting multiple permissions that might be unnecessary for the main functionality of their applications. However, most users grant the permissions requested by the applications without understanding the potential harm of those applications and how the requested permissions can be misused to disclose their privacy [7].

Furthermore, the Android permission model does not impede privilege escalation or information leakage. In other words, the permission model is not fine-grained enough to provide sufficient means of control on an application's activities and specify what private information or resources are accessible to the application [8]. Thus, many attacks could possibly occur by exploiting critical permissions, such as privilege escalation sabotages [14].

These challenges are important because there is no standard of which permissions should be requested by which application. Therefore, applications' developers can request any permission even if it is not related to the application's main functionality. Unscrupulous developers are fully aware of the Android permissions model's vulnerabilities that they can exploit to gain unauthorized access or abuse sensitive resources [11]. This could impose costs on Android users' privacy if a malicious application uses an overly broad permission to perform inappropriate operations.

Android applications in the marketplaces are classified based on their functionality, referred as category. Applications within the same category should share common features and seek similar intents in general. For instance, requested permissions within applications from the same category are most likely consistent [10]. However, malicious applications perform deviant behavior that is far from what they claim and could request permissions that are not related to their main functionality. Thus, this paper proposes a new assessment method to measure the risk of requested permissions within an application based on the application category. By

ICISDM'19, April 06–08, 2019, Houston, TX, USA.

DOI: http://dx.doi.org/ 10.1145/3325917.3325941

constructing an optimal set of permissions for each category, each permission within an application from the same category will be assigned into one of three security risk regions; low, moderate, or high risk score. Also, this paper argues for the need of Android permission model readjustment to lock down unauthorized operations. Briefly, this paper makes the following contributions:

- Analyzing a dataset of Android applications and investing-ting the usage of Android permissions requested by benign and malicious applications.
- Introducing a new scoring method based on the application category to evaluate the security risks of permissions within an application.
- Conducting extensive evaluation to prove the usefulness and the effectiveness of our scoring method.

The reminder of the paper is organized as follows: Section 2 discusses the background of Android OS. Then, Section 3 provides an overview of the related work. Next, Section 4 explains our methodology and approach in-depth. After that, Section 5 discusses the evaluation and results of our model. Finally, Section 6 concludes our paper and presents our future work.

2. BACKGROUND

The security of Android system was designed with multiple layers to deliver flexibility, especially for the developers to build secure applications, and sufficient protection for all users of the platform. Android provides some security mechanisms to protect users, such as application sandbox and permission model [13].

Each application runs in an isolated sandbox. Any application wishes to communicate with another application or access any sensitive resources must have the appropriate permission. Hence, permission model is used to protect the communication between applications and the Android OS and restrict access to sensitive information, which would be otherwise compromised [19]. Applications' developers have to declare all permissions they need for the application in the AndroidMainfest.xml file to be granted by the user [4].

In addition, run time permissions were introduced to allow applications to request permissions as needed at run time. However, the permission model can grant an over-privileged permission to an application because a single permission handles a varying set of APIs. For example, if READ_PHONE_STATE permission is granted to an application, the application can obtain varying phone information, such IMEI, SIM card serial number, Sim card operator, and device ID [3]. Consequently, applications can access more resources than what they need and the user cannot approve a part of this permission.

3. RELATED WORK

In this section, variety of research attempts are discussed, which have proposed the concept of risk scoring to evaluate Android applications and permissions. The pattern of application's permission request has been studied to generate risk indication to improve users' awareness for the potential malicious activities. For instance, Gates et al. [9] introduced a new method to generate risk signals based on the permissions requested by an application to provide users with feedback about why the application is risky. The risk score is measured using different Bayesian probabilistic models to satisfy three goals, namely, monotonicity, coherence, and ease. Based on their experimental results, the authors stated that using Naive Bayes with informative Priors (PNB) generates a better risk score comparing to the other Bayesian probabilistic models. However, the authors consider a limited set of Android permissions in their study. Another limitation is that the malware dataset used to evaluate their method is very small and the applications categories are unknown comparing to the used benign dataset, which can impact the reliability of their approach.

Deypir [5] proposed a new approach to measure the security risk score of Android permissions based on the concept of information theory and entropy. For each individual permission, the information gain is computed by calculating the entropy of the entire used dataset and the entropy of the permission within the dataset. In this approach, larger risk score is assigned to the permission that are more informative. However, the author neglected the applications categories of the analyzed applications, which negatively affect the risk score since applications from different categories represent different functionality. In addition, information gain has symmetric property, which can incorrectly assign a high risk score to a permission that is mostly used by benign applications.

Furthermore, Wang et al. [20] proposed DroidRisk, a quantitative method to compute the risk of Android applications and permissions. The risk score is calculated based on the potential impact as well as the type of the permission whether it is a normal permission or dangerous permission. The authors used a static weight to represent the impact of a permission based on the permission category, 1 for normal permission category and 1.5 for dangerous permission category. However, measuring the impact of a permission based on applied static weight is inadequate and could affect the validity of the risk score [12]. Similarly, Dini et al. [6] introduced MAETROID, a framework that evaluate the applications requested permissions along with its metadata. The permission risk score is computed by assigning a static weight to each requested permission based on the permission threat category. However, the authors did not provide any empirical evaluation and analysis of their proposed framework.

In addition, Shen et al. [15] introduced a risk assessment model to evaluate the risk of permissions requested by an application according to the application category. The risk score is measured by calculating the number of applications requesting a specific permission within the same category over the number of applications requesting the same permission for all categories. However, the calculation of a permission risk score depends on how frequent this permission requested by other categories, which have different functionality compared to the inspected category. Consequently, a permission could have a low risk score because it was frequently requested by other categories, however, it could be risky to be granted for an application in the inspected category since the functionality of this category does not require such permission to work properly.

To overcome these limitations, PUREDroid is proposed to evaluate the risk of permissions within an application based on the application category and provide the users with a risk region for each permission to help them making the right decision whether to grant or deny the requested permission.

4. PROPOSED RISK ASSESSMENT MODEL

The goal of this research is to introduce a reliable method to measure the security risk of Android permissions requested by applications and the magnitude of harm resulting from granting extraneous permissions. This method can be utilized to evaluate the risk of each permission based on a constructed optimal set of permissions for each application's category. This section describes the proposed PUREDroid, a risk assessment model which is essential to reveal the privacy risk of an application based on its re-

quested permissions.

In order to measure the risk score of a permission in PUREDroid, the non-requested and requested permissions of applications within the same category are represented by two orthonormal states vectors $|0\rangle$ and $|1\rangle$, respectively, as follows:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{1}$$

Every permission is characterized by how many times is requested by both benign and malware applications and represented by two vectors, $|m\rangle$ for malware and $|n\rangle$ for benign as shown in Equation 2. The components of vectors $|m\rangle$ and $|n\rangle$ are depicted on Figure 1.

$$|m\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}, \quad |n\rangle = \gamma|0\rangle + \delta|1\rangle = \begin{pmatrix}\gamma\\\delta\end{pmatrix} \tag{2}$$

The coefficients α, β and γ, δ are the x and y components of vectors $|m\rangle$ and $|n\rangle$, respectively. They satisfy the normalization conditions as shown in Equation 3.

$$\alpha^2 + \beta^2 = 1, \quad \gamma^2 + \delta^2 = 1 \tag{3}$$

These coefficients are related to the occurrence probabilities described by the formulas in Equations 4. P_m is given by the ratio of the number of malware applications for a selected permission to the number of the entire malware dataset. Similarly, P_n is given the ratio of the number of benign applications for selected permission to the number of the entire benign dataset.

$$\begin{aligned} \alpha &= \sqrt{1 - P_m} \qquad \beta = \sqrt{P_m} \\ \gamma &= \sqrt{1 - P_n} \qquad \delta = \sqrt{P_n} \end{aligned} \tag{4}$$

The vectors $|m\rangle$ and $|n\rangle$ are used to construct a risk vector based on the following steps:

First, a two dimensional vector states are extended to three dimensional vectors with the z component equals to zero such that

$$|m\rangle = \begin{pmatrix}\alpha\\\beta\\0\end{pmatrix}, \quad |n\rangle = \begin{pmatrix}\gamma\\\delta\\0\end{pmatrix} \tag{5}$$

Second, the cross product $|n\rangle \times |m\rangle$ is computed where its i^{th} component is given by

$$(|n\rangle \times |m\rangle)_i = \epsilon_{ijk}|n\rangle_j|m\rangle_k \tag{6}$$

Where ϵ_{ijk} is commonly known as the Levi-Civita symbol and the Einstein summation convention is used. The indices $i, j, k \in \{1,2,3\}$ correspond to vector components. The only non-zero component of the cross product $|n\rangle \times |m\rangle$ is in z direction, which has the following form:

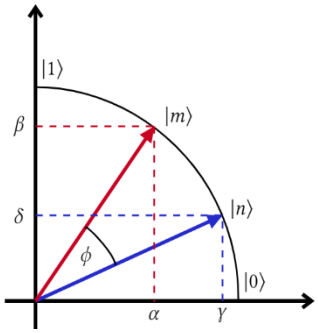

Figure 1. The geometrical representation of |n⟩ and |m⟩.

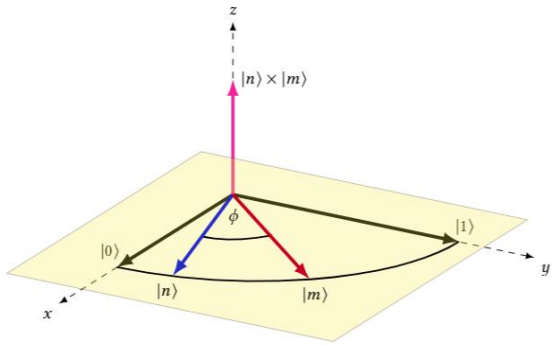

Figure 2. Cross product formed by |n⟩ and |m⟩ vectors.

$$\begin{aligned} (|n\rangle \times |m\rangle)_3 &= \epsilon_{3jk}|n\rangle_j|m\rangle_k \\ &= |n\rangle_1|m\rangle_2 - |n\rangle_2|m\rangle_1 \\ &= \gamma\beta - \alpha\delta \\ &= \sqrt{(1-P_n)P_m} - \sqrt{(1-P_m)P_n} \end{aligned} \tag{7}$$

The cross product $|n\rangle \times |m\rangle$ in positive direction is depicted on Figure 2.

Third, the z component of cross product can have both negative and positive values lies in the interval $[-1, 1]$. Therefore, the z component of $|n\rangle \times |m\rangle$ is re-scaled to the set of values $[0, 1]$ and is used as a scaled risk measurement function $r(P_m, P_n)$ described by

$$r(P_m, P_n) = \left(\frac{1+\sqrt{(1-P_n)P_m}-\sqrt{(1-P_m)P_n}}{2}\right) \tag{8}$$

One can see that when $P_m = 0$, (i.e, a permission is not requested by any malware) the risk function is reduced to the following form:

$$r(P_m = 0, P_n) = \left(\frac{1-\sqrt{P_n}}{2}\right) \tag{9}$$

It is obvious that the risk function should be zero in the case when $P_m = 0$ independently of P_n (permission requested by benign applications). Therefore, there is a need to introduce a proper weight function $w(P_n, P_m)$ defined as

$$w(P_n, P_m) = \frac{(P_m)^2}{(P_m)^2+(P_n)^2} \tag{10}$$

This leads to the expression for the weighted risk function R, which can be written in the following form:

$$R = w\left(\frac{1+\sqrt{(1-P_n)P_m}-\sqrt{(1-P_m)P_n}}{2}\right) \tag{11}$$

and if $P_m = 0$, the weighted risk function R is always *zero* for any $P_n > 0$, as indicated previously.

The algorithm pseudo code of PUREDroid is illustrated in Algorithm 1 and the symbols used are presented in Table 1. This algorithm calculates the risk score of all permissions requested by an application A by obtaining four parameters, which are the benign applications set S_B ,the malware applications set S_M ,the set of permissions S_P, and the inspected application A. Next, the usage of permission P_i is counted and accumulated in P_n for benign applications and P_m for malware applications, which are essentially needed to compute the risk score of the permissions. After that, the risk of each permission is gauged by computing the cross product $|n\rangle \times |m\rangle$ and re-scaled to avoid negative values. Then, the weight

is computed. Finally, the risk score *R* of the permission is computed by multiplying the computed weight *w* by the risk function *r*.

Algorithm 1 Computing permission risk score R

Data: N=$|S_B|$; M=$|S_M|$; R=0

begin

  foreach $permission(P_i) \in S_P$ **do**

    foreach $Application(A_i) \in S_B$ **do**

      $P_n = P_n + 1$

    end

    foreach $Application(A_i) \in S_M$ **do**

      $P_m = P_m + 1$

    end

    $|n\rangle \times |m\rangle = \sqrt{(1-P_n)P_m} - \sqrt{(1-P_m)P_n}$

    $r = 0.5 \cdot (1 + |n\rangle \times |m\rangle)$

    $w = \frac{(P_m)^2}{(P_m)^2+(P_n)^2}$

    $R = w \cdot r$

  end

  return R;

end

Table 1. Symbols used in pseudo code of Computing Permission Risk algorithm and their meaning

Symbol	Meaning
S_B	Set of benign applications
S_M	Set of malware applications
S_P	Set of permissions
A	Application
P_i	Permission
P_n	Number of benign applications use permission P_i
P_m	Number of malware applications use permission P_i

5. EVALUATION AND RESULTS

PUREDroid is evaluated on a dataset obtained from AndroZoo [2], which is the largest publicly available benign and malware dataset for Android. After persistent effort, more than 25000 applications, including 5773 malware applications and 19242 benign applications, from AndroZoo dataset were analyzed and labeled to evaluate our distinct model. Those applications belong to seven categories, namely, Lifestyle, Music & Audio, Books & Reference, Entertainment, Tools, Education, and Photography. These

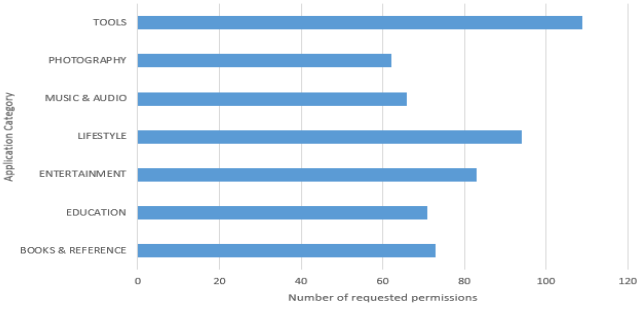

Figure 3. Number of requested permissions for the inspected applications categories.

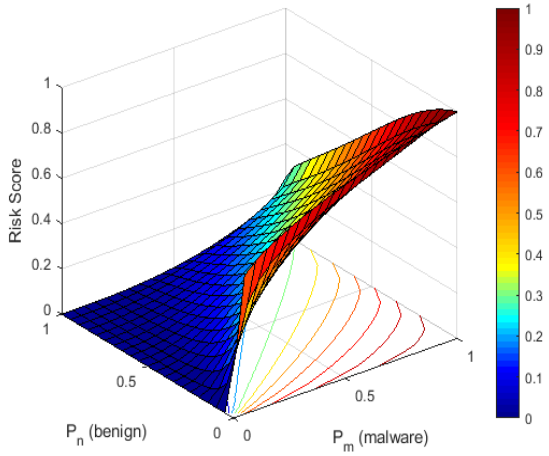

Figure 4. The surface of the risk function $R(P_n, P_m)$.

categories are the most targeted categories by malware according to a report from Symantec lab [18].

To extract all permissions within an application, each Android Application Package (*.apk*) of the inspected applications are analyzed through a static analysis technique. The static analysis is performed to disassemble and convert the *APK* contents to a readable format using apktool [1]. Once the *APK* is disassembled, the information about the .apk and its AndroidMainfest.xml file is retrieved. By examining all the requested permissions by all applications within the same category, it was obvious that every category has different set of permissions. Figure 3 illustrates the number of requested permissions for every category.

After analyzing the permissions, the risk function assigned the permissions with different scores based on the number of times a permission was requested by benign and malware applications as depicted in Figure 4. The risk score can belong to one of three regions, which are **L**ow risk region, **M**oderate risk region, or **H**igh risk region. These regions are introduced by the following cases:

- **Case1**: When $P_n = P_m$, $R = 0.25$. For any $P_m \leq P_n$, $R(P_m, P_n) \in [0,0.25]$ is considered as a low risk region.
- **Case2**: For $P_n = 0$ the limit behave as follows

$$\lim_{P_m \to 0} R(P_m, 0) = \frac{1}{2} \quad (12)$$

  and the risk function R is increasing as P_m is growing and for $P_m = 1$, $R_{max} = R(1,0) = 1$. Thus, all values of $R(P_m, P_n) \in (0.5,1]$ is considered as high risk region.

- **Case 3**: This case is the consequence of the boundary behave of the risk function R discussed in cases 1 and 2. Thus, for any value $R(P_m, P_n) \in (0.25,0.5]$ is considered as moderate risk region.

The effectiveness of our model was tested by computing the risk score of all permissions requested by the applications in the dataset. Due to the limited pages of the paper, only result and evaluation of Books & Reference category is discussed. Therefore, it is clearly seen from Figure 5 and Table 2 that PUREDroid assigned high risk scores to the permissions that are significantly requested by malware applications within Books & Reference category. On contrary, the permissions that are significantly requested by benign applications were assigned to lower risk scores. There are some

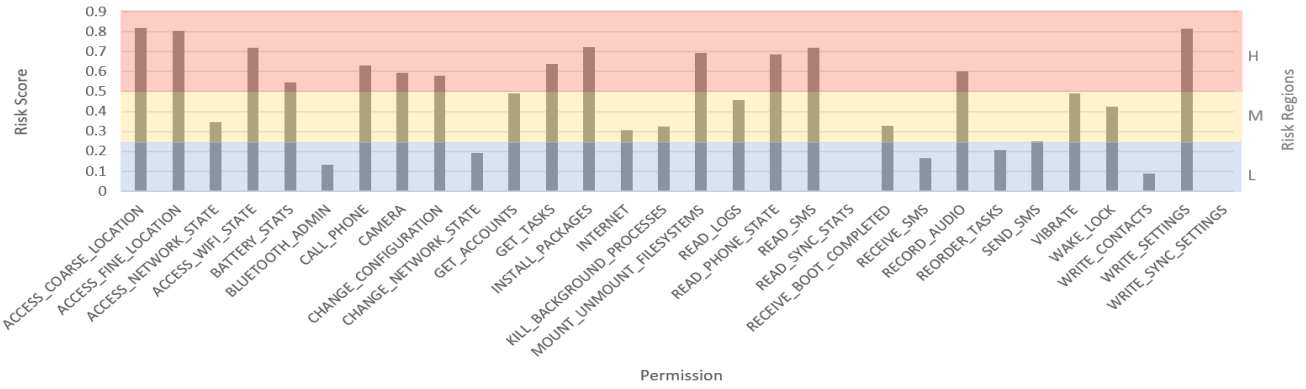

Figure 5. Example of the risk scores and regions of the permissions based on our dataset.

permissions that are assigned to moderate risk scores since there are a few differences in usage between benign and malware applications.

Most of the permissions, especially the ones frequently requested by malware applications, have potential security and privacy risk, such as personal information leakage or financial loss. As shown in Figure 6, malware applications usually request permissions that can potentially harm the user privacy, such as identifying the user location, accessing user messages, and changing the device setting. Granting those permissions can effectively give the application the power to access critical resources. Therefore, PUREDroid assigned the permissions that are frequently requested by malware applications and rarely by benign applications with high risk to avoid and mitigate such risks. This can help the user to understand the level of risk of the permissions requested and if these permissions have high risk regardless of the application, the user should deny them.

To fairly and accurately evaluate our model and show how it efficiently assigns high risk scores to permissions that are mostly requested by malware applications, PUREDroid was compared to a previously proposed work by [5]. For this evaluation, same dataset is used to evaluate both methods, which includes top downloaded 600 benign applications and 200 malware applications from the same category. As shown in Table 3, based on PUREDroid model, ACCESS_FINE_LOCATION was assigned to the highest risk score because it was frequently requested by malware applications. On the other hand, WAKE_LOCK was assigned to the lowest risk score since it was frequently requested by benign applications. Furthermore, using Deypir method [5], READ_PHONE_STATE and WAKE_LOCK were assigned to the same risk score. However, READ_PHONE_STATE was frequently requested by malware applications compared to WAKE_LOCK, which was frequently requested by benign applications. Practically, the risk score of READ_PHONE_STATE permission should not only be different from the risk score of WAKE_LOCK permission, but also much higher, which was not achieved by Deypir method [5] due to the symmetric property. In contrast, PUREDroid accurately assigned different risk score for the aforementioned permissions.

Table 2. Example of some of the permissions and their risk scores based on PUREDroid model

Permissions	P_n	P_m	Risk Score
ACCESS_COARSE_LOCATION	0.088	0.889	0.818
WRITE_SETTINGS	0.033	0.684	0.816
ACCESS_FINE_LOCATION	0.097	0.881	0.803
INSTALL_PACKAGES	0.001	0.234	0.721
READ_SMS	0.001	0.228	0.719
ACCESS_WIFI_STATE	0.204	0.935	0.717
READ_PHONE_STATE	0.270	0.971	0.686
MODIFY_AUDIO_SETTINGS	0.013	0.234	0.653
RECORD_AUDIO	0.027	0.241	0.602
CAMERA	0.029	0.241	0.595

Table 3. Comparison between PUREDroid and another model proposed by [5]

Permissions	Usage in		Risk score	
	benign	malwere	PUREDroid	Deypir
ACCESS_FINE_LOCATION	0.108	0.680	0.776	0.216
CAMERA	0.028	0.145	0.587	0.028
READ_PHONE_STATE	0.350	0.650	0.503	0.049
GET_ACCOUNTS	0.150	0.255	0.420	0.009
INTERNET	0.975	1.000	0.296	0.007
WAKE_LOCK	0.650	0.350	0.078	0.049

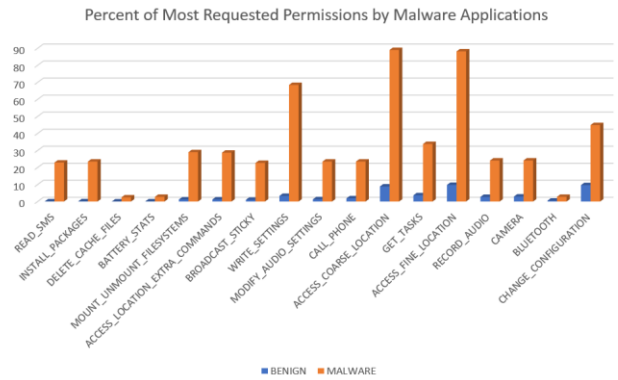

Figure 6. The most used permissions by malware in the datasets as a percent of apps that request those permissions.

6. CONCLUSION

Android applications pose many potential risks that may harm users' privacy. Malicious applications usually request permissions that are not related to their main functionality to access sensitive resources. Therefore, ordinary users grant permissions without understanding the potential harm that may occur upon using those applications. Therefore, it is imperative to assess the security risks of nonnotarized applications and the permissions that an application may request through a risk assessment model that can intimate the users about the potential threats an application may pose to users' privacy. This paper introduces PUREDroid, a risk assessment method that can be used to increase user's awareness of the risk involved with granting permissions to Android applications. This model has shown that permissions that perform suspicious activities are assigned to high risk scores unlike those of normal activities. In our future work, more features such as API calls and intents will be considered in our model to introduce a sophisticated model to measure the risk score of each application.

7. ACKNOWLEDGMENTS

This research work is partially supported by the National Science Foundation under Grants DGE-1623713. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] Apktool. 2018. https://github.com/iBotPeaches/Apktool.

[2] K. Allix, T. F. BissyandÃľ, J. Klein, and Y. L. Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). 468–471. https: //doi.org/10.1109/MSR.2016.056.

[3] Ali Alshehri, Anthony Hewins, Maria McCulley, Hani Alshahrani, Huirong Fu, and Ye Zhu. 2017. Risks behind Device Information Permissions in Android OS. Communications and Network 9, 04 (2017), 219.

[4] Android Developer. 2018. Request App Permissions. https://developer.android.com/training/permissions/requesting.

[5] Mahmood Deypir. 2018. Entropy-based security risk measurement for Android mobile applications. Soft Computing (04 Aug 2018). https://doi.org/10.1007/ s00500-018-3377-5.

[6] Gianluca Dini, Fabio Martinelli, Ilaria Matteucci, Marinella Petrocchi, Andrea Saracino, and Daniele Sgandurra. 2018. Risk analysis of Android applications: A user-centric solution. Future Generation Computer Systems 80 (2018), 505–518.

[7] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. 2012. Android Permissions: User Attention, Comprehension, and Behavior. In Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS '12). ACM, New York, NY, USA. https://doi.org/10.1145/2335356.2335360.

[8] Elli Fragkaki, Lujo Bauer, Limin Jia, and David Swasey. 2012. Modeling and enhancing androidâĂŹs permission system. In European Symposium on Research in Computer Security. Springer, 1–18.

[9] C. S. Gates, N. Li, H. Peng, B. Sarma, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. 2014. Generating Summary Risk Scores for Mobile Applications. IEEE Transactions on Dependable and Secure Computing 2014. https://doi.org/10.1109/TDSC.2014.2302293

[10] H. Hao, Z. Li, and H. Yu. 2015. An Effective Approach to Measuring and Assessing the Risk of Android Application. In 2015 International Symposium on Theoretical Aspects of Software Engineering. 31–38. https://doi.org/10.1109/TASE.2015.16.

[11] Q. H. Mahmoud, D. Kauling, and S. Zanin. 2017. Hidden android permissions: Remote code execution and shell access using a live wallpaper. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. 599–600. https://doi.org/10.1109/CCNC.2017.7983184.

[12] Alessio Merlo and Gabriel Claudiu Georgiu. 2017. RiskInDroid: Machine LearningBased Risk Analysis on Android. In ICT Systems Security and Privacy Protection, Sabrina De Capitani di Vimercati and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 538–552.

[13] Pragati Ogal Rai. 2013. Android Application Security Essentials. Packt Publishing.

[14] A. Sadeghi, R. Jabbarvand, N. Ghorbani, H. Bagheri, and S. Malek. 2018. A Temporal Permission Analysis and Enforcement Framework for Android. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). 846–857.

[15] Yidong Shen, Ming Xu, Ning Zheng, Jian Xu, Wenjing Xia, Yiming Wu, Tong Qiao, and Tao Yang. 2018. Android App Classification and Permission Usage Risk Assessment. In Collaborative Computing: Networking, Applications and Worksharing, Imed Romdhani, Lei Shu, Hara Takahiro, Zhangbing Zhou, Timothy Gordon, and Deze Zeng (Eds.). Springer International Publishing, Cham, 567–577.

[16] StatCounter. 2019. Mobile Operating System Market Share Worldwide. http://gs.statcounter.com/os-market-share/mobile/worldwide

[17] Ryan Stevens, Jonathan Ganz, Vladimir Filkov, Premkumar Devanbu, and Hao Chen. 2013. Asking for (and About) Permissions Used by Android Apps. In Proceedings of the 10th Working Conference on Mining Software Repositories (MSR'13). IEEE Press, Piscataway, NJ, USA, 31–40. http://dl.acm.org/citation.cfm?id=2487085.2487093

[18] Symantec. 2017. Internet Security Threat Report. https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf

[19] Franklin Tchakounté. 2014. Permission-based malware detection mechanisms on android: Analysis and perspectives. Journal of Computer Science 1, 2 (2014).

[20] Yang Wang, Jun Zheng, Chen Sun, and Srinivas Mukkamala. 2013. Quantitative Security Risk Assessment of Android Permissions and Applications. In Data and Applications Security and Privacy XXVII, Lingyu Wang and Basit Shafiq (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 226–241.