Negotiation as a Collaborative Tool for Determining Permissions and Detection of Malicious Applications

Rabia Riaz[1], Mubashar Ahmad[3], Sana Shokat[4]
Department of CS & IT
University of Azad Jammu and Kashmir, Muzaffarabad, 42714, Pakistan

Sanam Shahla Rizvi[2]
Raptor Interactive (Pty) Ltd
Eco Boulevard, Witch Hazel Ave, Centurion 0157, South Africa

Se Jin Kwon[5*]
Department of Computer Engineering
Kangwon National University, Samcheok, 25806, Republic of Korea

***Abstract*—In Android OS users find it very difficult to understand and comprehend its permission mechanism. Frequently, users tend to ignore permission negotiations dialogs during installation of an application. Users, who pay attention to the permission negotiation dialogs, find it tough to comprehend the description and evaluation of permission procedure. They do not know the impact of granting these permissions on their data. One major issue is that user is unaware about how application uses their data. He has no insight after granting permission to the application and effect of these permissions on his data's privacy and security. This research reveals that discrete permission settings are helpful for user to secure his device resources and data. This study uses a distinct technique to detect danger of unnecessary permissions. It helps end users of Android OSs to understand the problems and provides them batter way to deal with the problems and grounds to explore alternatives.**

Keywords*—Collaborative learning; intrusion detection; mobile applications; information security; web based learning

I. Introduction

Smart phone is becoming most widespread device in the world used for different purposes. The cloud storage is easily accessible, proficient for performance and data storage makes mobile as a primary computer for numerous end users. Many mobile operating systems (OSs) and platforms have become popular during past few decades. The integration of mobile phone, internet, and peculiar computation has resulted in the appearance of the smart phone. Symbian's, iOS, and Android are very popular OSs. The demand of smart phone is increasing every year. Universal growth in the consignment of the smart phone is approximately 17% each year.

Android is most widely used OS in mobile phones because of its open source frame work. Its open source makes its customization easy for smart phone companies to accomplish their features and pricing requirements for their targeted group of users. It is a feature rich platform for application developers for building applications. It is the OS for large number of smart phone devices in more than 192 countries all around the world. Its growth is increasing day by day. Well renowned brands have started using Android in their smart phones. In this study, we focus on Android due to its abundant use all over the world. Main reason for increasing use of Android OS is that it is very easy to use, even common less educated users can use and comprehend its functionalities easily. Users can easily install applications for Android from different market places. It is very easy to install any third party application without any major modification in the Android frame work to fulfill ones need and requirements while other platforms do not provide such freedom. Android is open source which has more fairness to software developers and end users mainly focusing on three key philosophies: durability, clearness and ownership. For software level, they aim to enhance durability by supporting their product for as long as possible. Android's goal for transparency includes making software as open as possible, making users more aware of the data they are sharing, and giving them more control over their applications. It is believed that the phone is yours, and your data belongs to you.

The security features of Android applications are components protection, type safety, permissions and memory management unit [1].

Centralized approach for end users allows them to search applications quickly to satisfy their needs however, this is useful for distributors also. It is devastating for the operators of this open market to accurately scrutinize their genuine intent of applications uploaded onto the marketplace. Applications of third parties run on Android and can increase the risk of security [2]. Therefore, centralized market has made an environment easy for malware developers to exploit many potential victims. Several smart phone applications like Google Play Store, which is most popular of them, provides end users a choice of either programmed or manual updates for the applications that are present in market.

Operating systems have more aggressive permissions than desktop apps and can increase complications in user's decision taking procedure about smart phone applications updates [3]. People can use your GPS to estimation for example what time you are not at home; this fact can put you in danger. By default some applications gain read and write access to the address book, which permits hackers addition of a reprobate email address to existing email addresses and receive email correspondence. At present, there are several Android applications in market that uses the Android smart phone as a security surveillance camera, e.g. Android Eye, these malicious camera applications can periodically check the screen state and run the stealthy video recording only when the screen is off,

[*]Corresponding author

which means that the user is not using the phone and the camera device is idle [4].

In this paper, the proposed technique will check additional permissions at application installing time. Some techniques have already been proposed during recent researches that work at installation time of an application [5-6]. Customized installer Apex allows to selectively giving permissions at installation stage [7]. However, it gives right of decision to the end users which is very complex for end users because of lack of comprehension of the permission systems. In the same way, Kirin and Saint both proposed a security policy for making automatic decision at the time of installation [6] [7]. Security policy on the bases of Prolog is used by Kirin and Saint using security rules stored in database. Both these techniques do not give permissions for advanced end users. These tools by no means collect end users views about expected service from the applications.

Current permission systems are ineffective due to two main reasons; firstly most of the users of Android OSs are inexperience and secondly the description of permission is very technical for novice users. They do not understand the whole process of granting permissions. This research provides user flexibility for sharing their date. Users can manually allow or deny permissions for an application. To understand the complexity of the permissions descriptions, risk factors are calculated using risky permission combinations. Proposed frame work increases the understanding of the users regarding to the permission mechanisms used in Android OS. This framework also helps to find malicious applications in Android market which uses excessive permissions.

Rest of the paper is organized into different sections. Section II presents literature review. Section III highlights the methodology. Proposed algorithm is presented in Section IV and proposed application is discussed in detail in Section V. Section VI presents the design of survey questions and Section VII consists of results and discussion along with comparison of our proposed scheme with different applications vulnerable of different attacks. Section VIII and Section IX consist of findings of user survey and analysis, respectively. Conclusion and future directions are highlighted in Section X.

II. Literature Review

Permissions of Android are used basically for two purposes. Primarily OS used them to access certain functions of an application and secondly is to convey that information to the user. A researchers group at Berkeley formed a software package to map the permissions an application is requesting for to the permissions that are in fact presented to users [8]. The objective was to determine if apps are using characteristics that they are not revealing to the users in the permission rules. The research found that about one-third of the 940 applications the researchers considered were using permissions that were not making known in the permissions offered to users. The researchers found that in most cases the applications were only missing a few permissions in their disclosures and as a result came to the conclusion that in most cases these are probably simply errors of documentations. Their results show that applications generally are over privileged by only a few permissions, and much extra permissions can be attributed to developer's confusion. This indicates that developers attempt to obtain least privileges for their applications but fall short due to application programming interface (API) documentation errors and lack of developers understanding. The researchers concluded that application developers most likely misunderstood the connection between the functioning of the application and disclosing those functions to the users [9]. A further question is that even if the permission policies were constructed correctly, do users actually read or understand them?

Since 2008 updates for Android OSs are apparently regular, compared to their PC matching part as there have been twenty five firm issues. Over the air (OTA) fresh version brings up to date meaningfully variations in the standing version by adding and amending large quantity of files through Android platform certifying uprightness of current user data and applications [10]. New variety update is assisted from end to end a service called package management system. Previous research completed a broad revision of pileup susceptibilities that can be misused by malware applications in case of new version improvements. For example, older version can declare dangerous permissions in AndroidManifest.xml that has been introduced in next version. All through in update process, Android does not ask users to verify newly active permissions in that current application and awards them automatically. Thus, it affects the security of the Android device.

A programmer implements permissions inside the API package and for the duration of application installation all of these permissions are presented to end users. On the bases of those presented permissions, end users have to make decision either the application is malevolent or benign. It seems big policy fault to ask the users decide the nature of applications by purely looking permissions used by the application.

The fundamental point is that the users are not the only stack holders but they are simply a part of the entire process. End users just scan application permissions where the critical goal is to filter the application at a number of layers. It looks good decision but the most important demand is whether the layer is operative in its job. Basically adding a number of safety layers could not do anything without those layers are carrying out their tasks. Previous studies mostly estimate user's responsiveness and understanding of concerning permissions [9]. From study they tested if end user gives any consideration to Android permissions in advance to installing an application, they also tested that the user can comprehend how these permissions link to an application license. They viewed very low proportion of consideration. Many of the end users are totally heedless of permissions. Those who has knowledge of permissions to some extant also does not pay any consideration and saw a very small fraction of comprehension. Those end users who noted permissions while installation achieved superior in comprehension than rest of the others. They determined that the mainstream of Android handlers do not give consideration to or know permissions cautions.

Two most interesting facts observed are that those users who have more understanding of permissions installed applications from Google play store, while others used unofficial application stores. Secondly they are not aware of security consequence [9]. Assuming these two situations, we have enthusiasm in that the end users will show extra consideration to permissions if they are conscious of security consequences.

The results of previous studies do not show complete failure or success of the current securities policies. Some of the participants of the studies understand the permissions, but they are very small in numbers, and are likely to proliferate with time, awareness and education [11-14].

In this research, our main focus is to increase the comprehension of permission mechanism of Android applications. Users do not understand the permissions usefulness by their observations only. This research calculates risk factor involved in permissions of an application during installation, where many permissions are not risky when they are used in a package alone. But they become risky when they are used in combination with other permissions. This research investigates combination permissions.

III. Methodology

The proposed research technique consists of two main steps. In first step, we compare common take it or leave it approach with the collaborative permissions approach. In this step, we also detect the malevolent Android applications using permissions change procedure during updates.

Second step of this research calculates risk factor of Android third party applications by using combination of permissions to enhance the knowledge of end users. Mostly permissions are not risky when these are used alone by any application. For example access contact list permission is not risky when any application uses this permission without using any network group permission (e.g. internet permission). It will become risky when both internet and access contact list permission uses at the same time as a combination in an application.

The objective of the risk factor function is to provide assistance to end users of third party Android applications to decide if they want to use specific applications on their devices. It can help them for selecting any application having fewer security implications and assist them in selecting any application with other parameters such as total number of downloads, assessments and mouth references etc.

Package manager of Android permits end users to retrieve all those permissions which are requested by Android applications during installation procedure. To estimate risk factor using these permissions needs an algorithm. If risk factor of every permission is considered same than any application which has less number of permissions are more secure than others but all permissions are not risky and do not breach the security of smart phone.

Classic take it or leave technique provides less flexibility for the end users of the Android applications. It has more chances to attack by the malware using permissions oscillations and application update attacks by changing the manifest file during updating applications. To avoid such situation, hashing functions are used. For calculating hash of permissions and compare values with the previous stored values in the database, SHA512 hash function is used. For any case of mismatching in hash values, it is declared that application has a malicious intent and prompt users to take appropriate action. Users have choice to revoke permissions or continue.

This research is conducts on Android, most popular OS for handheld devices. That is a fast growing OS. Aim of this study is to make end users aware about what the excessive permissions in an application do without the users concerns. Users are not focusing on the security implications of third party application distributers. Those are taking the advantage of the user's ignorance and steal their information and share that with third parties and generate revenue for distributors and developers.

This research falls under clearness and ownership. It starts with the intention to make users clear about the risk factors that are hanging over the Android users while using third party Android distributor's applications. It then makes them able to understand security implications of Android applications. The end users can revoke Android permissions of any installed application as per their needs and desires. In current version Android permissions are categorized in to different set of permissions, user of the device has only option to allow or deny that specific set of permissions that is under the normal security tag.

Core research objective is to enhance the usability of the Android permission systems in order to increase user awareness and comprehension with respect to security and privacy. Such as:

1) How can Android permissions be translated into something that is easy to understand?

2) How to use them for raising responsiveness on security of the Android users?

3) Do these modifications raise attentiveness and comprehension in users?

IV. Proposed Algorithm for Finding Malicious Combination

In this paper, we propose an algorithm that focuses on malware detection by using excessive permissions only. We believe those combinations are risky which has any dangerous permission in it, see Algo. 1. Here *p* is permission used by Android application. *dPcomb* represent all combination in which dangerous permissions are present. To calculate risk factor we use Eq. 1 for all dangerous permission combinations.

$$Risk\_factor = [\{dPcomb / (dPcomb + N)\} * 100] \qquad (1)$$

Here *N* is the number of other permissions which are less dangerous. As an alternative of minor permissions, total numbers of combinations affect the risk factor. This research looks at combinations of permissions that can be possibly dangerous.

ALGORITHM 1. *MyPrivacy*

```
Input:   All application permissions
Output:  List of dangerous permission combinations
1.  dPcomb=0
2.  P_Comb_List←All applications having p>2
3.  For each comb P_Comb_List do
4.          if comb has dP then
5.                  dPcomb (comb) = dPcomb (comb) + 1
6.          else
7.                  dPcomb = dPcomb
8.          end if
9.  end for
```

V. Proposed Application Detail

We propose an application called MyPrivacy, it contains two main modules. First calculates hash value and compares the hash value stored in the database for detection of malicious application. Hash value is calculated using SHA512 hash algorithm.

Secondly, user might discover that an application uses a particular permission excessively using risk factor calculation. For example, an application that has access to your contacts keeps reading them out every day at noon. By using our proposed application the user might know that this behavior is unnecessary and indicates suspicious activity.

A. Interface Structure

This research is used to build an expressive consensus tool for Android OSs. It is an investigational tool which is used to detect malwares and also enhance the knowledge and understanding about the user's security of smart phone. During installing or execution of application user asked which data he wants to share.

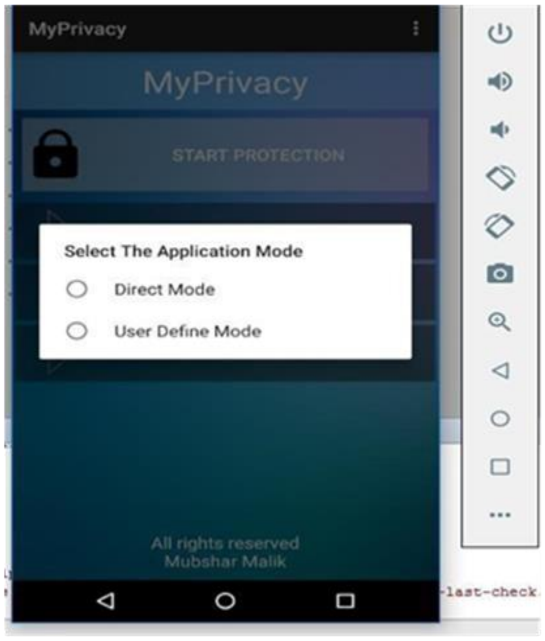


Fig. 1. Main Screen of Myprivacy Application.

The volume of data and fact is being shared affects the privacy of the user. The more volume user chooses to share the more chances of security breaches. This research chooses a permissions set that are mostly requested by applications in two events, settings and assessment of initial user study, see Fig. 1.

1) Settings: In this section privacy settings are displayed to the end user. These settings define permissions and the type of data end users are agreeing to share. The implementation of these settings is of two different modes; direct and user define.

a) Direct mode

Before Android 6.0 individuals must agree to all permissions for installation of an Android application. Users have only option to accept or deny permissions. If user denies then application does not installed. Accepting permissions is mandatory for installation of application. Installation of application is aborted without accepting permissions. User cannot change permissions, only option is to uninstall application to revoke permissions.

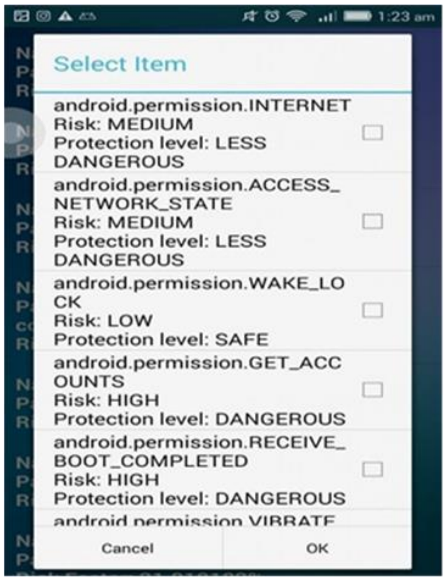
b) User define mode

Main objective of designing the customize mode is to allow end user to collaborate with OS for application permissions to access their personal data. User has option to give or revoke permissions at any time when he wants to change.

Based on the given permissions, risk factor of the application is calculated, which shows user how much his security and privacy is affected by the application which he wants to install. It makes very easy for user to decide which permission or combination of permissions is risk for his security. This mode is dual purpose. Firstly, it provides end users complete awareness of the penalties and consequences of the selected permissions. Permission settings make it solid and important to the user. Secondly, it permits to measure level without a doubt and equates different collaborative designs. Permissions selection is shown in Fig. 2.

c) Understanding and comprehension of study process

We test permissions set of common Android applications. Study procedure was comprehensible to participants and they were alert of the selections that have been asked to do. In precise, our investigational design needs that participants should consider their selections have honest privacy significances and impacts.



Select Item

android.permission.INTERNET
Risk: MEDIUM
Protection level: LESS DANGEROUS

android.permission.ACCESS_NETWORK_STATE
Risk: MEDIUM
Protection level: LESS DANGEROUS

android.permission.WAKE_LOCK
Risk: LOW
Protection level: SAFE

android.permission.GET_ACCOUNTS
Risk: HIGH
Protection level: DANGEROUS

android.permission.RECEIVE_BOOT_COMPLETED
Risk: HIGH
Protection level: DANGEROUS

android.permission.VIBRATE

Fig. 2. Permission while using User Define Mode.

d) Suitability of selected statistics

We test if the application is trustworthy and the data random sampling is a part of the application, in fact gathering possibly sensitive material that would provide participants cause to ponder twice about the probable significances of publication of data.

2) Initial user study: This research performed initial test with the core purpose to inspect three key factors:

a) Insight to participant reaction

We desire to increase primary insight into the choice building procedure of our participants. Specifically, we are interested in features of gathering data or probable concerns are taken by members as they assess. However, whether or not they desire to share certain categories of individual fragments of facts with others.

In this research, we conduct a study to investigate key factors affect the security of Android OS and end user. The gap between user understanding and permissions policy is the major risk for Android application.

VI. Design of Survey Questions

In survey, we asked ten questions to the participants about method they are using while installation of new application or during application update. Ten questions were asked in survey about method they use and prefer as well as the effectiveness of Google Play's current application installation method.

Users were inquired how frequently they install applications and method they choose for installation. Participants were provided with options such as, "I feel safe using this method," "My phone's OS does support other methods," and "For each case I use different method". These interrogations give us comprehensions about users' behaviors and how they are involved with application installation.

Participants were also inquired about their experiences with application installations principally in situations they select method or they regret on any installation.

From these user know-hows, we can judge whether existing methods of installation of applications are useful and helpful for positive decision making.

Participants inquired which elements influence the decision making procedure for application installation. In this research users are asked about the factors like author, application store, number of download, application trust etc.

These outcomes help us to gain insight in planning application updates and installation and factors we should focus on. Other information we collected are age, gender, Android experience and education.

A. Deployment of Survey

To deploy this survey, those participants were selected who fulfilled the following conditions. Age should be more than eighteen years, he or she should understand and read English language and at least 3 months of experience of Android usage with other application stores and Google play store.

B. Analysis of Survey Data

In our survey both multiple choices as well as open questions are asked by the participants, and both types of questions are analyzed contrarily. At first we analyzed answers of the multiple choice questions. Percentage of each selection is computed for getting insight about user's application installation behaviors and approaches toward application update.

For the evaluation of our application think-aloud study is used with independent participants who were requested to install application and use it. After gathering basic demographic information, we requested each participant to install 12 applications. Six applications using direct mode and rest using customized mode that has collaborative interface.

In order to provoke diverse responses we offered the participants with changing pricing setups, in which the combination or grouping of permissions was appreciated contrarily in terms of risk factor.

Participants observed similar pricing factor in both strategies and design. In each case, they interrelate interchangeably with the review settings activity.

We asked users how they install application. They were provided choice to install application using take it or leave mechanism or negotiation permission granting policy. Responses of these questions provide us insight the behavior of users that how they are involved in security mechanism of application and the factors they think are affecting user security [15-17].

Participants asked to share their experience during installation of third party applications and if they have any difficulty for installation of an Android application. Respondent were questioned about factors they think affect their decision making.

VII. Results and Discussions

For experimental purpose we built a database of downloaded Android applications from Google play store. These are 1000 prominent applications from 30 categories shown in Table I. From un-trusted sources we have downloaded cracked version of those applications to examine the accuracy of proposed technique using sample of malicious applications. Cracked versions of top paid Android applications are also downloaded. Table II shows the most frequently requested permissions.

TABLE I. Application Categories and Average Number of Permissions

Category	Average Permissions	Total number of Application
Communication	23	56
Games	4	89
Finance	2	12
Media & Video	10	43
Entertainment	9	32
Family	4	17
Photography	16	34
News & Magazines	18	76
Social	30	43
Tools	12	59
Lifestyle	12	53
Comic	9	10
Medical	15	21
Music & Audio	14	82
Education	9	78
Personalization	19	4
Productivity	18	17
Health & Fitness	17	28
Sports	10	16
Transport	13	15
Travel & Local	24	31
Shopping	15	10
Weather	11	14
Others	9	160

This research has two parts, first is detection of malicious applications, and second is to increase the knowledge of users by computing risk factors of applications and providing run time permissions for Android applications. As discussed in literature many malware families can steal data of user of mobile phones.

For detection of malicious applications, this research developed direct mode in our prototype which is designed for analysis. We installed different applications from Android play store and their cracked versions form third party application stores in direct mode. We found that detection rate of our technique is encouraging. Results of this comparison are shown the Table III. We can see that our scheme detected malicious and cracked applications in utilities with at least 91% accuracy rate. It gives maximum accuracy 98.5% in social media category.

TABLE II. Most Requested App Permission and Percentage in Play Store

Permission	What application does after installation?	Percentage of permission	Access resource
Full network access	It established a network outlet using protocols of routine network. It provides applications and browser resources to send data over the internet.	83%	Hardware
View network	information about network connections such as which networks exist and are connected	69%	hardware
Modify contents of your USB storage	It writes to the USB and allows writing on the SD card.	54%	User info
Read phone status and identity	This permission allows the application to determine the phone number and device IDs, whether a call is active, and the remote number connected by a call.	35%	User info
Precise location	Apps may use this to determine where you are, and may consume additional battery power	24%	User info

TABLE III. Application Categories and Detection Rate

Application categories	Cracked Apps	Detection rate
Social media	20	97.5%
Games	20	94%
Media and videos	10	97%
Entertainment	10	96%
Utilities	10	91%

C. Attacks

1) Update via internet: Update via internet should be through authenticated web address that is written in Manifest.xml. If web address is not correct while updating an application, this behavior shows that the specific application have piece of malicious code in it and user of the smart phone should not update it. When any change of address is appeared in Manifest.xml of package our application detects it accurately.

2) Side installation: Our technique does not detect side installation first time but when once application is installed than it detects malicious and cracked applications with 100% accuracy.

Comparison of our technique with other techniques is given in Table IV.

Second part of our research is related to user understanding of application's permissions and to increase user knowledge and comprehension.

As in the point of contact, Android OS is a three-party association among the Google, user and developers of third party applications. The role of Google in this relationship is as a moderator between the third party developer and user of the application using permissions set for every application downloaded by user. Permissions are way of demanding designers to reveal how the application will be co-operating with the handlers of device and what resources the application will access.

In ecosystem of Android, the whole liability is on the designer of the application to select the accurate permissions that define to the end user what function application is undertaking. It does not mean to say Google is completely out in this process, but the initial phase starts with the developer of the application.

After completion of application built process, select the right permissions; generate the list by which users will ultimately agree, scanning of the application for malicious code, and malware is done by Google.

Our research is mainly related to the user's role in Android ecosystem. Mostly the users who are using Android systems are unaware of the risks that are compromising the security of user devices.

TABLE IV. COMPARISON WITH OTHER SIMILAR TECHNIQUE

Name of application	**Internet update**	**Side installation**	**Pre-crack Application**
Anti-malware	No	No	No
Anomaly malware Detector	No	Yes	No
Malbee Malware Scanner	No	No	Yes
Droid Dream Malware Cleaner	No	No	No
MyPrivacy	Yes	Yes	Yes

VIII. RESULTS OF THE USER SURVEY

Suitability of specific information and participants reactions enthusiastically tied up in the collaborative procedure. Every application is asking approximately 3 to 10 permissions for their installation on smart phone device. For experimental evaluation we conducted a survey and collected the data about the users' behavior and their understanding about the security of Android permission mechanism.

We collected approximately 120 filled questionnaires from the participants of this research study. Participants showed lot of interest for decision making in the permission granting procedure during installation of an application. They were also interested in reviewing their data and permission settings. Amongst the 120 participants about 65% of them are male, 35% are female, see Table V. About 14% of them are security experts, graduates are about 46% and 40% are undergraduates and the 25.5 years are average age of the participants, see Table VI. Fig. 3 shows how the education background and experience affects the decision making process.

We found that participants are very much interested in the application installation process and want to know how have permissions effect their data and how?

Analyzing their responses, 70% of survey participant's use take it and leave it policy because of the lake of understanding of permission systems, about 15% use their applications manually, 8% use certain applications manually and certain applications automatically, 7% of them have not known the whole process but they are also interested in reviewing their data and permission settings, see Fig. 4.

TABLE V. NUMBER OF RESPONDENTS WITH RESPECT GENDER

Gander	**Number of respondents**	
	Frequency	***Percent***
Male	78	65%
Female	42	35%
Total	120	100%

TABLE VI. NUMBER OF RESPONDENTS WITH RESPECT EDUCATION

	Number of respondents	
	Frequency	***Percent***
Security experts	17	14%
Graduates	55	46%
Undergraduates	48	40%
Total	120	100%

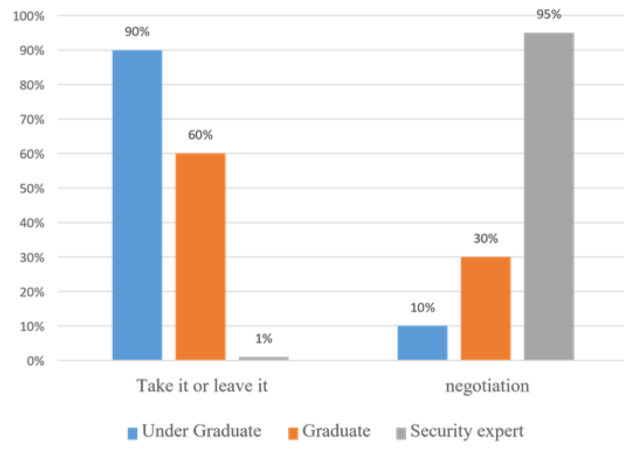


Fig. 3. Different Permission Mechanism used Percentage.

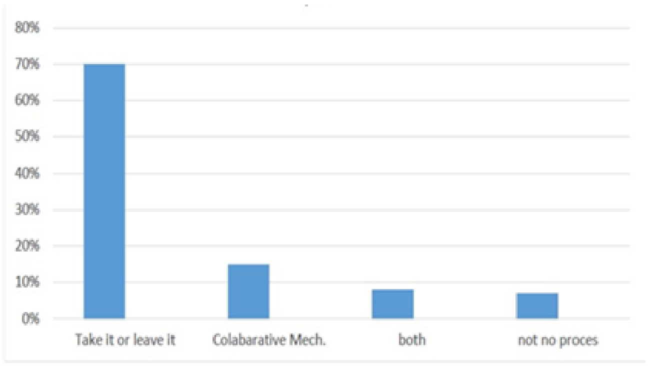


Fig. 4. Application Installation Mechanism used by the End Users.

A. *Call Logs*

Mostly the participants are very keen about the permissions related to their calls. They show lot of concerns about this permission and are not willing to share at any cost level. Some participants think that their calls are meaningless to others. But majority of the participants does not allow this permission.

B. *SMS and MMS Messages*

Maximum numbers of participants are anxious about sharing SMS and MMS messages. However, some participants do not show interest in SMS and MMS related permissions.

C. *Read and Write in Memory*

Reading and updating contents in the memory are most important permissions and participants of our study show huge interest in these. Any application permission that wants to access the storage area of the device is not granted by majority of the participants.

D. *Location*

Majority of the users are using applications that are using location information for providing them services like weather forecasting. They are comfortable to share location with server for getting services and they revoke this permission when they want.

E. *Photographs and Contacts*

Approximately all of the participants are reluctant toward photographs and contacts sharing. They show lot of reservations for sharing the photographs. They think no one have right of decision about the contact details of other people.

F. *Cookies*

No or minor concerns are raised about sharing of browsing history. Participants are comfortable to share browsing history to get the better recommendations. Although they know it is used for advertising purpose also.

IX. Users Analysis

This research observed five main observations into participant thinking throughout the study, which is although preliminary because the small user sample and period of study. Customized mode provides review of permission, which is easy for participants and allows them to instantly restrict sharing of their information and data.

A. *Minute Knowledge of Permission*

Participants have very little knowledge about permission demanded by the applications and effects of those permissions on security and privacy of the end users. Although well educated and experienced users who had knowledge of causes and effects are concerned about the security of their devices.

B. *Collaboration of Permission is Favored*

When the question is asked to the user which mechanism they prefer to use for installation and usage of applications mostly they want to use collaborative mechanism for permissions because of flexibility provided in use this mechanism. While using this mechanism they feel they have more control than ever before and feel more secure using customized mode rather than direct which is based on take it or live it policy.

C. *Increase User Knowledge*

Using combination of permissions, we calculated risk factor of applications. Application ranked malicious when risk factor crossed threshold which provides users insight of the application usage and help them in deciding process. Users become able to decide which application is malicious.

D. *Education and Experience*

Education and experience of Android affect the user security decision making procedure. Those who are highly educated make effective decisions related to their privacy and security.

Main drawback of this approach is that it shows those applications malicious which actually does not cause any malfunctioning. Quantitative results analysis of this research shows that social networking and game applications show high risk factors see Table VII. These applications share end users contact information to the server. These servers may use the information for advertising purpose.

TABLE VII. RISK-FACTOR OF INSTALLED APPLICATION

Package name	Risk-factor
Internet Browser	92.30%
Download Application	33.33%
Imo	77%
Candi Crash	89%
Face Book	85%
Contact Manager	91%

TABLE VIII. COMPARISONS WITH OTHER TECHNIQUE

System	Method of detection	Object detection	Layers	Side of working	Usage
Kirin [5]	Based on rules	Permission	1	Mobile	Lower
CrowDroid [18]	Based on behaviour	CallsofLinuxKernel	2	Server	Medium
DroidRanger [19]	Based on permission	Permission	2	Server	Higher
MyPrivacy	Based on permission	Permission and Hash value	3	Mobile	lower

Application contact manager may send your contact list to the server that affects the security of the end users. Users are very concerned about their personal information. We analyzed the applications which are freely available asks for more permissions than paid applications, free applications get about 20% more permission then paid ones. True caller is an application which gains information and stores it on its server.

The detection of risky intending is a procedure to screen applications that behave malevolently established on the exiting intention. Dependent on the desires of real situation this detection method can be reiterate over and over. Particularly, we compared and analyzed the performance of our proposed technique with other three classic malware detection systems including detection layer, detection method, resources usage and other five aspects. Here, the detection layer means the number of layers of malicious detection techniques to analyze the characteristics and usage specifies the detection techniques require using the total amount of the resources during working. As revealed in Table VIII the outcome of this comparison show that the proposed framework has obvious advantages in detecting efficiency, workload, number of layers, method of deployment and performance.

Android applications need user to approve permissions as a necessary condition for using an application. Such as hardware permissions that allow an application to fine tune the volume for user's phone. Sometime these are crucial for the primary functioning of an application. These set of permissions have inclusive effects for personal data. Some of the important outcomes of our analysis of the Android permissions include:

1) *Internet connectivity the most common permissions to get access in the smartphone*: Commonly these permissions help the applications to get access to the internet. The full network access permission used by 81% of application and permission view network connections are used by about 70% of application. The third is to get to memory access on the smart phone and fourth allows saving data to the device.

2) *The application of business and communication require more permissions*: Google divide applications into 41 diverse classes. Among them all categories business and communication classes request the maximum permissions to function.

3) *Large number of permissions are about hardware related information*: Permissions are placed into two comprehensive categories: 1) permissions for hardware functions 2) permissions for user information access. About 70 permissions could access user information whereas 165 allowed control some hardware function such as volume and vibration function or controlling flash of the camera.

X. CONCLUSION AND FUTURE WORK

Our novel technique seems better way out for finding of over privileged applications based on permissions model. It joins suitable fragments. Our technique enhances a computable appearance of the probable risk factor of an Android application.

Furthermore, the risk factor is established on intuitive division of permissions according to the risk category based on existing research and application category as well as proposed method. For working properly our technique does not require any permission or root access. Results we attained in this work considerably rest on formed rules in the suggested analysis methodology. Whole procedure was founded on independent view, which is revealed in these results. The attained results can be changed, if other constraints are used in this risk factor counting methodology with approved conditions, not the same lists of permissions, or dissimilar estimation functions. The current constraints notify the user of probable over privileged, and need his energetic contribution in identifying probable malware. Advance study in this field is necessary to improve the several parameters for a further automated over privileged and potential detection of malevolent applications in Android market places.

In future, data mining techniques can be applied for permission recommendations at the running time of an application on Android devices. This can automatically suggest permissions to the user, which can preserve user's security and privacy.

Acknowledgment

“This work was supported by Basic Science Research through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A3B04031440). This study was also supported by 2017 Research Grant from Kangwon National University (No. 000000000).”

Conflict of Interests

The authors declare no conflict of interests.

References

[1] F. Al-Qershi, M. Al-Qurishi, S.M.M. Rahman and A. Al-Amri, “Android vs. iOS: The security battle,” *in Proc. IEEE World Congress on Computer Applications and Information Systems (WCCAIS)*, 2014, pp.1-8.

[2] V. Rastogi, Y. Chen and W. Enck, "AppsPlayground: automatic security analysis of smartphone applications," *in Proc. ACM conference on Data and application security and privacy*, 2013, pp. 209-220.

[3] K.E. Vaniea, E. Rader and R. Wash, "Betrayed by updates: how negative experiences affect future security," *in Proc. SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 2671-2674.

[4] L. Wu, X. Du and X. Fu, "Security threats to mobile multimedia applications: Camera-based attacks on mobile phone," *IEEE Communications Magazine*, vol. 52,no. 3, pp. 80-87, 2014.

[5] W. Enck, M. Ongtang and P. McDaniel, "On lightweight mobile phone application certification," *in Proc. ACM conference on Computer and communications security*, 2009, pp. 235-245.

[6] M. Nauman, S. Khan, A.T. Othman and S. Musa, "Realization of a user-centric, privacy preserving permission framework for Android," *Security Comm. Networks*, vol. 8, pp. 368–382, 2015.

[7] M. Ongtang, S. McLaughlin, W. Enck and P. McDaniel, "Semantically rich application-centric security in Android," *Security and Communication Networks*, vol. 5, no. 6, pp. 658-673, 2012.

[8] A.P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," *in Proc. ACM conference on Computer and communications security*, 2011, pp. 627-638.

[9] A.P. Felt, E.S. Ha, A. Egelman, E. Haney, E. Chin and D. Wagner, "Android permissions: User attention, comprehension, and behaviour," *in Proc. Symposium on Usable Privacy and Security*, 2012.

[10] L. Xing, X. Pan, R. Wang, K. Yuan and X. Wang, "Upgrading your android, elevating my malware: Privilege escalation through mobile OS updating," *in Proc. IEEE Symposium on Security and Privacy*, 2014, pp. 393-408.

[11] R. Neissea, G. Steria, D. Geneiatakisb, I.N. Fovinoa, "A privacy enforcing framework for Android applications," *Computers & Security*, vol. 62, pp. 257-277, 2016.

[12] W. Wang, Z. Gao, M. Zhao, Y. Li, J. Liu, X. Zhang, "DroidEnsemble: Detecting Android Malicious Applications With Ensemble of String and Structural Static Features," *IEEE Access*, vol. 6, pp. 31798-31807, 2018.

[13] S. Arshad, M.A. Shah, A. Wahid, A. Mehmood, H. Song, H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," *IEEE Access*, vol. 6, pp. 4321-4339, 2018.

[14] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, Z. Yan, "Machine Learning-Based Malicious Application Detection of Android," *IEEE Access*, vol. 5, pp. 25591-25601, 2017.

[15] R. Riaz, S.S. Rizvi, E. Mushtaq, S. Shokat, “OSAP: Online Smartphone’s User Authentication Protocol,” International Journal of Computer Science and Network Security, vol. 17, no. 3, pp. 7-12, 2017.

[16] R. Riaz, T.S. Chung, S.S. Rizvi, N. Yaqub, “BAS: The Bi-phase Authentication Scheme for Wireless Sensor Networks,” International Journal of Security and Communication Networks, 2017.

[17] R. Riaz, S.S. Rizvi, F. Riaz, N. Hameed, S. Shokat, "Analysis of Web based Structural Security Patterns by Employing Ten Security Principles," International Journal of Computer Science and Network Security, vol. 17, no. 10, pp. 45-56, 2017.

[18] I. Burguera, U. Zurutuza, and S.N. Tehrani, “Crowdroid: behavior-based malware detection system for Android,” *in Proc. ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 15-26.

[19] Y. Zhou, Z. Wang, W. Zhou, X. Jiang, "Hey you get off of my market: Detecting malicious apps in official and alternative Android markets," *in Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 5-8.