DOI 10.12694/scpe.v18i3.1304
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 243–252. http://www.scpe.org

PRIVACY ANALYSIS OF ANDROID APPLICATIONS: STATE-OF-ART AND LITERARY ASSESSMENT

GULSHAN SHRIVASTAVA AND PRABHAT KUMAR *

Abstract. In todays world, Android has revolutionized every facet of our lives. Almost all the important services such as banking, transportation, stock trade, medicine, education, etc. are extended to Android these days. Everything is available in the application market of android. Unfortunately, at the same time, the prosperity of these applications also attracts abusers and malicious attackers to perform different types of attacks. An appropriate action needs to be taken to protect Android device from these attacks. Android applications privacy analysis is an extension to the android privacy model, which traditionally emphasizes on prevention, and detection of attacks. It also deals with capturing, recording and analysis of Android applications to detect and investigate Android device intrusions. In this paper, we explore the comprehensive study about different techniques proposed to perform Android applications analytics. In addition to this, various aspects of android applications analytics have been reviewed along with related technologies and their limitations. This gives enhanced recognition of the problem, existing explanation space, and potential research scope to analyze and investigate various Android device intrusions against such attacks efficiently.

Key words: Android Analytics, Smartphone intrusion, Android attack

AMS subject classifications. 68M12, 68M14

1. Introduction.

When a user download application in an Android device from Android market (Google Play, 1Mobile, Todd etc.) and try to install (use application in advanced version of Android) on device, that application require some permissions from user to use Android device functionality. It is the wish of user to grant those permissions or not. Recent studies have shown that majority of users does not bother about this permission due to lack of knowledge about it and use it directly; at that stage user is not aware about the interruption. Even if any user wants to read these permissions, but these permission descriptions are too confusing and tough to understand [1, 2].

Privacy is a serious concern because these applications often ask for the access to confidential and sensitive information on Android device to work properly. Twitter, for example, asks to record audio with the microphone (at all time without user confirmation), read Google service configuration, read phone status and identity, etc. A recent research explored, 96% of applications need EMAIL, 92% needs ADDRESS_BOOK, 84% needs LOCATION, 52% need CAMERA, and 32% need CALENDAR permissions [4]. Many social networking websites like Facebook, Twitter and Linkedin have lots of user and they use advertisements for instance sponsored ads and posts, to generate good revenue. To get the significance and accomplishment of their advertisements i.e., targets advertisements, such website collect users private information through their applications demanding unnecessary permissions. As a conservatory of current research on permissions, we examine end users experience regarding the level of permissions being requested by various Android applications.

1.1. Android as a Major Application Provider.

In between August 2010 to May 2016, Android had over 65 billion application downloads. Also Apple and Google store combined revenue is $2.23 billion in the US [26]. Android application is not pre-screened for quality even in its official market. AppBrain gives a report that 33% applications are rated low quality based on its rating and recommendation. In 2011, Juniper Networks reported that 47.2% Android malware samples increased in between July and November 2011 [27]. Some reputed organization like McAfee [28] and Kaspersky Lab [29] also reported for continued exploits. The intention of these malware is to collect the users personal data and share or sell them to targeted advertisement companies.

1.2. Motive of Amendment in Android Application Display.

Permissions are important part of an application but they are only displayed at the time of installation. Therefore, Android has done some amendment in their policy to display application permission. As per Android 6.0, whenever an application

*Department of Computer Science and Engineering, National Institute of Technology Patna, INDIA({gulshanstv@gmail.com, prabhat@nitp.ac.in}).

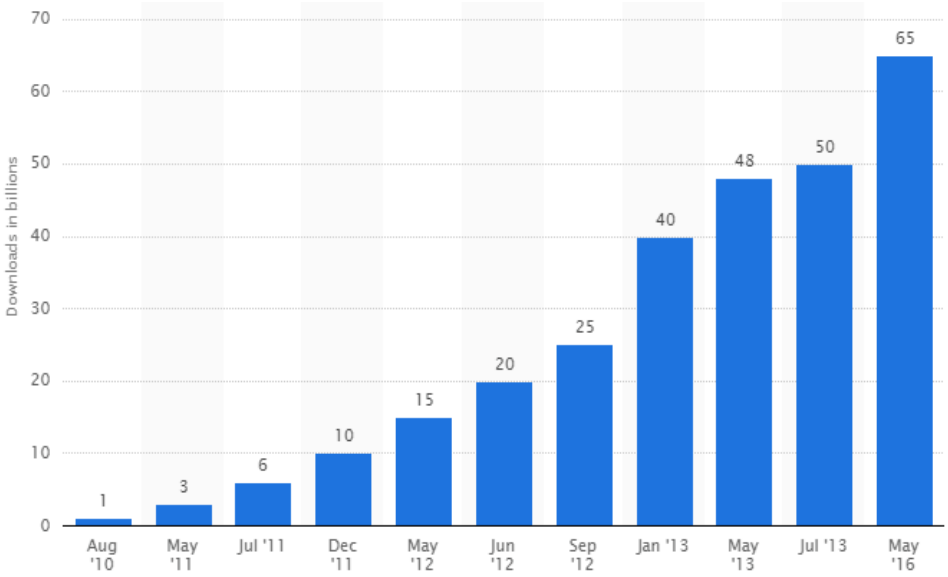

FIG. 1.1. *Google Play: number of Android app downloads 2010-2016 [26]*

requires any permission then it will only ask for that particular permission, instead of these lifetime grants of permissions.

1.3. Privacy on Android Permission. Android application permissions asked by an application are listed in AndroidManifest.xml. As per many research survey [35, 36] it has been found that very less number of users understand that what these permissions actually asking for. These surveys are based on large risk assessment of users about security and privacy risks. Felt et al. [37] also did survey about the way of asking this permission and presented some set of guidelines for user to take into account while making decision about their privacy and security concerns. King et al. [38] presented the users expectation from their Smartphone for entire usage. In this work, they collected interview results of iPhone and Android users and highlighted the problem faced by them at the time when application and web links, gathers the information of possible privacy faults, and process to select the best application from app markets.

The paper is organized as follows: Section II presents Literature review. In Section III, we explore the Android applications permission and Android application mechanism in detail. Section IV discusses about the limitation of the Android permission. In Section V, we describe the permission analytics. Finally, Section VI concludes the paper.

The paper is organized as follows: Section II presents Literature review. In Section III, we explore the Android applications permission and Android application mechanism in detail. Section IV discusses about the limitation of the Android permission. In Section V, we describe the permission analytics. Finally, Section VI concludes the paper.

2. Literature Review. As Android device usage is increasing day by day, at the same time the risk factors of security breach are also increasing. The biggest challenge is to identify and prevent those risks. Malicious Android applications analysis and detection is admired research fields as per escalation in usage of Android market. While Android has survived openly since 2008, a major amount of effort has been conduct on studying the Android permissions/security model. A lot of focus have been done on creating hypothetical formalizations of Android security mechanism or presenting enhancements to system security, and is mainly out of scope.

In Android 4.3, hidden features like App Ops allow the user for revoke the irrelevant permissions as per application but as per the unawareness user do not explore it. According to Verizons 2015 data, in middle attack or applications that store sensitive information insecurely or send data incomplete. According to the Jupiter network report [11, 12], 76% of mobile users depend on their mobile devices to access their most sensitive

personal information, such as online banking or personal medical information. This trend is even more noticeable with those who also use their personal mobile devices for business purposes. Nearly nine out of ten business users report that they use their own mobile devices to access sensitive work-related information. It has been reported by the Junipers Network Mobile Threat Centre that in 2011, there was an unparalleled increase in Android malware attacks with an increase of about 155% from previous years in all platforms.

According to Googles CEO, 1.5 billion known users use Android. Recent version used is Android 6.0 Marshmallow. Around 8 out of 10 are Android users. In addition, 1.6 million devices access Google play. The SAN report analysis outsourcing growth to 1.4 million in 2016. Even in the latest edition of Android operating system, there is no provision to handle and manage security and privacy issues. Google Play provides security to play store and manage its privacy concern security.

Researchers have to be very conscious about permission usage in Android. Kelley et al. [2] explained about unawareness of permissions. Felt et al. [9] explored that Android failed to clearly inform their users about permissions. Kelley et al. [6] proposed a privacy checklist for the privacy risk of an application, they also showed that proposed checklist significantly affected user for application selection. Felt et al. [1] explored about gap analysis between free and paid applications in the frequency of dangerous permission being requested after surveying 100 paid & 856 free applications; for instance, ratio of free and paid is 7:2 in the aspect of INTERNET permission. Hypothesis of [7] presents the frequency of request from free applications for INTRNET permission for advertisements. Felt et al. [8] fixed some permission that are common between normal, dangerous and signature permissions by Google but still behave like dangerous permission legitimately. Felt et al. [9] concluded that 1/3 of 940 applications are measured as over privileged. Vidas et al. [10] proposed a static analysis tool to enhance the set of permissions and minimize it so that application behaves correctly. This literature mainly focuses on malicious analysis and detection based on Android applications permission. Enck et al. [30], TaintDroid has filled the distance between system security and user permissions, focusing on analysis of application, which request for permissions through permission list. Vidas et al. [31] gives details about application permissions requests, finding ubiquitous permissions creep, because of existing developer APIs that composes it making it complicated for developers to align their permission requirements with application functionality. Felt et al. [32] endeavor to make clear permissions to developers. However, neither of these articles investigates end users considerate for permissions.

Researchers have exposed, attack vectors for applications to compose permission requirements that are not detailed to users. Others who have come across at Android permissions have attempted to collect applications that necessitate similar permissions to simplify the existing method or have attempted an evaluation of the dissimilarity between current Android permission systems [34]. Haris et al. [13] concentrated on the risk in mobile computing and privacy leaks in mobile connectivity and mobile sensing. State-of-art is similar to [3, 5] for multiple mobile platforms. Crussell et al. [14] proposed DNADroid that approach used program dependency graphs i.e. detects the similar applications. This approach generates a low false positive rate. For the increase in the system performance, they evaluate application against similar names. Zheng et al. [15] proposed a static analysis framework i.e. DroidAnalytics. It detected obfuscated Android malware. Signature generation performed in three level methods. Repackaged malware detected by the class and application level in that sequence. Then it is compared to other effective obfuscated technique such as method renaming, further control flow goes to-obfuscation and string encryption. In this article, authors didnt discuss about the time taken to analyze the applications. Hanna et al. [17] proposed the Juxtapp, they introduced the Feature Hashing for the detection of similar applications. It is resilient to some amount of obfuscation. They used obfuscation resilience of algorithms to detect the repackaging apps.

Zhauniarovich et al. [33] proposed detection of repackaged method that showed the results of similar techniques involved in code analysis. This approach is based on the content of. apk file. Repackaged applications have minor changes in lots of file, which is difficult to identify. Zhou et al. [16] proposed DroidMOSS method to repackaging detection through Fuzzy Hashing. In this paper, authors generate a hash to perform divide and conquer method for that application. This proposed system is robust as claimed by author but it is based on number of chunks, which is subpart of an application. In another research, Zhou et al. [18] proposed AppINK based on client side for repackage detection. In this method, watermarking technique is used for authentication of an application. Based on watermarking, user can identify the repackaged application. Authors validate their

Android Application	Category	Permission Requested	Flow Permission
My Calendar	Productivity	STORAGE	PHONE NO → NETWORK
		LOCATION	
		NETWORK	
		PHONECALLS	
My Space	Social	STORAGE	IMEI N →NETWEOK
		NETWORK	
		PHONECALLS	
Gmail	Communication	NETWORK	STORAGE→NETWORK
		STORAGE	

FIG. 3.1. *Android Applications and their Permissions Examples*

proposed method with the help of some pattern matching algorithm. Wu et al. [19] provided the novel policy that deals with repackaged issues. This is dynamic event flow and verifies the applications in the future, by the help of watermark injected in the applications. The weakness of this techniques is it takes more overhead and inherently limited resources on Android device when identify the watermark from applications. It is not sufficient automatic watermark extraction technique.

Kang et al. [20] designed fast malware detector by using creator information. In this detection is done according to malware permission and behavior. This is based on the static analysis that is used to certify serial number and detects the malware. It is light weighted process, which analyze particular part of applications according to malware behavior and permission. This provides the very high detection accuracy.

Enck et al. [21] explained about Android security mechanism based on permissions and security issues for inter-communication between apps. Inter-Component Communication (ICC) and Intents have not been explored the way permissions have been investigated. Most of the existing ICC based studies focuses on finding the ICC related vulnerabilities. The work depicted in [22] investigated the IPC framework and interaction of system components. ComDroid [23] detects the ICC related vulnerabilities. The work depicted in [24] suggested improvement in ComDroid by segregating the communication messages into inter and intra-application groups so that the risk of inter-application attacks may be reduced. The work depicted in [25] performs information flow analysis to investigate the communication exploits.

3. Android Application Permission. Permission system is a very basic security aspect implemented in todays Android devices. The Android Market [43] has experienced a great boon as it allows anyone to upload applications to the market. By which, a developer can be rewarded with a monetary gain by receiving ad revenue or by charging for the services provided by the application. Permissions are provided for security mechanism to defend the receptive APIs thus to be utilized only by trusted applications. Before installing an application, a list of permissions is requested to the user. The permissions are granted on all or nothing basis i.e. a user is not allowed to selectively allow or disallow permissions that an application is requests.

There are mainly four types of Android permissions:

1. Signature
2. System
3. Normal
4. Dangerous

Signature and System are reserved for the application that comes preinstalled on the firmware or have a key signed in firmware. These types of permissions are not available to third party applications. When an application is reinstalled, it should be signed by the matching certificate as the previous application in order to obtain the old applications data, and then only Signature permission will be granted by the system without user interaction.

Normal and Dangerous are other most important type of permissions. Normal permissions are low level of permission which do not required any user grant. User can examine this permission and if found any doubt then no other option rather than un-installation of that particular application is available. Whereas dangerous

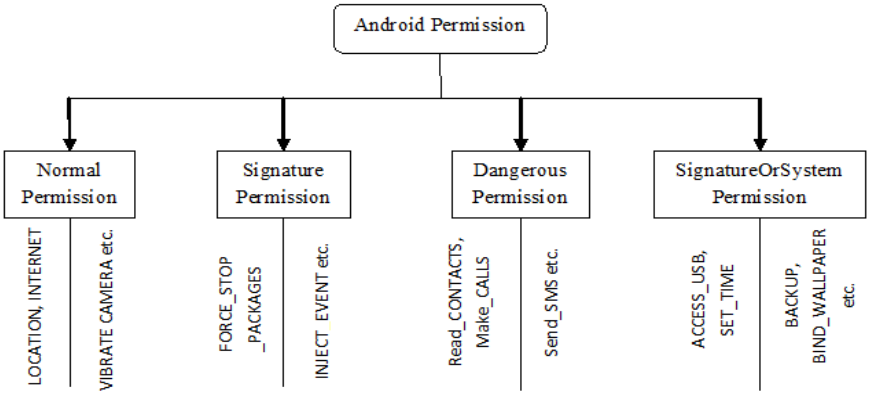

FIG. 3.2. *Categorization of Permission*

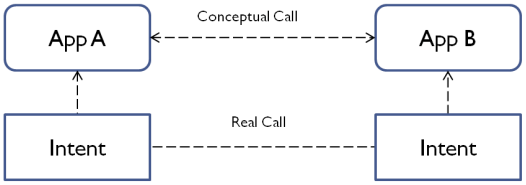

FIG. 3.3. *Inter Process Communication*

permission, are asked by user and it is the user who wish to grant them or not. If a user grants any such type of permission, then it is permanently granted. These types of permission are mandatory to display from application developer at the time of installation of that application.

3.1. Inter-process Communication. Two ways process communication can be divided into two categories firstly by using any of the conventional UNIX-type mechanisms for instance, the file system, local sockets, or signals etc. Whenever an application directly communicates with another application without user interaction it shows, its conceptual call but that time it uses intent and intent filter in real call as fig.4. Whenever a component is requested by an application, the Android system checks if the corresponding process is running and the component is instantiated or not. If either of them is not available, it is created by the system.

Android also provides IPC mechanisms:

1. **Binder:** It is a frivolous remote process call system based on compatibility. It is intended for elevated performance while executing process calls.
2. **Intents:** It represents an "intention" to obtain something, which is finished. For instance, if application needs to demonstrate a web page, its intent to view the URL is expressed by creating and sending an Intent instance to the system. The system finds corresponding code (for this example Browser) which knows to offer service to that Intent, and executes it as shown in figure 5. Intents are also used to broadcast system-wide events. Intents are capable of crossing process boundaries and can transfer information between applications.

   Types of Intent:

   (a) **Explicit Intent**

```
String url = www.google.com;
Intent explicit = new Intent(Intent.ACTION_VIEW);
explicit.setData(Uri.parse(url));
explicit.setPackage(com.Android.chrome);
startActivity(explicit);
```

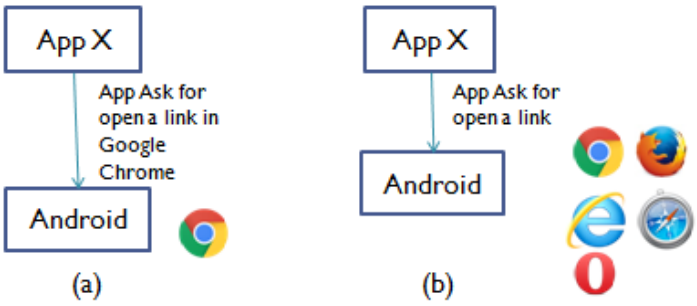

FIG. 3.4. *(a) Explicit Intent; (b) Implicit Intent*

(b) **Implicit Intent**

```
String url = www.google.com;
Intent implicit = new Intent(Intent.ACTION_VIEW);
implicit.setData(Uri.parse(url));
startActivity(implicit);
```

3. **Content Providers:** ContentProvider serves as a data storage, which provides access to data stored on the device; for example, user's list of contact scan can be thought of as ContentProvider. An application can access data provided by other application via ContentProvider, and an application creates a ContentProviders to share its data.

3.2. Android Permission Mechanism. The middleware layer of Android architecture provides permission mechanism. To implement mandatory access control (MAC) on inter-communication calls, reference monitor is used. Standard Android permissions such as INTERNET, PHONE_CALLS, and SEND_SMS etc. are used to protect security sensitive interfaces. Application holds these permissions to be capable of making phone calls, to access the Internet or to send text SMS. Apart from that, application can declare different types of permission labels to bar access to its interfaces. Necessitated permissions are displayed in a file and approved during installation based on user confirmation. When an application requests inter-communication calls, reference monitor checks whether the application has required permissions or not.

4. Limitations. The limitations that lead to weaknesses, of Android security model are summarized as follows:

1. Limitation 1: Android uses an install-time permission model:
   (a) The install-time permission works on an all-or-nothing principle: in older versions, a user has to grant all permissions requested by an application, in order to proceed for installation.
   (b) Use is mostly unaware of consequences of requested permissions, which leads to approval of permissions [8].
   (c) Application developers try to expand their requested permissions, many of which are not necessary for applications execution. [40]
   (d) Lack of sufficiently fine-grained permissions. [41]
   (e) There is no provision of runtime permission revocation, and control [42].

   Once the application is installed on a device, the application is authorized to use the device, as it wants to use all its granted permissions. There are no controls to limit the uses of permissions.

2. Limitation 2: In comparison to desktop environment, such as Windows and UNIX executables, Android applications are easier to reverse engineer. This is possible due to presence of hardware-independent Dalvik bytecode files. These files contain great amount of data of original Java sources.

3. Limitation 3: Lack of isolation mechanism for third-party libraries contained within applications [43].

Due to this, third-party libraries cannot be stopped from misusing the granted permissions of host applications. Conversely, ill-behaving host applications may tamper with the libraries, for example, by performing improper advertisement display or click fraud.

4. Limitation 4: Android lacks a configurable, runtime ICC control for the following purposes:
   (a) To prevent an application from accessing any open interfaces of another application, despite the former having obtained the required permissions at its install time [23, 9]. Combined with weakness 5 below, this lack of runtime inter-application access control can lead to data leakage and confused deputy problems.
   (b) To prevent an application from intercepting an intent broadcast and possibly stopping its propagation afterwards [23]. By intercepting system-event broadcasts, a malicious application is able to stealthily intercept important system events that contain sensitive information, such as an incoming call or SMS.
   (c) To isolate applications and prevent them from communicating via ICC and other shared channels [44]. The presently unrestricted ICC among applications in Android can be exploited by colluding applications.

5. Limitation 5: Application developers could be unfamiliar with the subtle aspects of Android ICC [45], which may lead to unintentional exposure of applications private interfaces and data.

5. Permission Analytics. An accurate estimate of success matrix is one of the most basic requirements for Android based organization and researchers planning for analytics to get ultimate goal of identified risk and satiate this need of Android users respectively [39]. Analytics have moved on from data analytics to all other digital media and hence renamed accordingly. For example, Web Analytics for data on web, Mobile Analytics for mobile applications, Multi-Channel Analytics for collecting analytical data across heterogeneous platforms, Social Media Analytics for banking, Facebook Insights, Twitter Analytics and YouTube Analytics.

To maintain long term sustainable relationship with customers, rather than being content with short-term boost in the conversion rate by giving priority to customer over an experts or analysts view helps in decreasing the bounce rate and optimize global maxima rather than local maxima for every search, by focusing on key performance parameters (KPIs) for success and failure. These new age analytics that intend data from each achievable medium including web and social site is helping business syndicates to access even the most personal information about their customers and prospect research aims in managing of disclosure of personal information and giving only those aspects which could amplify their breach investigation, reports that 0.03% of Android device per week infected by the malicious code out of 10 million of mobile device. The Applications vulnerability is consisted of risky applications prone in man customer base. Permission analytics are categorized based on their main approaches as follows.

1. **Static Analysis**: A static analyzer performs inspection of application without its execution; analyzer is able to achieve high code coverage. Since it have been analyze the applications complete source or recovered code. However, it does not say anything about execution context as well as its execution. Apart from that, it faces difficulties in the presence of dynamic code loading and code obfuscation [46]. Analyzer is required to perform accurate static analysis on an event-driven Android application to deal with the various Android-specific issues such as: [47]:
   (a) Presence of multiple entry points in application;
   (b) App components may run asynchronously
   (c) Possibility of multiple callbacks because of applications lifecycle states, as well as listeners to system and GUI events;
   (d) Application might accept both intra and inter application ICC. In addition, Java reflection and native code possess additional challenges [48].

2. **Dynamic Analysis:** It is performed by executing the application. This can be done using either real or virtual environment for instance, Android Virtual Device (AVD) and monitoring the application while

it is executing. Huge variety of dynamic analysis techniques are given in the literature. Performing dynamic analysis on Android applications faces new difficulties such as Binder-based ICC, Androids managed resources and event triggers [49].

3. **Static & Dynamic Analysis:** Due to limitations of static and dynamic analysis techniques individually, many systems choose to perform both: static and dynamic analysis to complement each other. Google Play uses Bouncer, which scans for malicious applications automatically. Many security researchers have made attempts to figure out working of bouncer, as it is not public [50, 51]. According to Oberheide et al. [54] seemed to perform dynamic analysis on an application by running it on a QEMU-based emulated Android environment for only 5 minutes. Mobile-Sandbox [59] also employs both static and dynamic analysis. Application requested permissions are analyzed for static analysis. Then, the applications Dalvik bytecode is converted to small code by using baksmali5. During dynamic analysis stage; to improve code coverage information on timers and broadcasts are also collected, Mobile-Sandbox stores runtime information at the three levels:
   (a) Dalvik level monitoring using TaintDroid and a customized version of DroidBox;
   (b) Native code monitoring using ltrace; and
   (c) Network traffic monitoring into a PCAP file.

   Mobile-Sandbox utilizes the monkey runner *tool6To* simulate user interaction, to trigger potentially malicious behavior external events, such as incoming calls or SMS messages, are also simulated.

4. **Analyzing Application Metadata:** Metadata is a source of useful information that can be used to infer the applications capability, and therefore potential behavior. In order to access security risks, Metadata of untrusted applications is analyzed in several works. WHYPER [52] is a Natural Language Processing (NLP) based system, which uses textual application description of an application and permission list to see if application information justifies permission lists. Although WHYPER is shown to achieve substantial improvement in accuracy over keyword-based search, numerous challenges still exist for it to give a dependable, automated application risk estimate.

The mobile analysts have done many analyses for Android applications but permission analysis is done only for hypothetical analysis and to get its accuracy of risk. Many researchers state that permission based privacy analysis is not to be done by Android application developers [53].

6. Conclusion. In this paper, we have described the Android application permissions and its mechanism in detail. This paper also presented the weakness of permissions also various aspects of Android applications analytics have been reviewed along with related technologies and their limitations. We have provided better way to understand the problem that may occur in research solution and the future scope of analyzing the various Android device permissions against various risks.

REFERENCES

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

[2] Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., & Wetherall, D., *A conundrum of permissions: installing applications on an Android smartphone* (2012, February). In International Conference on Financial Cryptography and Data Security (pp. 68-79). Springer Berlin Heidelberg.

[3] La Polla, M., Martinelli, F., & Sgandurra, D., *A survey on security for mobile devices* (2013). IEEE communications surveys & tutorials, 15(1), 446-471.

[4] Leyden, J., *The TRUTH about LEAKY, STALKING, SPYING smartphone applications* (2013). The Register.

[5] Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A.,*Evolution, detection and analysis of malware for smart devices* (2014). IEEE Communications Surveys & Tutorials, 16(2), 961-987.

[6] Kelley, P. G., Cranor, L. F., & Sadeh, N., *Privacy as part of the app decision-making process* (2013, April). In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 3393-3402). ACM.

[7] 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* (2010, October). In Proceedings of the 17th ACM conference on

Computer and communications security (pp. 73-84). ACM.
[8] Felt, A. P., Greenwood, K., & Wagner, D., *The effectiveness of application permissions* (2011, June). In Proceedings of the 2nd USENIX conference on Web application development (pp. 7-7).
[9] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D, *Android permissions demystified* (2011, October). In Proceedings of the 18th ACM conference on Computer and communications security (pp. 627-638). ACM.
[10] Vidas, T., Christin, N., & Cranor, L, *Curbing Android permission creep* (2011, May). In Proceedings of the Web (Vol. 2, pp. 1-5).
[11] *Trusted mobility Index* (2012, May), Retrieved from `http://www.juniper.net/us/en/local/pdf/additional-resources/7100155-en.pdf.`
[12] *2011 Mobile Threats Report* (2012, Feb.), Retrieved from `https://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2011-mobile-threats-report.pdf`
[13] Haris, M., Haddadi, H., & Hui, P., *Privacy leakage in mobile computing: Tools, methods, and characteristics* (2014). arXiv preprint arXiv:1410.4978.
[14] Crussell, J., Gibler, C., & Chen, H.,*Attack of the clones: Detecting cloned applications on Android markets* (2012, September). In European Symposium on Research in Computer Security (pp. 37-54). Springer Berlin Heidelberg.
[15] Zheng, M., Sun, M., & Lui, J. C., *Droid analytics: a signature based analytic system to collect, extract, analyze and associate Android malware* (2013, July). In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on (pp. 163-171). IEEE.
[16] Zhou, W., Zhou, Y., Jiang, X., & Ning, P., *Detecting repackaged smartphone applications in third-party Android marketplaces* (2012, February). In Proceedings of the second ACM conference on Data and Application Security and Privacy (pp. 317-326). ACM.
[17] Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., & Song, D., *Juxtapp: A scalable system for detecting code reuse among Android applications* (2012, July). In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 62-81). Springer Berlin Heidelberg.
[18] Zhou, W., Zhang, X., & Jiang, X., *AppInk: watermarking Android apps for repackaging deterrence* (2013, May). In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (pp. 1-12). ACM.
[19] Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., & Wu, K. P.,*Droidmat: Android malware detection through manifest and api calls tracing* (2012, August). In Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on (pp. 62-69). IEEE.
[20] Kang, H. J., Jang, J. W., Mohaisen, A., & Kim, H. K., *Androtracker: Creator information based Android malware classification system* (2014). In Information Security Applications-15th International Workshop, WISA (Vol. 8909).
[21] Enck, W., Ongtang, M., & McDaniel, P., *Understanding Android security* (2009). IEEE security & privacy, 7(1), 50-57.
[22] Enck, W., Ongtang, M., & McDaniel, P., *On lightweight mobile phone application certification* (2009, November). In Proceedings of the 16th ACM conference on Computer and communications security (pp. 235-245). ACM.
[23] Chin, E., Felt, A. P., Greenwood, K., & Wagner, D.,*Analyzing inter-application communication in Android* (2011, June). In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252). ACM.
[24] Kantola, D., Chin, E., He, W., & Wagner, D., *Reducing attack surfaces for intra-application communication in Android* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 69-80). ACM.
[25] Perkins, J., & Gordon, M., *DroidSafe* (2016). Massachusetts Institute of Technology Cambridge United States.
[26] *Statista* 2017. Cumulative number of apps downloaded from the Google Play as of May 2016.
[27] Vennon, T., & Center, J. G. T.,*Mobile malware development continues to rise, Android leads the way* (2016). In FAST.
[28] *McAfee*, S. C. CA, USA.(2011). Mcafee Threats Report: Third Quarter 2011.
[29] Namestnikov, Y, *It threat evolution* Q3 2011, 2011.
[30] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., & Sheth, A. N., *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones* (2014). ACM Transactions on Computer Systems (TOCS), 32(2), 5.
[31] Vidas, T., Christin, N., & Cranor, L., *Curbing Android permission creep* (2011, May). In Proceedings of the Web (Vol. 2, pp. 1-5).
[32] Felt, A. P., Egelman, S., & Wagner, D., *I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 33-44). ACM.
[33] Zhauniarovich, Y, *Security of the Android operating system*. In International Conference on Risks and Security of Internet and Systems (pp. 272-274). Springer International Publishing.
[34] Au, K. W. Y., Zhou, Y. F., Huang, Z., Gill, P., & Lie, D, *Short paper: a look at smartphone permission models* (2011, October). In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 63-68). ACM.
[35] Geneiatakis, D., Fovino, I. N., Kounelis, I., & Stirparo, P.,*A Permission verification approach for Android mobile applications* (2015). Computers & Security, 49, 192-205.
[36] Lockheimer, H., *Android and security* (2012). Google Mobile Blog, Feb, 2.
[37] Fragkaki, E., Bauer, L., Jia, L.,& Swasey, D., *Modeling and enhancing Androids permission system* (2012, September). In European Symposium on Research in Computer Security (pp. 1-18). Springer Berlin Heidelberg.
[38] King, J, *How Come I'm Allowing Strangers to Go Through My Phone? Smartphones and Privacy Expectations.*

[39] Google play (2017), Retrived from https://en.wikipedia.org/wiki/Google_Play
[40] Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P.,*Semantically rich applicationcentric security in Android* (2012). Security and Communication Networks, 5(6), 658-673.
[41] Jeon, J., Micinski, K. K., Vaughan, J. A., Fogel, A., Reddy, N., Foster, J. S., & Millstein, T., *Dr. Android and Mr. Hide: fine-grained permissions in Android applications* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 3-14). ACM.
[42] Hornyack, P., Han, S., Jung, J., Schechter, S., & Wetherall, D, *These aren't the droids you're looking for: Retrofitting Android to protect data from imperious applications* (2011, October). In Proceedings of the 18th ACM conference on Computer and communications security (pp. 639-652). ACM.
[43] Pearce, P., Felt, A. P., Nunez, G., & Wagner, D. *Addroid: Privilege separation for applications and advertisers in Android* (2012, May). In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (pp. 71-72). Acm.
[44] Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A. R., & Shastry, B., *Towards Taming Privilege-Escalation Attacks on Android* (2012, February). In NDSS (Vol. 17, p. 19).
[45] Poeplau, S., Fratantonio, Y., Bianchi, A., Kruegel, C., & Vigna, G., (2014, February), *Execute This: Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications*. In NDSS (Vol. 14, pp. 23-26).
[46] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., & McDaniel, P., *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps* (2014). Acm Sigplan Notices, 49(6), 259-269.
[47] Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X., *Riskranker: scalable and accurate zero-day Android malware detection* (2012, June). In Proceedings of the 10th international conference on Mobile systems, applications, and services (pp. 281-294). ACM.
[48] Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., & Zang, B.,*Vetting undesirable behaviors in Android apps with permission use analysis* (2013, November). In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 611-622). ACM.
[49] Oberheide, J., & Miller, C.,*Dissecting the Androids Bouncer.* (2013, November) SummerCon.
[50] Percoco, N. J., & Schulte, S., *Adventures in bouncerland: failures of automated malware detection with in mobile application markets* (2012). Black Hat USA, 12.
[51] Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., & Hoffmann, J., *Mobile-sandbox- having a deeper look into Android applications* (2013, March). In Proceedings of the 28th Annual ACM Symposium on Applied Computing (pp. 1808-1815). ACM.
[52] Pandita, R., Xiao, X., Yang, W., Enck, W., & Xie, T., *WHYPER: Towards Automating Risk Assessment of Mobile Applications* (2013, August). In USENIX security (Vol. 13, No. 20).
[53] Tan, D. J., Chua, T. W., & Thing, V. L., *Securing Android: a survey, taxonomy, and challenges.* ACM Computing Surveys (CSUR), 47(4), 58.
[54] Oberheide, J., & Miller, C, *Dissecting the Androids Bouncer.* SummerCon, 2012.

Edited by: Kavita Sharma
Received: May 24, 2017
Accepted: Sep 1, 2017