PrivacyPalisade: Evaluating App Permissions and Building Privacy into Smartphones

(Invited Paper)

Anthony Quattrone, Lars Kulik,
Egemen Tanin, Kotagiri Ramamohanarao
Department of Computing and Information Systems
University of Melbourne, Australia
Email: quattronea,lkulik,etanin,kotagiri@unimelb.edu.au

Tao Gu
School of Computer Science and IT
RMIT University, Australia
Email: tao.gu@rmit.edu.au

***Abstract*—Privacy has become a key concern for smartphone users as many apps tend to access and share sensitive data. However, it is not easily understandable for users which apps access what type of data and which are the minimal access permissions required to achieve a certain functionality. Although there are apps targeting privacy concerns, they only show which type of data is accessed but not whether it is necessary for an app to achieve its functionality. We propose a model that groups apps together in terms of advertised functionality and assesses an app's privacy intrusiveness based on the requested permissions relative to similar apps. To improve user comprehension of permissions, we implement PrivacyPalisade and demonstrate Android OS level modifications that use visual cues to indicate the privacy intrusiveness of an app. If an app requests a permission that is not common in its cohort, the user is notified and shown visually the permission implications. We demonstrate that the proposed approach is scalable and incurs little performance overhead.**

I. Introduction

The ubiquity of mobile smartphones combined with advancements in mobile network infrastructure has created a strong market for third party apps. While the apps and service providers are of great convenience to its users, there are concerns of the privacy implications [1]. Mobile apps greatly enhance the experience of using smartphones. These apps are typically developed by both large software firms and independent programmers [2]. Modern platforms allow for third-party developers to create apps and distribute them in app stores or marketplaces. Popular app stores for Android include Google Play, Amazon Appstore and GoAPK while iOS users primarily download apps from the Apple AppStore. Google Play and Apple AppStore now list over one million apps.

Third-party developers can access a large amount of user data using standard API calls provided by the mobile platform and send it directly to remote servers. Apps often make use of user data to provide functionality. For example, VoIP apps such as Viber require access to a user's contacts list to provide a list of people the user can call. For this case, iOS displays a popup asking the user if the app is allowed to access contacts. In contrast, Google Play only alerts the user at installation time. The contacts permission is justified for the VoIP app in order to provide auto-dialing. However, many other apps, such as weather apps, do not require the permission in general.

The architecture for privacy protection varies between mobile platforms. Android depends on a software based permissions system. When an Android user downloads an app, a dialog at installation explains what data an app can access. In contrast, Apple employs staff members that manually review apps with internal checks. Both approaches are not without issues. It has been shown that user comprehension of permissions is low among Android users. While iOS users need to place trust in the AppStore review process which is not entirely transparent. A more comprehensible privacy solution is needed that does not unnecessarily hinder app functionality while at the same time protects user privacy.

App similarity has been provided as a measurement by marketplaces. It is commonly based on download statistics across apps. For example, if users downloaded one app, how many of those same users downloaded another app. We propose a protection method that uses app similarity to detect anomalies. From building a table of permissions requested of an app and comparing it with the permissions of those of similar apps, anomalies can be detected. We use Isolation Forest [3], a data-mining technique for detecting outliers to find the anomalies based on the requested permissions.

We focus on Android, which is a popular open source Linux based mobile OS designed for smartphones and tablets. Android is used to prototype our approach as its open nature allows changes to be made to the privacy models. As of 2014 Android has an estimated global marketshare of 81.5% [4], thus privacy research of the platform is of great importance.

To evaluate our approach, we developed a web scraper and collected information of nearly 17,000 of the most popular free and paid apps from the Google Play store. For each app, data collected by the web scraper includes the permissions required by the app and a list of similar apps as suggested by Google Play. We demonstrate our approach for flagging outliers in app permissions will eventually lead users to pick apps that take more conservative paths to user data access.

Based on our model, we implemented a service called PrivacyPalisade that checks apps installed on the smartphone. If an app is found to contain permissions that are outliers, the user is notified about the nature of the data the app requested at launch time. We made OS level modifications to achieve this. Our evaluation shows that PrivacyPalisade does not add much overhead to the system and uses little resources. It works across free and paid apps as it does not need access to the bytecode files.

Detecting potentially harmful Android malware using requested app permissions as training vectors for use in data-mining approaches has been attempted before in [5]–[7]. PrivacyPilsade is more privacy focused and differs by determining if permissions are justified relative to app functionality. The permissions of similar apps are used as means of comparison to achieve this.

In summary our key contributions are to:

- Propose an approach to highlight outlier permissions relative to an app's category and functionality;
- Implement PrivacyPalisade, a ready-to-deploy application that highlights privacy implications to Android users;
- Demonstrate OS level modifications that receive PrivacyPalisade messages to help alert users of privacy implications;
- Provide case studies of apps that we believe do not follow the path of least privilege.

II. App Classification

In order to detect if an app has permissions that are excessive relative to advertised functions, a comparison can be made with apps that provide comparable features. Similar apps as suggested by marketplaces provides a good cohort for comparisons. Anomaly detection techniques like Isolation Forest can be trained on permissions of similar apps and evaluated on the target app to determine if it is an outlier.

A. Dataset

Apps in the Google Play marketplace are listed under a range of categories. Each detailed app listing states all permissions required, also presented are suggested apps that are similar. We developed a web scraper to collect this information across 16,581 popular apps. While Google Play contains millions of apps, scrolling through the catalog lists the most popular apps. Thus, we considered the popular apps interesting for analysis because they are widely used.

B. Isolation Forest Overview

Isolation Forest is a unique anomaly detection technique as it builds a profile that explicitly isolates anomalies as opposed to building a profile of normal points and finding those that do not conform [3]. In many cases, there are only a small number of similar apps to compare to a target app. Thus, Isolation Forest is a suitable choice because it has demonstrated high performance with minimal training.

C. Data-Mining Approach

Data is processed to represent each required permission as a separate field using a binarized value. A table is created for each target app. An example is shown in Table I.

App Name	**Perm1**	**Perm2**	**Perm3**	**PermN**
SimilarApp1	0	1	1	0
SimilarApp2	1	1	1	0
SimilarApp3	1	1	1	1
TargetApp	0	1	0	1

TABLE I: Representation of Apps Permissions

The permissions we used for training include:

- Read your contacts
- Read phone status and identity
- Approximate location
- Precise location
- Run at startup
- Record audio
- Call phone numbers
- Send SMS messages
- Read SMS messages
- Read calendar events
- Require full network access

An IsolationForest is constructed from the similar app vectors. The target app is then evaluated resulting in a IsolationScore between 0 and 1. A score of 1 indicates definite anomalies, while 0.5 indicates the app is consistent with similar apps. The following rules are applied to determine the level of severity of an app:

- Red Alert - If an IsolationScore is greater than ϵ and uses a sensitive permission;
- Blue Alert - If an IsolationScore of less than ϵ and uses any sensitive permissions;
- Green Alert - If an app does not require any sensitive permissions.

ϵ	Green(%)	Blue(%)	Red(%)
0.5	26.51	36.62	36.87
0.6	35.14	50.48	14.38
0.7	**36.45**	**56.34**	**7.21**
0.8	37.19	60.09	2.72
0.9	37.56	62.44	0.00

TABLE II: Percentage of Apps Flagged Across Alert Levels for Different Values of ϵ

The value for ϵ used by PrivacyPalisade was empirically determined by testing different values and taking the ratio of alerts that appeared the most reasonable, in the case of our dataset it was $\epsilon = 0.7$.

D. *Outlier Results*

We ran our app classification technique on every app in our dataset. Table III lists common categories and the percentage of alerts triggered by apps. Communications had the highest percentage of red alerts, while books were amongst one of the safest categories.

Category	# Apps	Green(%)	Blue(%)	Red(%)
Communication	381	14.70	73.32	12.07
Social	382	24.35	65.18	10.47
Music Games	320	59.06	30.94	10.00
Action Games	487	32.03	58.52	9.45
Adventure Games	447	39.60	54.36	6.04
Lifestyle	318	42.09	52.53	5.38
Books	356	55.62	39.89	4.49

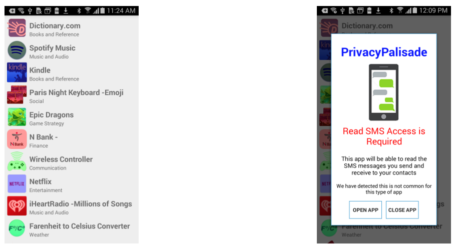
TABLE III: Number of Outliers Detected Per Category

III. PrivacyPalisade

In this section, we present the design of PrivacyPalisade, a system designed to help protect users from potentially privacy invasive apps and improve user comprehension. To improve user comprehension, our system flags outlier apps installed and displays permissions implications at launch time. This is challenging to implement because third-party apps do not allow to intercept and block an app when it is launched. Thus, we made enhancements at the OS level. The Android OS Launcher is modified to overlay privacy information and recompiled as a custom ROM that can be deployed on Android devices. The system consists of a web service, an Android app and an Android background service.

The web service maintains a database of Android apps and their respective privacy information as generated in Section II. By passing an Android app package name via a GET request, a JSON response is returned with the privacy rating and outlier permissions. The web service was implemented in PHP and deployed on Apache.

PrivacyPalisade runs a background service on the Android device that communicates with a server retrieving information about apps a user has installed. This information is stored in an internal database local to the user's device, entries are added and removed when a user installs and removes an app. Our background service communicates privacy information both to the PrivacyPalisade Activity and the Android Launcher. Other apps and widgets can also use PrivacyPalisade information.



Read SMS Access is Required

This app will be able to read the SMS messages you send and receive to your contacts

We have detected this is not common for this type of app

(a) List of Installed Apps (b) Visual Alerts

Fig. 1: PrivacyPalisade

Users can browse the privacy information for installed apps using the user interface displayed in Figure 1. Icons are colored based on invasiveness. Green denotes safe, blue indicates neutral and red for potentially invasive. When an app is opened, a popup dialog is loaded which presents a view explaining permissions used.

The Android Launcher displays the home screen, phone dialer, messaging and app icons for users to launch third-party apps. While the PrivacyPalisade UI is useful to display if apps are safe, it is more convenient for the user to see this information directly in the launcher. To build the customized launcher, we downloaded the OS source code for Android 4.4 KitKat from https://source.android.com and compiled it on Ubuntu Linux 14.10.

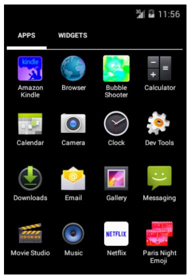
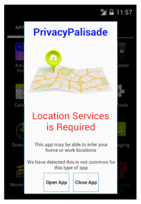


Fig. 2: Color Coded Android Launcher Icons

The original launcher was modified to listen for broadcasts from the background service. Based on the invasiveness level; the Bitmap of the app icon is overlaid with a color filter. An example is displayed in Figure 2. An additional class is added to the source tree to display custom dialog boxes when an app is executed from the Launcher application. If the user selects "Open App", the original Intent to start the

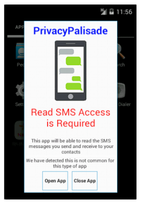

PrivacyPalisade

Location Services is Required

This app may be able to infer your home or work locations

We have detected this is not common for this type of app

(a) Location Alerts

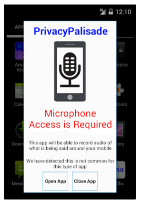

PrivacyPalisade

Read SMS Access is Required

This app will be able to read the SMS messages you send and receive to your contacts

We have detected this is not common for this type of app

(b) Read SMS


PrivacyPalisade

Microphone Access is Required

This app will be able to record audio of what is being said around your mobile

We have detected this is not common for this type of app

(c) Record Audio

Fig. 3: PrivacyPalisade Permissions Alert Dialogs

Activity is called, otherwise if a user selects "Close App", the app will not open and the dialog will dismiss.

We show the information of permission outliers with intuitive icons. If a location permission is detected as an outlier, the user is displayed an image of a map and a house marker (Figure 3a) making it obvious that the app can possibly know a user's home location. The SMS messages bubble (Figure 3b) indicate the app wants to read SMS messages, using bubbles is intuitive inspired by the way that users typically read messages on a modern smartphone. The microphone screen displayed in Figure 3c makes it clear the phone can start recording.

IV. App Case Studies

Our study shows that PrivacyPalisade detects between 5% to 10% of apps as outliers in each Play Store category. The apps we selected to present as case studies are very popular and require permissions that are not needed for their functionalities. It was found that many apps request "precise location" when only "approximate location" is required. Furthermore, some apps request permissions in which there is no direct use case to provide app functionality. We show some examples as follows.

A. *iHeartRadio*

iHeartRadio is a popular free music streaming service for Android and iOS. At present it has received between 10 to 50 million downloads from Google Play.

PrivacyPalisade flagged it as an outlier because it required the "precise location" permission which only 14% of similar apps required. This permission is used to determine what local radio stations are available in the area. For iHeartRadio to perform a local radio station lookup, "precise location" is not needed. TuneIn radio which provides a similar service only required "approximate location" to achieve the same function. All competitors received a similar number of downloads to iHeartRadio, indicating that precise location access is not a deterrent for users.

B. *Dictionary.com*

Dictionary.com provides a free online English dictionary app for its Android and iOS users. The Android version has received between 10 to 50 million downloads. The app requests 11 permissions, one of which is sensitive and flagged by PrivacyPalisade ("precise location").

The app requires location to support the local lookups feature which allows the user to see nearby word searches. Similar competitive apps such as the Oxford Dictionary of English and Merriam-Webster do not require any location information. Each of these dictionary apps are very popular, thus indicating it is not apparent to users that an alternative dictionary that is not location invasive is available. Precise location is not needed to provide a nearby search feature, approximate location is sufficient and would result in the app following the least privileged path.

C. *Hana Bank*

Hana Bank is one of the largest banks in South Korea. A mobile app is provided to their customers available on both Android and iOS. Google Play states that it has received between 1 to 5 million installs. PrivacyPalisade detected 25 permissions requested, many of which give access to sensitive data. For example, only 20% of similar apps required access to the "read your contacts" and "read call log" permissions.

The app requires access to contacts to allow the user to see a list of people they can transfer money. We observed banking apps require a combination of "directly call numbers" and "write call log" for the app to provide a direct way of calling customer support numbers. Removing the "read call log" permission would not hinder this functionality. Many similar banking apps require the "receive text messages" permission which allows the app to verify the phone number via reading a confirmation SMS. For this type of verification, "read your text messages" is not needed. However, it is requested by the app.

V. Related Work

Recent research on privacy is focused on the sensitivity of data stored on mobile devices. No current platform has achieved a good balance between control, information and interactivity [8]. Early protection techniques used frequent intrusive popups requesting a user for permission to access data. For example each time a location sample was required, the user is asked to give permission. Our solution only alerts users when an app is detected as an outlier. Therefore, it reduces the amount of popups from the traditional approaches.

Many users do not easily comprehend the implications of granting third party apps permission to access data [9]. A user study was performed via an Internet survey with 308 participants. It was found only 15% of the participants paid attention to the permissions at installation time. This highlights the need for improving user comprehension. PrivacyPalisade uses clear examples of the implications of permissions in outlier apps.

Self-organizing maps (SOMs) were used to visualize the Android permissions system and provide insights of where it could be improved [10]. Some permissions are used frequently while other permissions are only used by a small subset of applications. While there are many permissions a user can allow or disallow for mobile apps, it was found that few clusters cover most users' privacy preferences [11]. This can help in determining how strict to implement user privacy controls.

Risk signals have been proposed using a SVM model that compares apps across two datasets and looks for rare permissions across categories [12]. Privacygrade.org [13] researchers use crowd-sourced data which require extensive data collection. Rules for detecting dangerous combinations of permissions was proposed in [14]. For example, a combination of Internet and microphone permissions enable the app to record mobile conversations. Risk signals are also used by PrivacyPalisade to alert users of privacy implications.

A framework for detecting Android malware based on permissions was proposed in [5], where k-Means clustering is combined with decision tree learning. Malware samples were used to train the classifier. Similarly PUMA [6] compares extracted permissions from an APK and compares them to known malware samples provided by the VirusTotal online security tool. Crowdroid [7] also uses k-Means to detect malware, however instead of using permissions data, crowd-sourced samples of user behavior related to system calls is used.

AndroidLeaks [15] and Stowaway [16] decompile the Java source code and look for methods that pass personal information and detect overprivilege in apps. It has been determined that many apps do not follow the least path of privilege. While this is useful in detecting what data is captured from an app, it does not provide information on whether the collection of data was justified to provide functionality.

TaintDroid attempts to provide insights into how apps use and share data by providing continuous real time monitoring of data and when data leaves the phone [17]. While TaintDroid provides a good monitor of what data is leaving the phone, it does not provide any protection to prevent it from occurring.

VI. Conclusion and Future Work

In this paper, we evaluated the most popular Android apps and presented an approach to highlight outliers by using the permissions information of apps of similar functionality as inputs for the Isolation Forest anomaly detection technique. Privacy focused UI enhancements to Android were also demonstrated. By color coding the launcher icons, it can be easily seen by the user which apps are privacy invasive. The dialogs displayed when an outlier app is opened would help the user easily understand which apps are more privacy invasive.

We are currently in the process of deploying PrivacyPalisade on Google Nexus devices and conducting a user study. In future work we aim to implement a continuous monitoring solution to create a profile of each data entry an app has requested and sent to a server.

References

[1] L. Kulik, "Privacy for real-time location-based services," in *SIGSPATIAL Special. ACM 1(2), 2009*.

[2] A. Holzer and J. Ondrus, "Trends in Mobile Application Development," in *Mobilware 2009*.

[3] F. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *ICDM 2008*.

[4] S. Llamas, Chau, "Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14," 2015. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS25450615

[5] Z. Aung and W. Zaw, "Permission-based Android malware detection," in *IJSTR 2(3), 2013*.

[6] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. Bringas, and G. Álvarez, "PUMA: Permission Usage to Detect Malware in Android," in *CISIS 2012*.

[7] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based Malware Detection System for Android," in *SPSM 2011*.

[8] K. W. Y. Au, Y. F. Zhou, Z. Huang, P. Gill, and D. Lie, "Short Paper: A Look at Smartphone Permission Models," in *SPSM 2011*.

[9] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *SOUPS 2012*.

[10] 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 *CCS 2010*.

[11] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling Users' Mobile App Privacy Preferences: Restoring Usability in a Sea of Permission Settings," in *SOUPS 2014*.

[12] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android Permissions: A Perspective Combining Risks and Benefits," in *SACMAT 2012*.

[13] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and Purpose: Understanding Users' Mental Models of Mobile App Privacy Through Crowdsourcing," in *UbiComp 2012*.

[14] W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," in *CCS 2009*.

[15] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale," in *TRUST 2012*.

[16] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android Permissions Demystified," in *CCS 2011*.

[17] W. Enck, P. Gilbert, S. Han, V. Tendulkar, 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 *TOCS. ACM 32(2), 2014*.