Abusing Android Permissions: A Security Perspective

Mamdouh Alenezi [1]
[1]Computer Science Department, CCIS
Prince Sultan University, Riyadh, Saudi Arabia
malenezi@psu.edu.sa

Iman Almomani [1 2]
[1] Computer Science Department, CCIS,
Prince Sultan University, Riyadh, Saudi Arabia
[2] Computer Science Department, KASIT
The University of Jordan, Amman, Jordan
imomani@psu.edu.sa , i.momani@ju.edu.jo

***Abstract*—the drastic increase of mobile apps and its direct impact on the security of user's device and data cannot be neglected. Such data nowadays is related to (almost) all life aspects. Even with the growing awareness to develop more secure apps, but still existed mobile apps found on app stores cannot be considered fully benign. This paper is giving a special attention to Android permissions and how they can be abused by security attacks. Most rated education apps have been selected to perform deep permissions analysis and categorization in terms of protection level and mostly abused ones. Moreover, the apps have been examined to check if they support advertisements or not. The results reveal that 80.3% of the apps are requesting permissions more than what they need and actually used. Consequently, such over-privileged apps would be exposed to serious malicious behaviors. The paper discusses possible solutions to overcome this issue and suggests possible ways to select the required permissions throughout the app development process.**

Keywords—Andoird; Mobile Apps; Permissions; Security; Education; protection level.

I. Introduction

Over the last decade, the usage of mobile devices in our daily lives has increased, offering added functionality as personal computers. According to Armando et al. [1] mobile devices are becoming the main stream for accessing several critical infrastructures. The number of apps available and downloaded by smartphone users has increased dramatically [2-3] and surfing app stores has become an entertainment for a lot of people. Mobile apps are small piece of software, which provide functionality to mobile users through accessing sensitive data (e.g., account, password, financial records, medical records, GPS, camera, and microphone) and services located in the cloud (e.g., Google, Facebook, and Twitter). Apps markets such as Apple's App Store and Google's Play provide easy access to hundreds of thousands of apps. Markets streamline software marketing, installation, and update—therein creating low barriers to bringing these apps to market, and even lower barriers for users to obtain and use them. These markets fluidity presents security challenges such as coarse permission systems [4].

Recent reports on smartphones reveal that Android is the most used operating system (OS) [5] as it is an open source and it costs less per installation in comparison to Apple's iOS [6]. Android security permission policy relies on its security permission system as well as on the user's sound judgment. Regrettably, users have usually no security consciousness and they do not usually read the required permissions before installing an application. This is an important area where security teams can educate users about [7].

The permissions requested by the app and the permissions required to access the application's interfaces/data are defined in its manifest file. Permission assignment—and ultimately the security policy for the phone—is delegated to the phone's user. The user sees a screen that lists the permissions an application requests, which they can accept or reject. Making sure that these apps appropriately deal with such great-value sensitive data is an essential and challenging problem [8].

Android platform is very popular because of the available comprehensive framework API. This development API offers mobile apps developers the ability of gaining access to hardware information, knowing phone state, accessing user's data, changing phone settings, etc. The permission model is impacting how developers develop mobile applications. In order to write a mobile app, engineers have to determine for each functionality provided by an API, which permission is needed and make sure it is the correct permission and work correctly. They need a proper mechanism to map between API methods and requisite permissions. Android presently asks developers to say publicly what permissions an app uses, however there is no mechanisms to know the exact purpose of this permission and what kind of sensitive data will be used.

This paper is highlighting the importance of automating the permissions selection process at early stage of mobile apps development. Relaying only on users to decide about the permissions is not an efficient solution. With all current investments to provide secure apps for mobile users, unfortunately existed apps at app stores are still suffering from misused permissions model. To prove that, this study selects most rated education apps as sample and perform critical analysis for their permissions. Such analysis gives clear picture about current status of permissions usage. 71 apps have been selected. The requested and used permissions by each app are counted, compared, and analyzed. Apps' permissions have been categorized in terms of protection level into normal, dangerous, signature, signatureORsystem permissions. Additionally, mostly abused ones are presented. Apps under study are examined to check if they are also utilized to support ads or not. We assume that educational apps should not ask for more permissions and should be trustworthy. Even though, this study found that lots of them abuse the permissions intentionally or accidentally.

The following sections are organized as follows: section II presents set of recent related work. Section III proposes the system model to analyze Android apps permissions. Experimental results and critical analysis is provided in Section IV. The discussions and recommendations could be found at section V. Finally, the paper is concluded and possible future work is presented.

II. Related work

As mobile apps popularity has increased, several research works have been conducted on Android OS security and its permission system. Enck et al. [9] proposed an approach to help identifying Android apps that request a suspicious permission combination using a set of predefined rules. The authors in [10] studied the relationship between the permissions requested by popular and free Android apps and proposed a methodology to improve the expressiveness of app permissions. There are increasing fears about both buggy and malicious apps that may steal, destroy, or leak sensitive data. Modern occurrences of malicious apps found in the Android Market indicate that smartphones are vulnerable to similar malware that have long overwhelmed the PC world [2]. Furthermore, some apps may unintentionally compromise sensitive data.

Relying only on enforcing permissions is not sufficient to prevent security violations, since these permissions can be misused, intentionally or unintentionally, to introduce insecure data flows [11]. Latifa [12] highlighted the issues of security that can occur due to ease in publishing the application on App store and associated security threats due to allocation of permission on mobile phones. The authors' present PermisSecure application to create awareness of permission among users but it was used as analysis to find the permission among paid and free apps and give a statistical analysis of most over permission apps.

Rashidi in [13] presented a framework called DroidNet to install the application without allocating full permissions for a probation mode and later can be converted it into main mode. This application works on expert user reviews and peers to see whether the application is good or not. Expert user seeking is added as Bayesian model. Moreover, users can themselves manage the permission or not. Zhauniarovich in [14] presents the permissions for Android 6.0 where permissions can also be asked at runtime thus awareness in users is required. Moreover, the permissions that had to be asked explicitly from users have been added as default in android 6.0. This paper highlights the key issues and does not talk about awareness of permission of applications. Zhang in [15] highlights the issues that dynamic permission can cause for both android and iOS applications. These permissions can be used by any attacker to embed the code in applications or asking permissions from some stated applications and then exploiting it. FineDroid [16] authors present an application framework to check for inter application and intra application permissions. Felt et.al [17] statically examined requested permissions by using a map that maps permissions to API calls. Nevertheless, it is challenging for static analysis to conclude if a permission is actually used or not. Au et al. [18] showed that users usually disregard the notification and information and accept all requested permissions.

Felt et. al. [19] proposed Stowaway approach that uses static analysis to find the used API calls in order to map them to permissions. Over-permission and under-permission can be found out by making a comparison between the used permissions and claimed permissions in the manifest file. Au et al. [20] proposed PScout that also uses static analysis to the code of Android apps to structure API calls permission mappings that provide more information about these permissions. Pandita et al. [21] and Qu et al. [22] relied on the natural language processing to analyze the app description to automatically evaluate if that app description really requires these permissions.

As can be concluded from the literature, two main approaches exist to address the permission abuse dilemma. The first one is allowing the user to control these permissions upon installation [23]. This approach relies heavily on the awareness of users for the possible damage of these permissions. The second one is classifying apps as malicious or not in the marketplace to alert users while downloading them. This approach does not address already installed apps. Moreover, many static analysis tools have been attempted to study the apps permissions. The following sections present different model to analyze the permissions and introduce other solutions to tackle the permission misusing issue. Additionally, enhance the users' awareness of the importance of approving different permissions. Also, experience another static analysis service.

III. Android Apps Permission Analysis: System Model

The main issue is that most users assume efficient pre-screening process for the apps before making them available at app stores. Moreover, the researchers are usually considering these apps as benign and include them in their datasets whenever they want to compare with malware apps and act accordingly for the purpose of investigation [24-26]. This study emphasizes the importance of studying the uploaded apps at different app stores before classifying them as benign. This paper focus is the permissions usage and its impact on the security of apps and users' data. This study is the starting point to test apps under several categories, observe the permissions requested by each category and their usage, analyze and categorize permissions to define the most abused ones by security attackers. Also, draw conclusions and put recommendations to develop more secure apps and build trusted, benign apps databases. This paper aims mainly to enhance the security awareness of the users and put recommendations to help mobile apps developers take proper decisions regarding the permissions to be approved during the app development process.

Education apps category are studied in this research. Most 71 rated education apps have been collected and their apks were downloaded through [27]. After that, all these apps have been tested using AVC UnDroid which is a free online service of AV-Comparatives [28]. AVC UnDroid provides a static analysis of Android apps. The reports generated by this tool have been analyzed carefully especially in terms of permissions usage. Fig. 1 shows the system model of the proposed approach to analyze Android Apps' permissions.

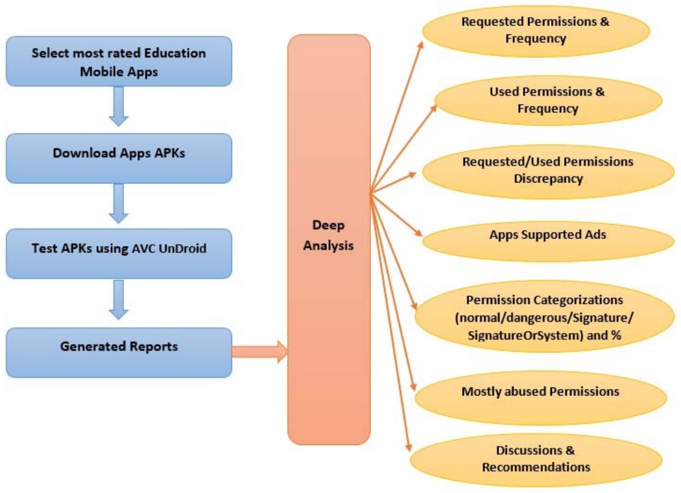
Select most rated Education Mobile Apps
Download Apps APKs
Test APKs using AVC UnDroid
Generated Reports
Deep Analysis
Requested Permissions & Frequency
Used Permissions & Frequency
Requested/Used Permissions Discrepancy
Apps Supported Ads
Permission Categorizations (normal/dangerous/Signature/SignatureOrSystem) and %
Mostly abused Permissions
Discussions & Recommendations

Fig. 1. Android Permission Analysis: System Model

IV. Experimental Results and Analysis

The permissions of each one of the 71 apps have been counted, studied, and analyzed. Fig. 2 shows the frequency of System permissions requested by these apps. As can be seen, the mostly requested permissions are:

- `android.permission.INTERNET`,
- `android.permission.ACCESS_NETWORK_STATE`,
- `android.permission.WAKE_LOCK`,
- `android.permission.WRITE_EXTERNAL_STORAGE`

with frequencies 70, 65, 45, 40; respectively. The reset as are shown in Fig 2.

Whereas, the frequency of the used permissions is illustrated in Fig. 3 which shows significant gap between what is requested and what is actually used in terms of system permissions. Detailed statistical analysis in terms of exact numbers of permissions requested and used by the 71 apps in addition to the difference between them are given in Table I. This includes not only the system permissions (listed in Fig 2), but also Custom permissions which reaches 25% of the total requested permissions. Custom permissions allow app developers to define their own permissions.

As can be observed from Table I, non-trivial discrepancy from what apps are requesting and what they are actually using. General observations and comparisons are summarized in Table II. 80.3% of the apps have requested permissions more than what they have used. Maximum difference in this case is 21; whereas the minimum is 1. On the other hand, 11.2% of apps have requested less than what they have used with maximum difference 15 permissions. Only, 8.5% of the apps have used what they have requested.

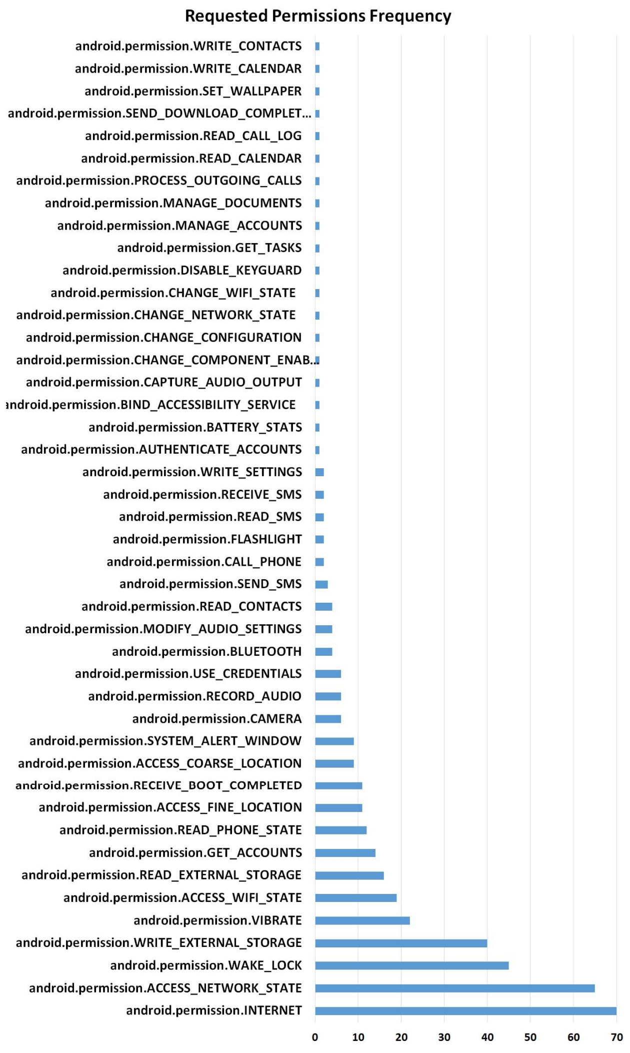

Fig. 2. List of requested permissions (System ones) and their frequency

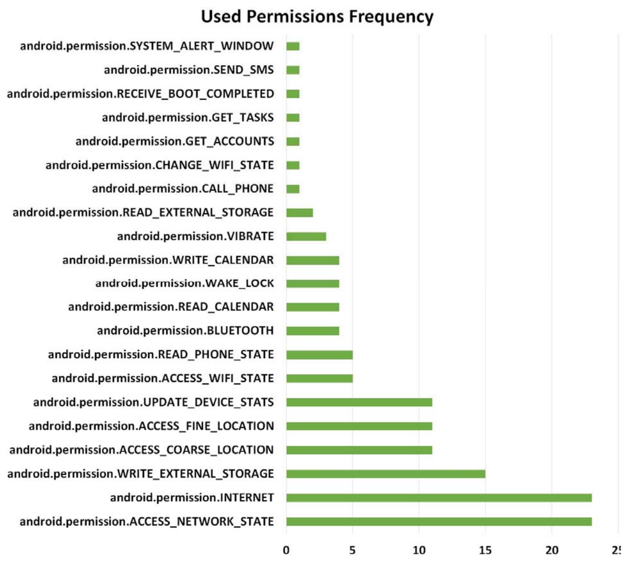

Fig. 3. List of used permissions (System ones) and their frequency

TABLE I. NUMBER OF PERMISSIONS REQUESTED AND USED BY EACH APP AND THE DIFFERENCE BETWEEN THEM

App. No.	Req. Perm.	Used Perm.	Diff. (R-U)
1	6	2	4
2	6	0	6
3	3	6	-3
4	12	0	12
5	20	0	20
6	1	0	1
7	10	0	10
8	8	0	8
9	10	0	10
10	10	0	10
11	2	0	2
12	8	0	8
13	9	0	9
14	15	0	15
15	13	11	2
16	3	0	3
17	8	0	8
18	9	0	9
19	7	0	7
20	6	21	-15
21	6	0	6
22	7	0	7
23	7	0	7
24	2	2	0
25	2	6	-4
26	12	0	12
27	1	2	-1
28	12	6	6
29	3	3	0
30	4	0	4
31	6	0	6
32	8	0	8
33	5	0	5
34	3	3	0
35	23	11	12
36	6	2	4
37	2	5	-3
38	2	2	0
39	4	6	-2
40	15	0	15
41	19	0	19
42	8	0	8
43	5	0	5
44	3	3	0
45	1	0	1
46	20	7	13
47	13	0	13
48	10	0	10
49	4	6	-2
50	13	11	2
51	5	3	2
52	2	2	0
53	2	6	-4
54	8	0	8
55	3	0	3
56	8	0	8
57	7	0	7
58	3	0	3
59	11	0	11
60	4	0	4
61	7	0	7
62	6	0	6
63	3	0	3
64	8	0	8
65	13	0	13
66	5	0	5
67	2	0	2
68	21	0	21
69	7	4	3
70	14	0	14
71	5	2	3

TABLE II. REQUESTED/USED PERMSSIONS ANALYSIS

	No. Of Apps	**Min**	**Max**	**%**
Requested Perm > Used Perm	57	1	21	80.3%
Requested Perm < Used Perm	8	1	15	11.2%
Requested Perm = Used Perm	6	0	0	8.5%

Moreover, this study has investigated to what extent education apps can be utilized for advertising purposes. Table III shows the number of apps support advertisement and their percentages.

TABLE III. PERCENTAGE OF APPS THAT SUPPORT ADVERTISEMENTS

Ad supported		**%**
Yes	19	26.76
No	52	73.24

The potential risk of Android permissions and the procedures need to be followed in order to grant the requested permission or not is characterized through a protection level [29]. Table IV classifies the permissions of the studied education apps in terms of protection level. As can be seen, the majority of permissions with percentage 63.77% are normal permissions. Dangerous permissions come next with 27.17%. Both Signature and SignatureOrSystem form 9.05% of the total permissions. This does not mean that security attackers cannot exploit normal permissions. Table V shows that even normal permissions (such as `android.permission.INTERNET`) are listed under the most abused permissions by malicious apps. This stresses the point of requesting only the needed permissions even if they are classified by Android developers as normal permissions.

TABLE IV. PERMISSIONS CATEGORIZATION AND PERCENTAGE

Protection Level	**Meaning**	**No. of Permissions**	**%**
Normal	Low risk permissions. Automatically granted without the user's approval at installation time.	338	63.77%
Dangerous	High risk permission that gives access to private user data or control over the device. May not granted automatically and needs user's approval.	144	27.17%
Signature	Permission is granted automatically without explicit approval from the user in case of certificate matching.	42	7.92%
SignatureOrSystem	Permission is granted only if the app is in the system image or in case of certificate matching.	6	1.13%

TrendMicro [29] has listed twelve most abused Android apps permissions. The majority of these permissions are also found in Education apps under study. Table V. shows the permissions found and how they can be exploited.

TABLE V. MOSTLY ABUSED ANDROID PERMISSIONS FOUND IN THE EDUCATION APPS.

Permission Name	**What it is Used For**	**How it can be Exploited**
Network Based Location	Approximates location of user	Location based attacks or malwares
GPS Location	Gives location through GPS	Location based attacks or malwares
View Network state	Checks cellular network connection	Download routine or malware
View Wi-Fi State	Gives Access to Wi-Fi network information	Steal Wi-Fi passwords
Retrieve Running Apps	Finding which apps or process are running	Kill running applications or get information about running apps
Full Internet Access	Allows Internet connection	Internet access could be used for exploitation or malwares
Read phone state and identity	Allows access to all information about calls, network, IMEI and other identifying information	Steal Information from data
Automatically start at boot	Allows apps to start at boot time	Run malicious apps at boot time
Control Vibrator	Allows access to device vibrator function	Stop vibration to prevent notification by Malicious app
Modify/ Delete SD card contents	Writes on external card	Use the storage temporally to store stolen information in order to distribute them later to command center
Sends SMS message	Sends text message	Send messages to premium numbers which causes money loss and also contacting command center

Interesting observations show that many of the requested permissions which are included in Table V are not used. This will expose the app to different security threats for no reason. For example, `android.permission.INTERNET` which is the mostly requested permission, only 33% of the apps have practically used it. Similar case was found in `android.permission.ACCESS_NETWORK_STATE`. Moreover, the percentage of usage was even worse in many other permissions which did not exceed 7% in some of them such as `android.permission.GET_ACCOUNTS`.

V. DISCUSSIONS AND RECOMMENDATIONS

It is observed from the studied applications that lots of apps request permissions even though they are not using them. Khatoon and Corcoran found after their analysis of android apps that this apparent need for a built-in mechanism that decides if a particular permission is necessary for that app or not [5]. This is a clear indication that developers and testers are not aware of this issue. Thus, a need rises to design a module as an add-on for any development framework which will assess these permission requests. It is necessary to have such module to help in the development of mobile apps. The add-on can add the request for permission but in case of non-necessary permission or security critical permission requires developers to look again to their code and make sure it is necessarily needed permission. Moreover, developers should be aware that Android systems could add permissions automatically to the project manifest due to platform change [30].

The need arises for an integrated solution that keeps track of permissions and their usage. The solution should make sure that permissions are being used and there are reasonably justified. Visualizing permissions will greatly help software engineers understand and deal with permissions. Intelligence component can be added to the solution to be the brain of the solution and learn from experience about the usage of permissions during mobile apps development. The integrated development environment (IDE) should have a mechanism to check if all requested permission are being actually used while the app is under development. A visualization will also aid developers to monitor these permissions and their behavior. This intelligent add-on will solve the issues that arise due to lack of developer experience or lousy coding practices that can result in security breaches.

CONCLUSIONS AND FUTURE WORK

The paper uncovers the critical status of permissions usage of the available apps offered by different app stores. The proposed system in this paper has been used to deeply analyze the most rated apps in the education category. The analysis results reveal that most of the apps are requesting permissions which are not required but expose the apps and their users to high possibility of being easily attacked. The main aim of this study is to enhance the security awareness of the users and put recommendations to help mobile apps developers to take proper decisions regarding the required apps permissions during the development stages. Also, this study is considered as a starting point to build a truly benign datasets of mobile apps that could be used for research purposes. The researchers will be able to reproduce the study and explore more possibilities in studying apps permissions

As future work, other apps categories will be addressed and analyzed. Also, metrics other than permissions usage could be considered to examine the apps from security perspectives.

REFERENCES

[1] Armando, A., Bocci, G., Chiarelli, G., Costa, G., De Maglie, G., Mammoliti, R., & Merlo, A, "Mobile App security analysis with the MAVeriC static analysis module", J. Wirel. Mob. Netw., Ubiquitous Comput., Dependable Appl.(JoWUA), 5(4), 2014, pp. 103-119.

[2] Gilbert, P., Chun, B. G., Cox, L. P., & Jung, J., "Vision: automated security validation of mobile apps at app markets", In Proceedings of the second international workshop on Mobile cloud computing and services, 2011, pp. 21-26.

[3] The Statistics Portal, " Number of available applications in the Google Play Store from December 2009 to June 2017", https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store /, last access 8th of Augest 2017.

[4] Enck, W., Octeau, D., McDaniel, P., & Chaudhuri, S., " A Study of Android Application Security", In USENIX security symposium (Vol. 2, p. 2), 2011.

[5] Abhijit Ahaskar, Android vs iOS: What makes the former more popular, http://www.livemint.com/Leisure/LtAQtZUiHlkZ6Rms9MvNUI/Android-vs-iOS-What-makes-the-former-more-popular.html, last Access 8th of August, 2017.

[6] Statista.com, Average cost per installation (CPI) for Android and iOS apps worldwide as of February 2017 (in U.S. dollars), https://www.statista.com/statistics/247411/cost-per-installation-android-ios-worldwide/ , last Access 8th of August, 2017.

[7] de la Puerta, J. G., Sanz, B., Grueiro, I. S., & Bringas, P. G., "The Evolution of Permission as Feature for Android Malware Detection", In International Joint Conference Springer International Publishing, 2015, pp. 389-400.

[8] A. Khatoon and P. Corcoran, "Privacy concerns on Android devices", IEEE International Conference in Consumer Electronics (ICCE) , 2017 pp. 149-152.

[9] Enck, W., Ongtang, M., & McDaniel, P. , "On lightweight mobile phone application certification", In Proceedings of the 16th ACM conference on Computer and communications security, 2009, pp. 235-245.

[10] Barrera, D., Kayacik, H. G., van Oorschot, P. C., & Somayaji, A., "A methodology for empirical analysis of permission-based security models and its application to android". In Proceedings of the 17th ACM conference on Computer and communications security, 2010, pp. 73-8..

[11] Fuchs, A. P., Chaudhuri, A., & Foster, J. S. (2009). Scandroid: Automated security certification of android.

[12] Latifa, ER-RAJY, and EL KIRAM My Ahmed. "A New Protection for Android Applications." International Journal of Interactive Multimedia and Artificial Inteligence 3. Regular Issue (2016).

[13] Rashidi, Bahman, et al. "Android Permission Recommendation Using Transitive Bayesian Inference Model." European Symposium on Research in Computer Security. Springer International Publishing, 2016.

[14] Zhauniarovich, Yury, and Olga Gadyatskaya. "Small Changes, Big Changes: An Updated View on the Android Permission System." International Symposium on Research in Attacks, Intrusions, and Defenses. Springer International Publishing, 2016.

[15] Zhang, Yuan, et al. "Rethinking Permission Enforcement Mechanism on Mobile Systems." IEEE Transactions on Information Forensics and Security11.10 (2016): 2227-2240.

[16] Yuan Zhang, Min Yang, Guofei Gu, and Hao Chen,"FineDroid: Enforcing Permissions with System-wide Application Execution Context", International Conference on Security and Privacy in Communication Systems, 2015, pp.3-22.

[17] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D., " Android permissions demystified", In Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 627-638.

[18] Au, K. W. Y., Zhou, Y. F., Huang, Z., Gill, P., & Lie, D. , Short paper: "a look at smartphone permission models", In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, 2011, pp. 63-68.

[19] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D., " Android permissions: User attention, comprehension, and behavior", In Proceedings of the Eighth Symposium on Usable Privacy and Security,2012, (p. 3)..

[20] Au, K. W. Y., Zhou, Y. F., Huang, Z., & Lie, D. , "Pscout: analyzing the android permission specification", In Proceedings of the 2012 ACM conference on Computer and communications security 2012, pp. 217-228.

[21] Pandita, R., Xiao, X., Yang, W., Enck, W., & Xie, T. , " Whyper: Towards automating risk assessment of mobile applications", In Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13) , 2013, pp. 527-542.

[22] Qu, Zhengyang, et al. "Autocog: Measuring the description-to-permission fidelity in android applications." Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014.

[23] Zhou, Y., Zhang, X., Jiang, X., & Freeh, V. W. , "Taming information-stealing smartphone applications (on android)", In International conference on Trust and trustworthy computing, 2011, (pp. 93-107). Springer Berlin Heidelberg.

[24] Bhaskar Sarma, Ninghui Li, Chris Gates, Rahul Potharaju, Cristina Nita-Rotaru, "Android Permissions: A Perspective Combining Risks and Benefits", Proceedings of the 17th ACM symposium on Access Control Models and Technologies, 2012, pp. 13-22.

[25] Jis Joe Mathew, Manoj T. Joy, "Efficient Risk Analysis for Android Applications", Recent Advances in Intelligent Computational Systems (RAICS), 2015, pp.382-387.

[26] Chenkai Guo, Jing Xu, Lei Liu and Sihan Xu, "Using Association Statistics to Rank Risk of Android Application", International Conference on Computer and Communications (ICCC), 2015, pp. 1-5.

[27] Apkpure.com, **https://apkpure.com/,** last access 9th of July 2017.

[28] AVC UnDroid, http://undroid.av-comparatives.info/index.php, last access 9th of July 2017.

[29] TrendMicro, http://about-threats.trendmicro.com/us/library/image-gallery/12-most-abused-android-app-permissions, last access 9th July 2017.

[30] Jessica Thornsby, Android UI Design, Packt Publishing Ltd, 1st Edition, 2016.