International Journal of Security and Its Applications
Vol. 10, No. 3 (2016), pp.223-232
http://dx.doi.org/10.14257/ijsia.2016.10.3.21

Android Permissions Management at App Installing

[1]Sujit Biswas, [2]Wang Haipeng and [3]Javed Rashid

[1]Computer Engineering Department, Jessore Polytechnic Institute, Jessore, BD
[1,2,3]Computer Science School, Northwestern Polytechnical University, China
sujitedu@gmail.com, haipeng.wang@gmail.com, javed_iiee@yahoo.com

Abstract

Android based smartphone users' privacy has been a hot issue recently in public concerns due to various instances of security attacks and privacy leakage on Android platform. Android existing security has been built upon a permission based mechanism which restricts critical resources accesses of third-party Android applications. This permission based security system is widely assessed for its major control of application permissions and critical management of permissions by developers, marketers, and end-users. Considering the critical management of permissions some previous research papers proposed automatic permission management tools. But those automatic tools never assessed end user's expectation about permissions perfectly. This paper presents a tool, SDroid (Secured anDroid) that assesses the best permissions management based on end users opinion. SDroid evaluates requested permissions and allow users selectively grant permissions considering his/her knowledge level depending on the opinion.

***Keywords**: Smartphone privacy and Security, Android Security, Over-claimed permission detection, Users permission choice.*

1. Introduction

The global smartphone user surpassed the 1 billion mark in 2012 and total 1.75 billion [1] in 2014. In 2014 there were 519.7 million smartphone users, with the number estimated to grow to 700 million by 2018 [2] only in China. The increase in the levels of convenience and features of smartphones causes a significant growth in the number of Smartphone users. Android is first comprehensive open-source mobile operating system destined toward consumer market that allows developers to create applications easily. Android based smartphones get worldwide acceptance, its knowledge base of security policies, and its openness. Because of openness of Android leads to rapid growth which certainly give boom to Android application markets. Though some apps stores are providing security and monitoring apps privacy and security threats strictly, now a days eavesdropping, spoofing and privacy leakage are very well discussion topic. Large numbers of researchers are researching to be free from these vulnerable attacks.

Android uses finer-grained permissions to allow or disallow applications or components to interact with other applications/components or critical resources. User approval is required before getting access to critical operations like to call, SMS in an application. Applications explicitly request the permissions in order to execute successfully and declare the permissions in its configuration file (AnroidManifiest.xml). When an application is installed, android prompts the user to either allow or reject requested permissions. According to the existing Android security framework users are bound to accept requested permission for installing otherwise installation aborted. Whenever the user installs the app, by means he/she approves the requested permissions of the app. As a result, Android is eligible to access the permissions related requested operations. Here mentioned that once the permissions are approved by the end user, he/she cannot revoke them but uninstall. Uninstalling is the only way to remove those permissions.

2. Significance and Background

In android each application runs as a separate process with its own data structures and prevent other process from interfering with its execution. When an application is installed on Android platform, it is assigned a unique user id (UID) and group ID (GID). Each application's self-data structure is associated with itself UID and GID. Permissions to access data structure and files are allowed only to application itself through its UID or to the supper user (root). Critical resources are restricted through permissions in Android. As mentioned before, Android restricts accesses to critical resources by permissions. These permitted permissions are stored in application's sandbox and inherited by all of the application's components. Here limitation is when an application is installing, user must accept all permissions to install otherwise he/she has to cancel the installation. There is no option to accept partial permissions. On the other hand required permissions for the app are defined by third party developers. Many developers have lack of awareness [4] about privacy and identify a number of barriers to improve privacy and security behaviors. About 56% over-privileged applications claimed an extra unnecessary permission and 94% claimed four or fewer extra permissions [5]. Developers include over permissions because of a) they tend to request for permissions with names that look relevant to the functionalities they design but do not justify whether the permissions are actually required or not; b) they may have made mistakes due to using copy and paste, deprecated permissions and testing artifacts; c) they also may have intension advertising like Google AdSense or any illegal target. Android user faces four types of permissions. These are as follows

1. Normal- low risk permissions. It allows an application to get access of API calls.
2. Dangerous- Allows the application to perform certain operations that can cost the user money or data. This is high risk permissions.
3. Signature- Granted only if the application signed with the same certificate as the application that declared permissions.
4. Signature or System- Granted only if its requesting application is in the same android system image or signed with the same certificates as those in the system image.

Android installer shows required permissions including over-claimed permissions of the app at installation before install. But most of the end users don't care about these information but they are scared about self-privacy. Behind reason for avoiding a) most of them have no depth knowledge about such permissions information and its future effect; b) don't care about privacy; c) information presented with too much technical words. As a result all permissions including over-claimed permissions are allowed by user self. Furthermore this is very cumbersome job to create awareness this huge number of user. Considering all phenomena this paper is going to propose a tool SDroid which will receive a single opinion from user about the app before installation. Based on the opinion SDroid will take smart decision about permission during installation stage.

3. Previous Works

Stowaway [7] is a tool that automatically detects over-claimed permissions in compiled application. MockDroid [8] allows to forward fake data when an application tries to retrieve device ID. It also allows revoking over-claimed permissions. Like MockDroid AppFence [9] sends a fake data instead of original in critical access period. It also blocks network transmission of data. Tissa [10] defines private mode that enables user to control permissions at runtime. PScout [11] is a static analysis tool which captures permission requirements for every API call by examining entire source code. Flex-p [12] manages permission system. It allows users to change any permission at any time even after installation. Because of advertising library over claimed permission problem is allowed mostly. New advertising API has been proposed in AdSplit [13] that allows certain permission for advertising without requesting privacy sensitive permissions. Aurasimum [14] works as middleware between OS and app. It monitors all services like internet, contact list etc. It maintains allowing or denying and also monitors I/O. Famous research prototype TaintDroid [15] tracks information flows by labeling data from privacy sensitive sources. If any sensitive resources information is going to leak by any application, it alerts user. In Soundcomber [16] a safe Trojan is proposed that can extract private information from audio sensor. Woodpecker [17] blocks apps with the help of Soundcomber that access audio data. QUIRE [18] tracks and records the specific Inter Component Communication (ICC) call chain transparently. Also recipient can monitor remote access chain. If any unauthorized chain shows, it can cancel the access. IPC [19] inspect mitigates confused deputy attack by reducing permissions. It checks whether apps acts as media between recipient and requester's permission. Sorbet [20] allows user to define policies to mitigate undesired deputy attack. Eclipse IDE plugin [21] assist developers to specify minimum set of permissions required for an application. Analyze source code and inform about minimum required permissions. Kirin [22] and Saint [23] work at installing stage and maintain permissions by own security policy. Kirin used a prolog based database for storing the policies.

3.1 Analysis of Previous Works

This paper proposed a framework that will check over-claimed permissions at installing stage based on user purposes. Apex [6], Kirin [22] and Saint [23] are such type of research prototype that works at installing stage. Apex allows selectively grant permissions at installation by Poly (customized installer) but to take decisions about right permissions are very complex job for end users while most of the users have no knowledge about permissions. Similarly, Kirin and Saint both used a security policy for automatic decision at installation. Kirin used a prolog based security policy on the other hand Saint used a security rules, stored in database. Both tools don't allow selecting permissions for advanced users. These three tools never collect users' opinion about service expectation from the app. Kirin ensures the downloaded application or .apk are complying predefined security policy. If any requirements don't comply, installation process will be canceled. Here predefined security policy is the main key to ensure security decision. For security policy P they proposed $P(s,o,r)=requires(o,r) \cap has\_permission(s,r)$ Where $s \in S$ (S is set of applications in System), $o \in O$ (O is the set of all components in the system), $r \in R$ (R is the all permission label in the system). Condition is requires (o,r) and has_permission(s,r) must be true. My argument with Kirin and Saint let ContactProvider is an app where s = ContactProvider, R={INTERNET, CAMERA, READ_SMS, CONTACT}, O= {CALL_PHONE, camera} Requires (camera, CAMERA)^ has_permission (Contact Provider, CAMERA) is true. But usually no need Camera for Contact Provider app. Here, Contact Provider is claimed Camera as over-claimed permission. The main goal of this research is to identify over-claimed permissions, inform user about those and selectively allow those permissions before installation.

4. Proposed Tool

4.1 Background of this Proposal

According to the survey [24] of Kathy Miller in December' 2012 about 64% users have never installed security software or apps on their device in order to make it more secure from viruses or malware. So, the maximum numbers of smartphone users are completely unaware about security. At install time some permission are asked to users that are sensitive for their security and privacy. The users don't like to share but ignore. This paper assume that users know at least which purposes they are going to use the app but not concerned about its related permissions or don't want to know about that security mater (he/she has no time to read documentation, just wants to run that app). SDroid is a tool which selectively allows sophisticated users (who has idea about permissions) to grant or deny the permissions. On the other hand it allows a default secured permission set for Naïve users (have little knowledge about permissions but like to avoid) and General users (have no knowledge about permissions) considering app activities.

4.2 SDroid Architecture

Basic framework architecture of SDroid has been depicted in Fig 1. SDroid ensures install time security of apps. In SDroid, a custom package installer has been used. The SDroid installer maintains installing decision depending on security policy. The security policies are stored in the SQLite database. Decision maker decides about installation considering four issues. Like, 1) Which permissions are required for the app (the SDroid-enhanced Android installer retrieves the requested permissions from the manifest file) 2) App category or user's choice (in which purposes user wants to use the app) 3) What are the default permission set for the category. 4) Is there any unmatched permission (Decision maker compare between default permission set & required permissions) is member of dangerous permissions.

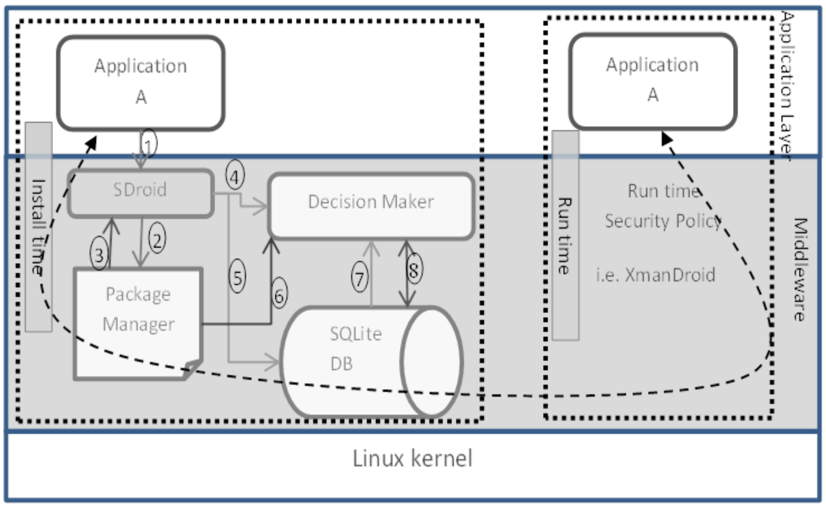

Figure 1. SDroid Architecture

According to the Fig1, SDroid's customs ApkParser parse the .apk file (Step 1) and collects required permissions from *AndroidManifiest.xml* of packages (Step 2, 3). Users choose purposes (App type) send to decision maker (Step 4) and also send to SQLite database (Step 5). Then collected required permissions are sent to decision maker (Step 6). The purpose related permissions set (suggested secured permission set) is collected by decision maker from DB (step 7). If decision maker observe that all required permissions are not exist in default permissions set but exist in dangerous permissions list then SDroid installer shows the unmatched permissions as warning with default disallow, otherwise installation continue.

4.3 Application Installer

In SDroid a customized installer (modified version of Android's application installer) has been used. The tool browse .apk file from secondary storage (sdcard) and sends the file path to the installer. SDroid installer receives the path and parses the package through *ApkParser* and collects declared permissions from *AndroidManifest.xml*. Simultaneously it collects user's opinion (Apptype / purposes) about why he/she wants to use the app. By using the single user's feedback, the tool takes the best compatible security decision and installs the app. Security decision process has been discussed in *technical details* section.

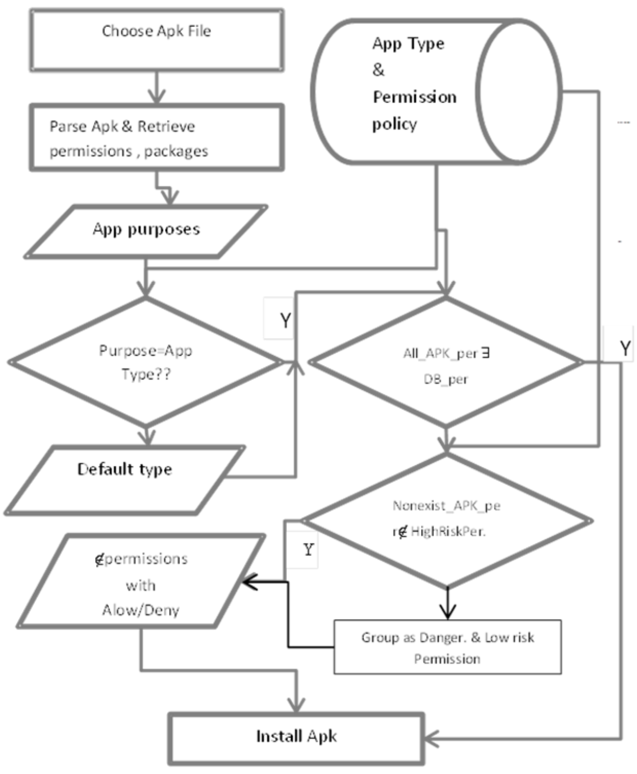
Choose Apk File
App Type & Permission policy
Parse Apk & Retrieve permissions , packages
App purposes
Purpose=App Type??
Y
All_APK_per ∃ DB_per
Y
Default type
Nonexist_APK_per ∉ HighRiskPer.
Y
∉permissions with Alow/Deny
Group as Danger. & Low risk Permission
Install Apk

Figure 1. SDroid Work Flow Diagram

4.4 Technical Details

SDroid follows three cases which may happen at installation. Based on knowledge level of users security policy will be selected which has been discussed below. Working flow diagram of SDroid has been shown in Fig2. For instance, the user wants to install an application to play Game (G), [$p \in P$] P is the set of required permissions in s; Pd is default permission set for type Game which are stored in SQLite database. Here Game is an app purpose. Three types of cases may form on that situation.

Case 1: *When*$(\forall p \in P)\exists\, Pd)$ *app will be installed without any warning.* That means all permissions string of s are exist in Pd. This is the best case of security policy for installing.

Case 2: When $(p \in P)\nexists\, Pd)$ but $p' \in Hp$ then P' will be displayed as dangerous and $\{p' - (p' \cap Hp)\}$ as low risk permission checked with disallowed (default) and installation continue. Here p'unmatched permission of P and Hp is is set of high risk permission preferences. That means, if any/some permissions are not exist in default permission set (Pd) but exist in high risk permission set (Hp), then it shows dangerous permissions in one group $(\mathrm{p}' \cap \mathrm{Hp})$ and other permissions $\{\mathrm{p}' - (\mathrm{p}' \cap \mathrm{Hp})\}$ as low risk permissions. Here mentioned both Dangerous and Low risk permission groups will be presented with default

denied. As a result sophisticated users will read the warning and take decision smartly (able to allow any permission), on the other hand for naïve and general users can continue without any decisions for that purposes warning permissions will be denied by default.
Case 3: When user (General user) doesn't choose/ purposes are not matched with app type, default type is selected automatically and run operations like case 1 and 2. Permission set is available for default type in database.

5. Security Policy

SDroid maintains security policy basically in two steps first it checks app purposes then default set of permission for the each purpose.

5.1 App Purposes

User purposes indicates that why he/she wants to install the app. These are the main key for defining security decisions. Every level of users should know why he/she is going to install the app. In this proposal total security decisions is coming from database based on user's purpose choosing. Google already defines two major categories for the programs in their market like *Game* and *Application*. Applications are also categorize as 26 and Games as 18 sub categories [25]. In Google play these categories has been used for convenient searching the app. In this paper the categories have been used for security definer. To be more user friendly the label has been changed. Like Social is a category of GooglePlay which holds all social communications categories application like Facebook, Wechat, QQ etc. SDroid presents Social as Social Communication. It will help better understanding to ordinary user. This paper represents some categories as a sample. But I believe that more standard and specific categories could be finalized by advanced research or statistical analysis of existing apps of different store.

5.2 Default Permission Set

This paper proposes default permission set only for Social communication categories app as instance which is mentioned in Google Play as Social. For defining a default permission set for social communication, two steps analysis process has been followed in this research a) collecting smartphone users' opinion b) analysis of seven high rating and most popular Social apps' required permissions. Seven social popular applications (FaceBook, Wechat, QQ, Twitter, LinkedIn, Instagram, SnapChat) have been chosen and ran reverse engineering process to retrieve its permissions from Androidmanifiest.xml. It has also been rated its requested permissions as Critical, High, Medium and Information disclosure depending on its activity related with personal information. Based on survey and obtained required permissions of Social apps this research selected & showed default and dangerous permission set for SOCIAL_COMMUNICATION type (Table 1) by yellow and red background respectively. It also proposed a dangerous permission set for the same category. Table1 shows the default permissions and dangerous permission set. The selection procedure of these two permissions set has been discussed in section 5.3

5.3 Default Permission Set Selection Process

As mentioned section 5.2, this paper presents default permissions set for Social communication apps as an example. Default set has been finalized considering three issues a) collecting smart user's opinion about services requirements b) analysis of seven high rating and most popular Social apps' requested permissions and c) required permissions' relationship with privacy. The authors conducted structured interviews by 24 questions on 113 multinational people who used social media apps. Most of the participants lived in China from different countries and continue research on different Engineering sectors. Main target was to know actually why they use social media app and what type of services they expect from those apps. Required permissions of seven popular apps have been collected through ApkTool.

Permissions	FB	QQ	We Chat	Twitter	Linked In	Snap Chat	Insta-gram	Users positive Response	Rating
CALL_PHONE	√	√						63.60%	4
CAMERA	√	√	√	√		√	√	68.60%	4
INTERNET	√	√	√	√	√	√	√	90.10%	3
READ_CONTACTS	√	√	√	√	√	√	√	68.60%	3
READ_EXTERNAL_STORAGE	√							57.10%	3
RECORD_AUDIO	√	√	√	√		√	√	60.00%	3
WRITE_CONTACTS	√	√	√		√			68.60%	3
WRITE_EXTERNAL_STORAGE	√	√	√	√	√	√	√	68.60%	3
ACCESS_COARSE_LOCATION	√	√	√					68.60%	2
ACCESS_FINE_LOCATION	√	√	√	√	√	√	√	68.60%	2
ACCESS_NETWORK_STATE	√	√	√	√	√	√	√		2
READ_SYNC_SETTINGS	√		√	√	√				2
RECEIVE	√		√	√	√	√	√		2
ACCESS_WIFI_STATE	√	√	√	√		√			1
WAKE_LOCK	√	√	√	√	√	√	√	25.70%	1
SYSTEM_ALERT_WINDOW	√	√	√	√					1
VIBRATE	√	√	√	√	√	√			1
WRITE_SYNC_SETTINGS	√	√	√	√					1
INSTALL_SHORTCUT	√	√	√	√					1
MODIFY_AUDIO_SETTINGS	√	√	√			√			1
AUTHENTICATE_ACCOUNTS	√X	√X	√X	√X	√X			5.70%	4
DOWNLOAD_WITHOUT_NOTIFICATION	√X		√X					17.10%	3
GET_ACCOUNTS	√X	√X	√X			√X	√X	17.10%	3
MANAGE_ACCOUNTS	√X	√X	√X	√X	√X			25.70%	3
NFC		√X						6.50%	3
READ_CALENDER	√X	√X			√X			48.60%	3
READ_FRAME_BUFFER							√X	25.10%	3
READ_LOGS		√X						14.10%	3
READ_PHONE_STATE	√X	√X	√X	√X	√X	√X		14.10%	3
READ_PROFILE			√X	√X	√X	√X	√X	35.10%	3
READ_SMS	√X	√X	√X					13.10%	3
RECEIVE_SMS				√X		√X		13.10%	3
SEND_SMS		√X				√X		42.90%	3
BLUETOOTH		√X	√X					13.50%	2
BLUETOOTH_ADMIN		√X	√X					13.50%	2
BROADCAST_STICKY	√X	√X	√X		√X				2
CHANGE_CONFIGURATION		√X							2
DISABLE_KEYGUARD		√X							2
GET_TASKS	√X	√X	√X					34.30%	2
KILL_BACKGROUND_PROCESSES		√X							2
PERSISTENT_ACTIVITY		√X							2
READ_SETTINGS			√X						2
READ_SYNC_STATS					√X				2
RECEIVE_BOOT_COMPLETED	√X	√X	√X				√X		2
RECORDER_TASKS	√X								2
RESTART_PACKAGES		√X							2
WRITE_CALENDER	√X	√X							2
WRITE_SETTINGS		√X	√X						2
BATTERY_STATS	√X						√X		1
CHANGE_NETWORK_STATE	√X	√X							1
CHANGE_WIFI_MULTICUST_STATE		√X							1
CHANGE_WIFI_STATE		√X	√X					34.30%	1
EXPAND_STATUS_BAR	√X								1
FLASHLIGHT		√X		√X		√X			1
GET_PACKAGE_SIZE			√X						1
READ_GSERVIES	√X			√X					1

SET_ALARM			√X						1
SET_WALLPAPER	√X								1
SET_WALLPAPER_HINTS	√X								1
USE_CREDENTIALS			√X	√X					1
GET_ORIENTATION							√X	√	

Table 1: Popular Social Apps permissions, Users opinions, (□) indicates requested and Accepted permissions and (□X) indicates requested but denied by default. 'Users' indicates positive response to allow the permission

5.4 Default Permission Set for Social Communication Purposes

This research investigates that top seven social communication apps are used for almost same purposes but only seven permissions has been used commonly. In Table 1, mentioned required permissions of top most popular seven social communication apps. In the column 'user', it has been mentioned users opinion about their required services. Top most right column indicates the security and privacy rating of the permission. Security rating is based on how much the permission is related to privacy of users. Say for example CAMERA can share users' photo and video which directly related to users' privacy as a result it is high (4 out of 4) in rating. On the other hand VIBRATE basically works for creating vibration there is no any other services of it. So, it is not related with privacy at all. So, it is low (1 out of 4) in rating. As mentioned before users opinion has been collected through a simple question. But we also have to keep in mind that users don't have enough knowledge which permissions are mandatory for better service but not too much risky for privacy. That's is why I also select some permissions which users don't want to allow but mostly social apps used for better services to user, on the other hand those are rating are low. For instances users don't want to allow WAKE_LOCK but all apps are used besides its rating is one; So, WAKE_LOCK is considered positively. The permissions which are considered positively by 50% or above users, has been accepted as default. There are some permissions for which didn't get users opinion but defined as default based on using statistics and rating. Like ACCESS_NETWORK_STATE is default permission. Because, through ACCESS_NETWORK_STATE permission apps collect information about networks and 100% social apps has been used this, contrary its rating is two. Dangerous permission set are defined also based on two factors. Firstly, 50% or more users are disagreeing to allow, secondly privacy security threat rating is high. Default permission set for default purpose will be defined considering dangerous permissions of all purposes. This is the future works of this research.

6. Result Analysis

For testing the SDroid services it has been downloaded and installed Facebook, Twitter, QQ, Wechat, LinkedIn Instagram & Snapchat through SDroid. Like when apk of Facebook has been chosen, it asked a purpose and after choosing purposes it shows all required permissions where dangerous permissions are denied and others permissions are accepted by default consulting with SDroid security policy. For the case of Facebook installation, SDroid offered selectively allow almost 18 Android official permissions and 21 unknown permissions. Considering the worst case users just continued installation without any allowance. As a result 29 out of 61 permissions denied by self. It was the challenging that after denied those large numbers of permissions what will be the performance of that app. But interestingly, it was run successfully and also provided all expected services of general and naïve users. Here mentioned that expert users (sophisticated users) may allow any default denied permissions before installation.

6.1 Statistical Analysis

To verify the co-relation in between social communication apps and social communication apps versus another app (like WPS office) Pearson correlation method has been used based on weight of these apps permissions. This correlation is verified in three steps. A) Correlation in between apps before using SDroid B) Correlation in between apps after SDroid policy applied C) Correlation between the app before applied SDroid policy and after applied. The lowest positive correlation value between the apps is 0.483. On the other hand highest positive correlation value between WPS office (other purposes) and others social communication apps is 0.176. This means that the selected social purposes are same categories. Those apps installed through SDroid installer and collected accepted permissions weight by reverse engineering process. The lowest positive correlation value between the social communication apps is 0.531. It means that after SDroid policy on the apps, have still very positive correlations. Their working behaviors have also no significantly changed.

Table 2. Correlation in between App before Applied SDroid Policy and after SDroid Policy

	SWFB	SWQQ	SWweChat	SWTwitter	SWLinkedIn	SWSnapChat	SWInstagram
WFB	.603**	.520**	.468**	.412**	.334**	.471**	.483**
WInstagram	.563**	.491**	.611**	.701**	.454**	.751**	.782**
WLinkedIn	.304**	.240*	.388**	.343**	.621**	.332**	.349**
WQQ	.424**	.509**	.289*	.245*	0.179	.319**	.342**
WSnapCht	.527**	.458**	.573**	.653**	.413**	.720**	.710**
WTwitter	.474**	.406**	.583**	.685**	.425**	.646**	.655**
Wwechat	.432**	.362**	.541**	.487**	.401**	.483**	.496**
**. Correlation is significant at the 0.01 level (2-tailed).			*. Correlation is significant at the 0.05 level (2-tailed).				

Table 2 shows the very positive correlation between the app before applying SDroid policy and after SDroid policy. It means that even after removing large number of permissions from the app by SDroid installer, they have very positive relationship. Their working behaviors have also no significantly changed.

7 Limitation and Future Works

This works have several limitations. SDroid security policies controlled based on app purposes and related default permission set. For categorizing the app purposes this paper considered only GooglePlay store categories. Only some apps have been considered for defining default permission set. Every app has some unique properties, this paper considered a group of apps behavior as coarsely. Till now it will work only at installation stage. Future works for this project is to fix up standard purpose list and its related permission set.

8 Conclusion

This paper complements existing mobile app privacy research for protecting over-claimed permission at install-time. SDroid probably be used to significantly alleviate user burden from complex permission decisions. It collects people mental opinion considering activities or functions of the app and ensures users best privacy settings. Operational steps of this tool will help different level of users taking decision about installation.

References

[1] E marketer, Smartphone users worldwide will total 1.75 billion in 2014, http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536, **(2015)**.

[2] S. Millward, "China now has 520M smartphone users, will top 700M by 2018", Tech in Asia, https://www.techinasia.com/china-520-million-smartphone-users-2014/, **(2014)**.

[3] A. Mylonas, A. Kastania and D. Gritzalis, "Delegate the smartphone user? Security awareness in smartphone platforms", Computers & Security, **(2012)**.

[4] B. Liu, J. Lin and N. Sadeh, "Reconciling Mobile App Privacy and Usability on Smartphones: Could User Privacy Profiles Help?", Proc. of the WWW, **(2014)**.

[5] A. P. Felt, E. Chin, S. Hanna, D. Song and D. Wagner, "Android permissions demystified", Proc. of ACM CCS, ACM; **(2011)**.

[6] M. Nauman, S. Khan and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints", Proc. of ACM ASIACCS, ACM, **(2010)**.

[7] A. P. Felt, K. Greenwood and D. Wagner, "The effectiveness of application permissions", Proc. of USENIX WebApps, USENIX Association, **(2011)**.

[8] A. R. Beresford, A. Rice, N. Skehin and R. Sohan, "Mockdroid: trading privacy for application functionality on smartphones", Proc.of HotMobile, ACM, **(2011)**.

[9] P. Hornyack, S. Han, J. Jung, S. Schechter and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications", Proc. of ACM CCS, ACM, **(2011).**

[10] Y. Zhou, X. Zhang, X. Jiang and V. W. Freeh, "Taming information-stealing smartphone applications (on android)", Proc. of TRUST, Springer; **(2011)**.

[11] K. W. Y. Au, Y. F. Zhou, Z. Huang and D. Lie, "Pscout: analyzing the android permission specification", Proc. of ACM CCS, ACM, **(2012)**.

[12] K. Mueller and K. Butler, "Poster: Flex-p: flexible android permissions", Proc. of IEEE S&P, **(2011).**

[13] S. Shekhar, M. Dietz and D. S. Wallach, "Adsplit: separating smartphone advertising from applications", Proc. Of 21st USENIX Security Symposium, **(2012).**

[14] R. Xu, H. Saı¨di and R. Anderson, "Aurasium: practical policy enforcement for android applications", Proc. of USENIX security, USENIX Association, **(2012)**.

[15] W. Enck, M. Ongtang and P. McDaniel, "Mitigating android software misuse before it happens", In networking and security Research Center, Technical Report, The Pennsylvania State University.

[16] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia and X. Wang, "Soundcomber: a stealthy and context-aware sound Trojan for smartphones", Proc. of NDSS, **(2011)**.

[17] M. Grace, Y. Zhou, Z. Wang and X. Jiang, "Systematic detection of capability leaks in stock android smartphones", Proc. Of NDSS, **(2012)**.

[18] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu and D. Wallach, "Quire: lightweight provenance for smart phone operating systems", Proc. of USENIX security, **(2011)**.

[19] A. Felt, H. Wang, A. Moshchuk, S. Hanna and E. Chin, "Permission redelegation: attacks and defenses", Proc. of USENIX security, **(2011)**.

[20] E. Fragkaki, L. Bauer, L. Jia and D. Swasey, "Modeling and enhancing android's permission system", Computer Security ESORICS Springer, **(2012)**, pp. 1-18.

[21] T. Vidas, N. Christin and L. Cranor, "Curbing android permission creep", Proc. of the Web, vol. 2, **(2011)**.

[22] W. Enck, M. Ongtang and P. McDaniel, "Mitigating android software misuse before it happens", In networking and security Research Center, Technical Report, The Pennsylvania State University.

[23] M. Ongtang, S. McLaughlin, W. Enck and P. McDaniel, "Semantically rich application-centric security in Android", Proceedings of Annual Computer Security Applications Conference (SCSAC), **(2009)**.

[24] K. Miller, "October is national cyber security awareness month: how secure are you?", http://blog.globalknowledge.com/technology/security/hacking-cybercrime/october-is-national-cyber-security-awareness-month-how-secure-are-you/, **(2015)**.

[25] Google play, https://play.google.com/store?hl=en&tab=w8, **(2015)**.