Permission-based Risk Signals for App Behaviour Characterization in Android Apps

Oluwafemi Olukoya, Lewis Mackenzie and Inah Omoronyia
School of Computing Science, University of Glasgow, U.K.
o.olukoya.1@research.gla.ac.uk, {lewis.mackenzie, inah.omoronyia}@glasgow.ac.uk

Keywords: Mobile Systems Security, Privacy Risk, Permission System, Malware Detection.

Abstract: With the parallel growth of the Android operating system and mobile malware, one of the ways to stay protected from mobile malware is by observing the permissions requested. However, without careful consideration of these permissions, users run the risk of an installed app being malware, without any warning that might characterize its nature. We propose a permission-based risk signal using a taxonomy of sensitive permissions. Firstly, we analyse the risk of an app based on the permissions it requests, using a permission sensitivity index computed from a risky permission set. Secondly, we evaluate permission mismatch by checking what an app requires against what it requests. Thirdly, we evaluate security rules based on our metrics to evaluate corresponding risks. We evaluate these factors using datasets of benign and malicious apps (43580 apps) and our result demonstrates that the proposed framework can be used to improve risk signalling of Android apps with a 95% accuracy.

1 INTRODUCTION

One of Android's major security mechanisms for enforcing restrictions on the privacy-related data and specific device operations that an application can access is permission control (Barrera et al., 2010; Felt et al., 2011c). Permissions are requested by applications as features describing an app's behaviour, indicating its attempt to interact with the device, local data or other apps (Enck et al., 2011). Some of the sensitive device data to which access can be granted include precise location, SMS messages, phone call logs, contact list etc., thereby making privacy an important challenge in the Android permission model. With the parallel growth of the Android operating system and its associated malware (Felt et al., 2011b; Zhou et al., 2012; Kelly, 2014), one of the ways to stay protected from mobile malware is by observing the permissions requested[1]. However, users do not fully understand the risks of granting permissions and the consequent implications (Felt et al., 2012b; Kelley et al., 2012; King et al., 2011). Users run the risk of an app being malware during adoption, with no reliable risk signal for characterizing behaviour based on permission request.

[1]https://www.symantec.com/blogs/threat-intelligence/persistent-malicious-apps-google-play

However, to assess the risk of installing an app, users have to rely primarily on community ratings to identify potentially harmful and inappropriate apps, even though community ratings typically reflect opinions about perceived functionality or performance rather than actual risks. We propose risk ratings, such that when users install a new app, they can assess the risk of the app being malware based on the behaviour characterization described by a permission sensitivity index and a permission mismatch, indicated by checking required against requested permissions. The main contributions of this work are summarised below:

- We provide a risky permission set, independent of typically known malware patterns, using a taxonomy of sensitive permissions. We created a ranking of the risks of Android application permissions by examining protection levels in the Android Ecosystem, permission category and permissions with a demoted security level.
- We provide an analysis of permission mismatch by evaluating the mismatch in the permissions the app is requesting and the permissions required for the functioning of the app. The proxy for the permission an app requires is an analysis of the permissions other apps with similar functions are requesting. We then generate security rules that

show the impact and likelihood of these signals.

- We evaluate our approach on two real-world datasets comprising of benign and malware dataset. The malware dataset contains 33580 apps, while the benign dataset contains 10000 apps. We demonstrate that the proposed framework can be used to improve user awareness about the risks of the Android permission system and apps. Our experimental results show that the proposed risk ratings are reliable with a precision of 97.56%, recall of 97.01%, and accuracy of 94.72%.

2 RISKY PERMISSION SET

In this section, we introduce our methodology for exploring permission-induced risk in Android apps.

1. **Protection Level:** Android provides a protection level attribute that characterises the level of sensitivity implied in the permissions requested. The protection level of the user-granted permissions is defined as: special, dangerous or normal, in order of decreasing risk. The first metric for determining the impact level of permissions is the default Android security protection level.
2. **Permission Group:** To avoid overwhelming users with complex and technical permission requests, Android organises permissions into groups related to a device's core functionalities. Under this system, permission requests are handled at the group level and a single permission group corresponds to several permission declarations in the app manifest. When a user grants a single permission in a group, subsequent requests of permissions in the same group are granted without user interaction. Our second metric involves ranking permission groups based on the number of permissions in the group.
3. **Demoted Permissions:** Our final metric for ranking permissions with respect to risk are downgraded permissions. These are permissions whose protection levels have been downgraded from higher risk, as they are now being granted by default. The security implications of these changes in the Android security state must be considered in any security solution (Zhauniarovich and Gadyatskaya, 2016).

The revised permission model results in four levels of sensitive permissions based on the three risk indicators. The taxonomy of sensitive information is presented in Figure 1.

3 SYSTEM OVERVIEW

Figure 2 provides an overview of the proposed system, where an app's behaviour is characterized by two risk ratings - its permission sensitivity index and permission mismatch, in order to explore the risk posed by the app. The permission sensitivity index is a sensitivity measure of an app based on the individual risk of the permissions it requests as determined by the proposed impact level of Android permissions. The permission mismatch checks the requested permissions of an app against its required permissions. The proxy for required permissions is an analysis of what similar apps in the same functional group are requesting. The key idea is to compare these presumed required permissions of an app (inferred from functional category) with its requested permissions (obtained from the permission set declared) and estimating any mismatch between the two. Finally, security rules are constructed from the risk ratings to characterize an app behaviour as potentially benign or malicious.

3.1 Permission Sensitivity Index (PSI)

In quantifying sensitive permissions from a security and privacy perspective, we propose a permission sensitivity index(PSI), which is a function of the permissions requested and their sensitivity level. PSI should be monotonic such that requesting less sensitive permissions reduces the value and vice versa. The overwhelming evidence indicates that most users do not understand what permissions mean, even if they are inclined to look at the permission list (Felt et al., 2012b; Kelley et al., 2012; King et al., 2011). With a sensitivity index, users can more easily understand the risk level of Android permissions and pay more attention to apps with high-risk values.

One way to approach this is to assign weights proportional to each sensitivity level to capture a numerical value that ranks the sensitivity of permissions requested with respect to the total number of permissions. In this approach, the PSI is calculated as a percentage of achieved value with maximal possible value for the app. While that would prove useful in some cases, it loses a lot of abstraction in its computation because it is difficult to determine what an optimal weight is. For example, given the request index of an app, G = [N_1, N_2, N_3, N_4] where N_i represents the number of permissions requested in level i. A potential weight vector W = [1, 0.75, 0.5, 0.25] could be assigned as an ordinal factor that captures the sensitivity of each level. Consider an app A with G_A = [9,0,0,0], and app B with G_B = [9,0,0,15]. By computing the sensitivity index using this approach, app

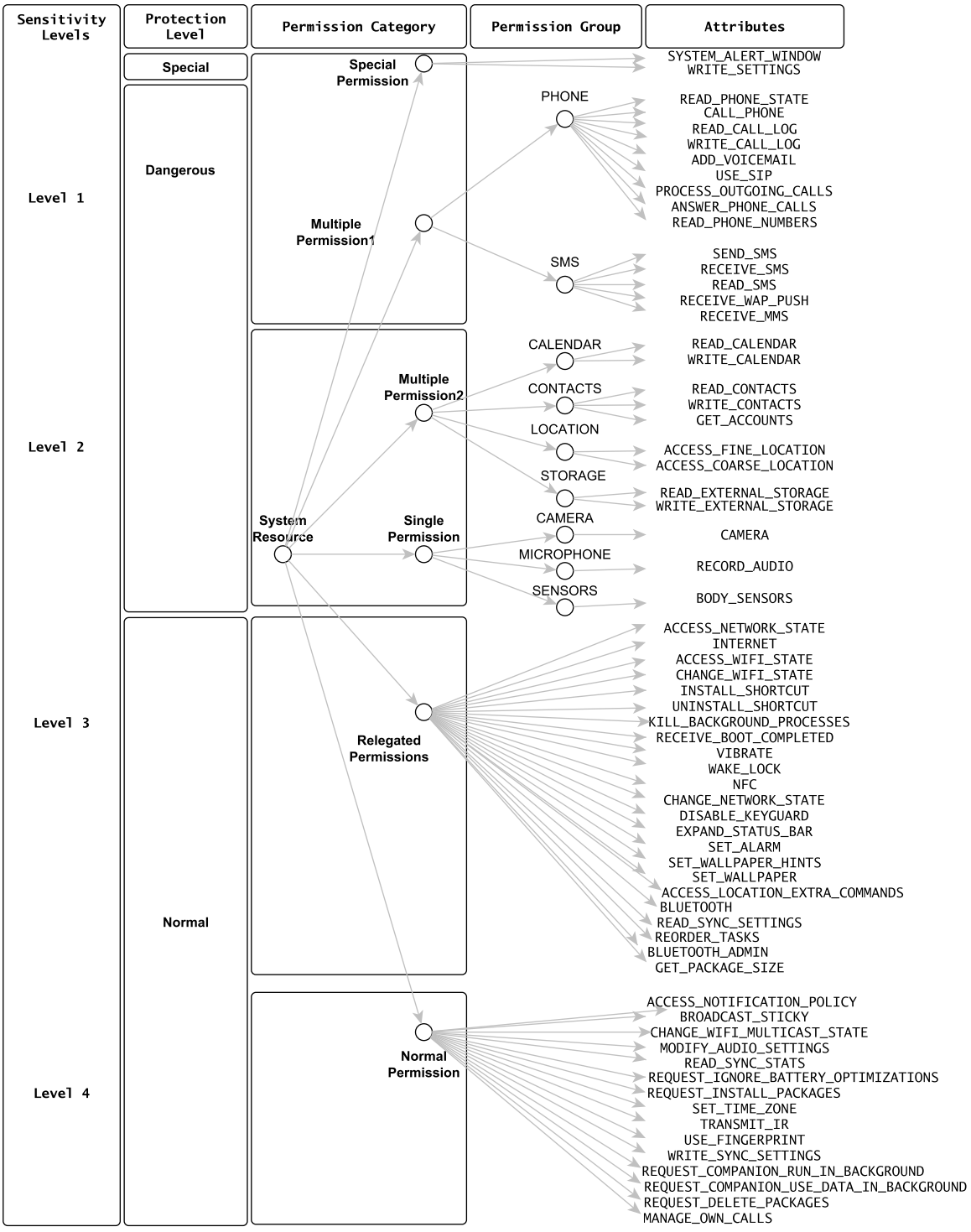

Figure 1: Proposed Impact Levels of Android Permissions (API Level 26).

A is assigned a higher value because all the permissions requested are in Level 1, while app B is assigned a lower value because the majority of the permissions requested are in Level 4, while ignoring the fact that app A and app B has the same amount of sensitive permissions in Level 1. This approach does not present a true representation of the sensitive permission request because $N_4 > N_1$ OR N_2. Moreover, N_4 is least-

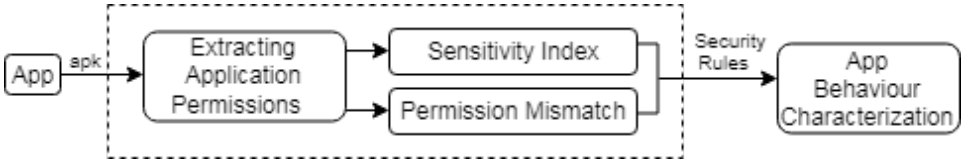

Figure 2: Proposed framework for Exploring Android Permission-Induced Risks.

sensitive with permissions granted automatically.

Hence, a characterisation of the sensitivity index should, therefore, be a spectrum from where an app does not require any permission in each level of sensitivity to when it requires all permission in the sensitivity levels. The sensitivity index should be able to (i) distinguish between an app that requests sensitive permissions and an app that does not. (ii) distinguish between apps that request sensitive permissions in varying quantity, while capturing the sensitivity levels of the permission request. It is observed that the extreme case of sensitive permission request occurs when the app requests a large number of sensitive permissions in Level 1 and Level 2. This guides our choice of characterisation, as it makes the previous example of request patterns of [9,0,0,15] and [9,0,0,0] identifiable. Finally, the proposed characterisation of the PSI is along five levels of sensitivity - `Moderate`, `Low` and `Very Low` (distinguishing apps that do not require sensitive permissions); `High` and `Extreme` (distinguishing between apps that require sensitive information). Table 1 shows the characterization of the PSI based on the sensitivity of the permissions requested. In this work, we only consider the permissions provided by the Android system, although an app can also define its own permissions.

3.2 Permission Mismatch

We propose to also evaluate permission risk by determining the mismatch between *requested* and *required* permissions based on the app category. We seek to investigate whether there is a mismatch between what an app is requesting and the functionality(purpose) of the app. Applications are grouped on distribution platforms based on the function they provide, e.g. Health &Fitness, Shopping, Finance etc. The proxy for requested permissions is the declared permissions, while the required permissions are evaluated by analysing permission requests of similar apps within the same functional category.

The approach to the level of mismatch using the app-group based approach is to generate a probability of occurrence vector, P_G, for the permission the group requires. This serves as the baseline for required permissions of apps based on their functional categories. The vector is of length 66, representing the set of user-granted permissions. The value of each index is a function of the number of times apps in the group requires the permission against the total number of apps in the group.

To determine the level of mismatch between what an app is requesting and what it requires, we represent the permissions it requires and the permissions its functional group requires in a vector space model. For each app group, the permission mismatch can be determined by evaluating the difference in the probability vector of the occurrence of the permissions in the app group against the permissions the app is requesting. The permission mismatch (M_{NO}) is a set of permissions an app is requesting whose probability of occurrence in its group is below the critical threshold value. The level of mismatch L_{MA} (Equation 1) is proposed as a measure of the sensitivity index of the set of permissions in M_{NO}.

$$L_{MA} = PSI(M_{NO}) \tag{1}$$

The level of mismatch is, therefore, a factor of the region in which mismatches occur, such that the level of mismatch is termed low-risk if `PSI(`M_{NO}`) = Very Low, Low, Moderate`, and high-risk if `PSI(`M_{NO}`) = High, Extreme`. We seek to investigate whether the level of permission mismatch could provide valuable insights as to determining the behaviour of an app as benign or malicious.

4 EVALUATION

We have presented a permission-based framework for risk signals that characterize an Android app behaviour. To evaluate this framework, we investigate the effectiveness of the risk signals in distinguishing between malicious and benign applications.

4.1 Study Subjects

We gathered Android malware samples from VirusShare [2], a repository of malware samples to provide security researchers. The files downloaded are Android 20140324.zip(`Mal_V1`) & 20130506.zip(`Mal_V2`). All

[2]https://virusshare.com

Table 1: Characterization of PSI.

PSI	Characterisation	Interpretation	Impact
VERY LOW	N_1=N_2=N_3=N_4=0	App does not request any permissions	Low-risk
LOW	N_1=N_2=N_3=0;N_4>0	App request ONLY Level 4 permissions	
MODERATE	N_1=N_2=N_4=0; N_3>0 N_1=N_2=0; N_3,N_4 >0	App request ONLY Level 3 permissions OR Level 3 & Level 4	
HIGH	$N_1 <= \frac{G_{max}[1]}{2}$ $N_2 <= \frac{G_{max}[2]}{2}$ (N_1 OR $N_2 \neq 0$)	App requests sensitive permissions in Level 1 OR Level 2 OR Both	High-risk
EXTREME	$N_1 > \frac{G_{max}[1]}{2}$ $N_2 > \frac{G_{max}[2]}{2}$ (N_1 OR $N_2 \neq 0$)	App request large sensitive permissions in Level 1 AND/OR Level2.	

the malapps from VirusShare were approved by VirusTotal[3]. We also selected the Drebin dataset, Mal_D, a malicious apps dataset from different malware families (Arp et al., 2014). Malware was also collected from Contagio repository[4], Mal_C and recent samples from the Koodous repository[5], Mal_K. Finally, the Android malware dataset, Mal_A(Wei et al., 2017) was used. For the benign app dataset, we downloaded 10,000 apps from the Top 25 categories from Google Play. After removing corrupt APK malware files and inaccessible files, the final dataset consists of a total of 43580 apps, with 33580 being malware[6]. For each sample, we extract the permissions requested using the AndroidManifest.xml file present in the package.

4.2 Validating Sensitive Permission Categorisation

To validate the risky permission categorisation, we compare our ranking with alternative ways to find impact levels of Android permissions. We compare our rankings with the study of (Felt et al., 2012a), which ranks the risk of permissions according to user ratings. In particular, we compare the ten-highest ranked risks for permissions that are still relevant in the Android permission system with our proposed sensitivity levels. We present the results in Table 2, where 6 out of the 10 highest risks ranked by users correspond to permissions that belong to most sensitive level in the proposed ranking system, while the other four risks correspond to permissions in Level 2. This suggests that the risk metrics used in the proposed impact level permission corresponds to how users perceive the impact of Android permissions.

Furthermore, we examine if the characterisation of the impact levels can be used for describing an app. We observe the PSI of samples in our dataset to check if its request pattern matches our prediction of high risk and low-risk apps based on PSI. For an app to be in the low-risk category, there are two characteristics: it does not require any Android permission (PSI = Very Low) or it does not require any sensitive permission (PSI = Low or Moderate). Based on the PSI of the data subjects, 1.78% of malware apps in the dataset belong to this category, while 16.01% of benign apps exhibit such behaviour. For an app to be a high-risk app, it requires sensitive permissions in Level 1 or Level 2 (PSI = High or Extreme). None of the apps in the dataset has a Low psi. This is the behaviour of apps that require the least sensitive permission in proposed systemic ranking of Android permissions (Level 4). This shows that one way to characterize an app behaviour is through sensitive permission request. We extracted security rules, (Rules 01, Rules 02) (Table 3) from the PSI of the data subjects as a form of measure characterizing the features identifiable with each class label (1 for Benign, -1 for Malware).

Malware apps are characterised by risk level of high based on the permission request patterns. This justifies the rationale of using permissions to rank the risk of Android apps. This further supports the risk indicators used and justifies the hypothesis that the sensitivity of the permissions requested by an app is an important factor in quantifying the risk factor. This result is particularly interesting with respect to malware detection, where apps are fingerprinted by observing their static features, such as the API calls, intent etc. We argue that the PSI would be an enhancement to static feature generation for malware detection of apps. While Rule 02 is characteristic of a malware, it is not dominant enough to distinguish a malware from a benign app. However, it shows a warning signal for the kind of apps that should be investigated. Therefore in fingerprinting an app for malware detection, reliable static features that can be used as a warning sign:if(PSI = High-Risk). Thus, low-

[3]https://www.virustotal.com/en

[4]http://contagiomobile.deependresearch.org/index.html

[5]https://koodous.com/

[6]Result files are available at https://www.dropbox.com/sh/62avus8x7oiaq3f/AABi7cdXGxa1qSjgvGmad0qXa?dl=0

Table 2: Comparison of User Ratings of Impact Level of Android Permissions and proposed systemic ranking.

Permission	User Rating	Impact Level	Permission	User Rating	Impact Level
CALL_PHONE	97.41%	1	READ_SMS	94.48%	1
READ_PHONE_STATE	97.41%	1	RECEIVE_SMS	94.48%	1
SEND_SMS	96.39%	1	WRITE_SETTINGS	94.39%	1
WRITE_CONTACTS	95.89%	2	GET_ACCOUNTS	93.37%	2
READ_CONTACTS	95.89%	2	READ_EXTERNAL_STORAGE	90.60%	2

risk apps are more likely to be benign, while high-risk apps are more likely to be malicious.

Existing approaches have categorised risky permissions based on the frequency of occurrence in malware datasets, benign datasets or a combination of both. We argue that this approach is limited because it does not emphasise the dynamic complexity and evolving nature of the Android permission system. Android keeps modifying permission space by removing permissions and also limitations based on malware dataset space. To justify our reasoning, we consider the permission request pattern of our malware data size and compare it with current state-of-the-art risky permission sets in (Peng et al., 2012; Wang et al., 2013; Wang et al., 2014; Sarma et al., 2012; Enck et al., 2009). In particular, we consider all permissions whose probability of occurrence is not zero. These comparisons show that the permissions considered by (Enck et al., 2009) are only relevant for 19% of malware in the dataset, while (Wang et al., 2014) characterizes 60% request patterns of malware in our dataset. The reason for the partial characterization of malware request patterns from these approaches is because defining risky permission sets in these works are dataset-dependent.

With the permission sensitivity index, we validate that when a user installs a new app, they run the risk of the app being malware and the impact of the risk is reflected in the categorisation of the PSI, even without knowing existing malware patterns.

4.3 Level of Permission Mismatch

To gain valuable insight into requested versus required permissions, the probability of occurrence of individual permissions for each app group is computed as the proportion of the frequency of the permission request divided by the total number of apps in the group. Using the probability of occurrence, with a critical threshold set at $\theta_f = 20\%$, produces some intuitive results, such as that location permissions are frequently required by apps in the Travel&Local and Maps&Navigation app category. The probability of occurrence can be used as a baseline for matching an app's requested permission with its required permissions based on its functional group.

To investigate the level of mismatch L_{MA} in the malware dataset, some malware metadata such as app title, app category are required. This is to ensure that the probability-of-occurrence vector used in the analysis corresponds to the app category of each malware app. For 49 app categories, we were able to find exactly the same package identifier retrieved from the apk files of the malware app on the store.

From the identified app categories, we investigated the level of the mismatch with respect to varying degrees of critical threshold - θ_f = [20%, 35%, 30%, 35%, 40%]. We examined the level of mismatch that occurs with a malware application requesting a sensitive permission that is requested by less than θ_f% apps in the corresponding benign category. For an app in the low-risk zone, the permissions requested closely mirror the permissions required by its benign counterparts with any mismatch occurring in less sensitive permission sets (Level 3 and Level 4). A high-risk app, on the other hand, has permission mismatches with benign counterparts on sensitive permissions (Level 1 and Level 2). The correlation between the spectrum of critical threshold against malware that is low-risk and high risk was found to be strongly negatively and positively correlated $(R^2(\theta_f, N_{lowrisk} = -0.988), R^2(\theta_f, N_{highrisk} = 0.988))$. With a set critical threshold, malware apps tend to request sensitive permissions, rarely required by their benign counterparts.

To investigate the level of permission mismatch as a means of distinguishing behaviour, we compare the level of permission mismatch of each benign app category with its corresponding malware, at a critical threshold of 20%. The behaviour is described by Rules[03-04] in Table 3. We observe that the permission mismatch is indeed distinguishable behaviour. While at least 6 out of 10 benign apps do not have permission mismatches in critical permissions, at least 6 out of 10 malapps request permissions not required by other apps in the group, at the 20% threshold.

4.4 Distinguishable Behaviour

The aim of this section is to investigate whether the combination of these risk ratings could give further insight into distinguishing the behaviour of malapps and benign apps.

4.4.1 PSI and Permission Mismatch

Since we have already shown that low-risk PSI apps are most likely to be benign with a 90% likelihood, we proceed to show if the combination of a high-risk psi and the values for the level of mismatch can provide further distinguishability. We investigate the likelihoods of the combination of the risk ratings with respect to the dataset as a means of discriminating between malapps and benign apps. The results are presented as `Rules 05-06` in Table 3.

With these likelihoods, a high-risk PSI and low-risk permission mismatch are indicative of a benign app, while a high-risk PSI and high-risk indicates the presence of a malicious app. Users do not often pay attention to the textual description of information related to the risk of permissions in the current android permission system (Felt et al., 2012b). We show that the permission sensitivity index and level of permission mismatch metrics provide a measure of distinguishability between malicious and benign applications which can improve user attention and awareness to the risks of Android permissions and apps.

4.4.2 PSI and Distinguishing Permissions

Malicious app developers might select a misleading category for the app by joining a group whose member apps request more permissions, even if functionality wise, it does not belong to that group. This threatens the validity of the risk rating produced by the level of permission mismatch. We, therefore, investigate permissions and combination of permissions that are characteristic of the dataset to form a compound risk rating alongside the proposed risk signals. Upon investigating, we discovered single permissions that clearly distinguish malware from benign apps. The single permission, `READ_PHONE_STATE` was requested by 98% of malware in the dataset, while the probability that a malware does not request `read access to the phone state` and `full network access` is 0.04. This suggests that most malware needs to read device and phone data with a full network access. This also supports our criteria for the proposed ranking of Android permissions, where the single permission was assigned to `sensitivity Level 1` - most sensitive level, without a prior knowledge of malware request pattern. We then proceed to investigate the likelihood of high-risk PSI apps and permission combinations for reliable risk signalling. The security rules that characterize the reliability of these features are presented as `Rules 07-09` in Table 3.

4.5 Reliability of Risk Ratings

We proceed to show the reliability of the security rules extracted from the proposed risk ratings in identifying potential malware. The impact is characterized by the sensitive permissions requested by the app (PSI), while the likelihood measures the probability that the security rule belongs to the class label. The impact and likelihood of these risk signals are presented in Table 4.

Based on Table 4, benign apps are characterised by low-risk and high-risk rating values, while malicious apps are only characterised by high-risk values. These risk signals can be triggered in identifying potential malware. User attention of the risk to Android permissions and apps can be improved by utilising the risk information provided by each rule, such that an app can be triggered as potential malware based on its level of impact and likelihood based on the Rules.

To evaluate the reliability of these risk ratings, we evaluate `High-Risk PSI and Permission Rules` with the `PSI Rules` (see Table 4) on a benign and malicious app dataset. The benign dataset is downloaded from GooglePlay, while the malware dataset is `Mal_A`(Wei et al., 2017), which provides an up-to-date picture of the current landscape of Android malware, categorized in 135 varieties among 71 malware families. The results are presented in Table 5, where the risk rating achieved an accuracy of 95% in reporting malapps. This indicates that the rules extracted from the risk ratings well discriminate malapps from benign apps, and thus, permission requests characterize the behaviours of Android apps to a certain extent. While our approach shows a good performance for reporting malapps, it suffers two limitations that impacted on its accuracy - i) malapps that do not require any permissions, and ii) malapps that have the same risk rating as benign apps. In such scenarios, relying on permission request only is not feasible.

Previous studies have shown that community ratings on distribution platforms are not sufficient to alert users to the level of privacy risk they are exposed to when they download an app (Chia et al., 2012). In conclusion, the risk signals can be used as a privacy risk indicator to support current forms of community ratings, such that a recommendation system for Android apps using these risk ratings can be built to assist users in their security evaluation of Android apps.

Table 3: Security Rules from Risk Ratings.

PSI Rules	LMA Rules	PSI & LMA

Rule 01
IF(PSI=LOW-RISK)
THEN Class=1[90%]
Rule 02
IF(PSI=HIGH-RISK)
THEN Class=-1[54%]
	Rule 03
	IF(LMA=LOW-RISK)
	THEN Class=1[65%]
	Rule 04
	IF(LMA=HIGH-RISK)
	THEN Class=-1[65%]
		Rule 05
		IF(PSI=HIGH-RISK)&(LMA=LR)
		THEN Class=1[61%]
		Rule 06
		IF(PSI=HIGH-RISK)&(LMA=HR)
		THEN Class=-1[70.00%]
		*LR=LOW-RISK, HR=HIGH-RISK

High-Risk PSI and Permissions Rules		
Rule 07 IF(PSI=HIGH-RISK) AND READ_PHONE_STATE=1 AND ACCESS_NETWORK_STATE=1 AND INTERNET=1 THEN Class=-1[75%]	Rule 08 IF(PSI=HIGH-RISK) AND READ_PHONE_STATE=0 AND ACCESS_NETWORK_STATE=1 AND INTERNET=1 THEN Class=1[89%]	Rule 09 IF(PSI=HIGH-RISK) AND READ_PHONE_STATE=1 AND CAMERA=1 AND READ_EXTERNAL_STORAGE=1 AND/OR RECORD_AUDIO=1 THEN Class=1[82%,93%]

Table 4: Malapp Reporting Rules.

Class	Rule	Impact	Likelihood
Benign	01,08,09	Low-Risk, High-Risk, High-Risk	[90%, 89%,[82%, 93%]]
Malicious	06,07	High-Risk, High-Risk	[70%,75%]

Table 5: Detection of Malapps Using Risk Signals.

TP	FP	FN	P(%)	R(%)	A(%)
18110	452	558	97.56	97.01	94.72

TP = True Positive, FP = False Positive, FN = False Negative, P - Precision, R - Recall, A - Accuracy

As malicious apps have become more complex, more features need to be explored in order to improve the capacity of detectors for the detection of novel malapps. Our analysis results show that the use of the proposed risk ratings can be effective in reporting malapps.

5 RELATED WORKS

Usability studies on the effectiveness of the Android permissions system have indicated that the current permission system does not help most users make informed security decisions about malware threats (Felt et al., 2012b). The complexity of Android permission requests has been investigated (Frank et al., 2012). (Barrera et al., 2010) used a Self-Organizing Map (SOM) algorithm to visualise the permission-based systems of the most popular Android apps. (Felt et al., 2011a) have presented a permission analysis tool, *Stowaway*, that detects *overprivilege* in Android apps, a situation in which an Android app requests more permission than necessary. A study of permission evolution in the Android ecosystem and its usage was conducted by (Wei et al., 2012). *PScout* was developed, as an extension to *Stowaway*, to extract a permission specification API from Android OS source code with static analysis (Au et al., 2012). While our methods also provide a better understanding of permission request patterns of Android apps, our work examines the relationship between permission request and their risk in Android apps as a measure of distinguishing between malapps and benign apps.

The closest research to our work was reported by (Enck et al., 2009), (Sarma et al., 2012), (Peng et al., 2012), (Wang et al., 2013) and (Wang et al., 2014). Using the permission information as the data source, the risk presented by an Android app was evaluated by examining how rare are the permissions request patterns in comparison with the permissions requested by other apps in the same categories (Sarma et al., 2012). Probabilistic generative models such as Naive Bayes could be used for risk scoring and risk ranking for apps (Peng et al., 2012). In contrast to the 9 permission rules proposed by *Kirin* (Enck et al., 2009), 200 permission detection rules for malapp detection built on risky permissions set were also developed by (Wang et al., 2014). The common denominator amongst all these approaches is obtaining features from both benign and malicious dataset for malapp detection. Security rules are then built based on identified permission request patterns in the dataset. This approach is often hindered by the evolving nature of the Android permission system. For example, 2 detection rules in (Enck et al., 2009) refer to permissions that are no longer supported. Our approach develops risk ratings that rely on a proposed impact level of Android permissions independent of a *priori* request patterns that emphasize the complex and dynamic nature of the Android permission model. 6 out

of the 9 proposed security rules are independent of known malware request pattern, while 3 depends partially on identified request patterns. Our analysis thus provides a vision regarding the use of security rules that are not completely dependent on known permission request patterns for the detection of malapps.

6 CONCLUSION

We have investigated the feasibility of characterizing app behaviour using risk ratings from proposed impact levels of Android permission. We demonstrate that the risk signals can be used to assist layman users in their security evaluation of Android apps. The first rating is the sensitivity index of the app based on the impact levels of requested permissions, the second rating is on its permission request compared to similar apps, while the last rating is on its permission that characterizes class label of apps as benign or malicious. Our result demonstrates that the proposed framework can be used to improve risk signalling of Android apps with a 95% accuracy.

For risk signalling, the ratings provide a measure of awareness and scrutiny as a user defence against malicious applications. For malware detection, a detailed knowledge of app behaviour is essential for signatures. Existing approaches and future analysis could incorporate these ratings as the first phase in prioritizing, especially when a large dataset of apps is involved. For future work, we are exploring features to make the risk ratings adjustable to contextual factors, such as actual usage of permissions in the source code.

REFERENCES

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., and Siemens, C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26.

Au, K. W. Y., Zhou, Y. F., Huang, Z., and Lie, D. (2012). Pscout: analyzing the android permission specification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 217–228. ACM.

Barrera, D., Kayacik, H. G., van Oorschot, P. C., and Somayaji, A. (2010). 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*, pages 73–84. ACM.

Chia, P. H., Yamamoto, Y., and Asokan, N. (2012). 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*, pages 311–320. ACM.

Enck, W., Octeau, D., McDaniel, P. D., and Chaudhuri, S. (2011). A study of android application security. In *USENIX security symposium*, volume 2, page 2.

Enck, W., Ongtang, M., and McDaniel, P. (2009). On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245. ACM.

Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. (2011a). Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM.

Felt, A. P., Egelman, S., and Wagner, D. (2012a). I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 33–44. ACM.

Felt, A. P., Finifter, M., Chin, E., Hanna, S., and Wagner, D. (2011b). A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM.

Felt, A. P., Greenwood, K., and Wagner, D. (2011c). The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012b). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the eighth symposium on usable privacy and security*, page 3. ACM.

Frank, M., Dong, B., Felt, A. P., and Song, D. (2012). Mining permission request patterns from android and facebook applications. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 870–875. IEEE.

Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., and Wetherall, D. (2012). A conundrum of permissions: installing applications on an android smartphone. In *International Conference on Financial Cryptography and Data Security*, pages 68–79. Springer.

Kelly, G. (2014). Report: 97% of mobile malware is on android. this is the easy way you stay safe. *Forbes Tech*.

King, J., Lampinen, A., and Smolen, A. (2011). Privacy: Is there an app for that? In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 12. ACM.

Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C., and Molloy, I. (2012). Using probabilistic generative models for ranking risks of android apps. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 241–252. ACM.

Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., and Molloy, I. (2012). Android permissions: a perspective combining risks and benefits. In *Procee-*

dings of the 17th ACM symposium on Access Control Models and Technologies, pages 13–22. ACM.

Wang, W., Wang, X., Feng, D., Liu, J., Han, Z., and Zhang, X. (2014). Exploring permission-induced risk in android applications for malicious application detection. *IEEE Transactions on Information Forensics and Security*, 9(11):1869–1882.

Wang, Y., Zheng, J., Sun, C., and Mukkamala, S. (2013). Quantitative security risk assessment of android permissions and applications. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 226–241. Springer.

Wei, F., Li, Y., Roy, S., Ou, X., and Zhou, W. (2017). Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 252–276. Springer.

Wei, X., Gomez, L., Neamtiu, I., and Faloutsos, M. (2012). Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 31–40. ACM.

Zhauniarovich, Y. and Gadyatskaya, O. (2016). Small changes, big changes: an updated view on the android permission system. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 346–367. Springer.

Zhou, Y., Wang, Z., Zhou, W., and Jiang, X. (2012). Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In *NDSS*, volume 25, pages 50–52.