Android App Security – A Survey

Ritu Sheoran[1], Meenakshi Chaudhary[2]

[1]M.Tech Scholar, Department of Computer Science & Engineering, Seth Jai ParkashMukandLal Institute of Technology, Radaur (Yamunanagar)

[2]Assistant Professor, Department of Computer Science & Engineering, Seth Jai ParkashMukandLal Institute of Technology, Radaur (Yamunanagar)

ABSTRACT

Android is an open source mobile platform .Android applications use the advanced hardware and software, as well as local and served data uncover through the platform to bring values to the customer. To protect that value platform can provide an application environment that make sure the security of users, data, application, device and network. The main objective of this survey paper is to study and analyze various schemes that can enhance security management in android.

Keyword :- *Support Vector Machine, Computer Vulnerability Emergency Response Team, Permissions, Malicious, Malware, Correct Permissions Set, Inter Process Communication.*

---

1. INTRODUCTION

ANDROID is most popular mobile operating system these days. Android is an open source software for mobile devices. It allow everyone to write own application and then distribute it freely .Android is a software bunch contains not only the operating system but also contains key applications and middleware. It is a powerful operating system that provides a huge number of applications in smart phones. Due to lack of control in the development of the applications and the distribution process, the user can download and install the malicious software, due to this the telecommunication network may be harm and personal information may be uncover.

Android application framework enforces the permission based security policy that only allows the installed applications to access the other parts of the system when they are permitted to do so. When the application is installed on user device then the permission is also granted by the human user. These permissions are very simple and easy to understand but there is a lot of security administrative burden and too much burden on user does not matter the user is familiar with it. Android does not impose the strong controls on the application development life cycle because of openness. The platform source code is released to the public and the public can make any changes to the code without being certified by goggle. Due to lack of control there is also a lot of vulnerabilities, especially when the security of the system is mainly depend on the user authorization decision.

1.1 Development

Android is developed by Google until the latest changes and updates are disposed to be released, at which point the source code is made public. This source code only runs without modification on the selected devices. Android source code does not contain the device drivers that are required for certain hardware components.

In 2007, Google designed the green Android logo by a graphic designer Irina Blok. The design team was a task with a project to construct globally identifiable icon with insertion of Robot in final design. After number of design developments, the team inspired from the human symbol on restroom doors and then modified the symbol into a robot shape. As android is open-sourced, so the logo must be likewise, and since its launch the green logo has been reinterpreted into numberless variations on the original design.

1.1.1 Update Schedule

Google provides additional upgrades to android every six to nine months, with candy-themed names, which most of the devices have the ability of receiving over the air. The latest release is Android 6.0 "Marshmallow".

Compared to its first mobile operating system, iOS, android updates commonly reach many devices with significant delays. Except the Google Nexus brand devices, updates mostly arrive months after the release of the next version, not at all. This is mainly due to the variation in the hardware if android devices, to which each upgrade must be tailored, as the official Google source code is only runs on their Nexus devices. For Android developers porting the Android is a time and resource consuming process, they prioritize their newest devices and leave the older ones behind. Hence, the older Smartphones are not updated many times if the manufacturer decides it is not worth the investment of resources, however the devices may be compatible. This problem is compounded when the manufacturers made the android with their own interface and apps, which should be replied to every new release.

In 2012 Google start decoupling certain aspects of the operating system so that they could be updated through Google Play Store independently to the operating system. One of these components, Google Play Services, is a closed-source system-level process that provides API for Google services, installed automatically on all the devices that is running on Android version 2.2 and higher. With these changes, Google add the new operating system functionality through application updates and Play Services without distributing the upgrade to the operating system.

1.1.2 Linux Kernel

Android's kernel is based on the Linux kernel. Since April 2014, Android devices mostly use versions 3.4 or 3.10 of the Linux kernel. The particular kernel version depends on the actual android device and its hardware platform. There are various kernel versions that are used by Android, the version 2.6.25that was generally used by Android 1.0.

In August 2011, Linus Trovalds said that " Android and Linux would come back to common kernel, but it will may not be for four to five years". In December 2011, Greg Hartman started the Android Mainlining Project, which main aim is to put Android drivers, features and patches back in to the Linux Kernel, begin in Linux 3.3

The Flash storage on Android devices is break into various partitions, such as /data for user data and application installations and /system for operating system itself. In distinction to desktop Linux distribution, Android devices owner not given root access to the operating system and sensitive partition such as /system are read-only.

1.1.3 Software Stack

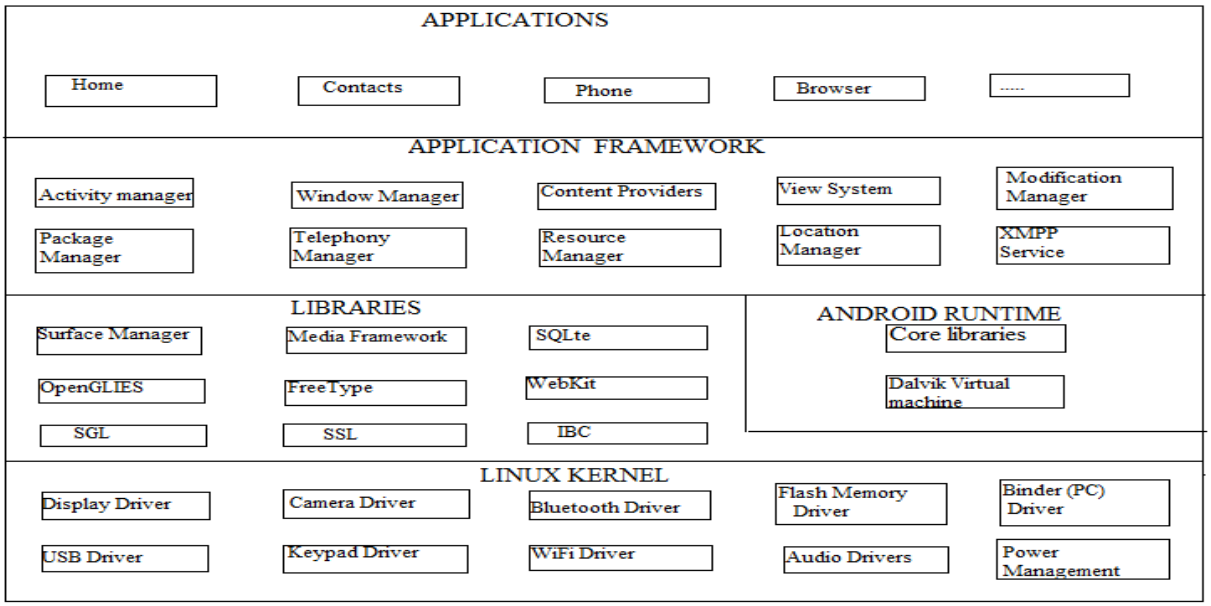

Fig-1: Android Architecture Diagram

On the top of the Linux Kernel, there are APIs, libraries and middleware written in C, and application software which is running on an application framework that contain Java-compatible libraries. Development of the Linux Kernel is continual independently of other Android source code bases.

Android 4.4 introduced Android Runtime (ART) as a new runtime environment. During installation of the application it compile the application byte code into machine code by using ahead-of-time (AOT) compilation. In Android 4.4 ART is not enabled by default, it was an experimental feature. In the next major version of Android,5.0 ,it became the only runtime option. In December 2015, Google publicized that the next version of Android would switch to java implementation based on open JDK.

Android's standard C library, Bionic, which was developed by Google for Android, as a derivation of the BSD's standard C library code. Bionic itself developed various features that are specific to the Linux Kernel. The main advantage of using a Bionic instead of GNU C Library are its smaller runtime footprint and optimization for low-frequency CPU's.

2. Security Issues

Android can implement the security issues at each device by applying permission based security policy and limit the ability of installed applications by permissions. Permissions are the character strings that are different from each other. When a permission is jump to a resource object and operations, permissions should be apply to execute the operation on the object. A set of default permissions is provided by the android framework in android manifest permission class, and it also allows us to define a new permissions. When application is installed then these new permissions should be mention in the applications and introduced into the system. At the time of application installation permission authorization is performed. When an application is installed on a device then it requests the set of permissions for the completion of its task. The lists of all the requested permissions are visible on the screen so that the user can explore them. All the requested permissions are given to the application only if the user agrees the application is installed.

3. Permission Management

Android is a popular platform for mobile devices. Applications of Android is written in java. Android has a unique permission mechanism. At development time, app writer can request the permissions by including them in configuration file. During installation each app requests the permissions, the user can analyse these permissions and allow them for the duration the app is installed.

3.1 Permission Mechanism

Flow permissions can outline as relations between three types of permission domains. Four types of flow permissions are:

- Source ⟶ Sink: flow from source domain to sink domain.
- IPC ⟶ Sink: flow from IPC source domain to sink domain.
- Source ⟶ IPC: flow from source domain to IPC sink domain.
- IPC ⟶ IPC: flow from IPC source domain to IPC sink domain.

4. Related work

4.1 Static Analysis For extracting Permission Checks

In this paper author can proposed a security architecture on android by applying protection on resources through permission checks. Even it has also some disadvantages sometimes some applications permit more permissions then they actually use, due to this all types of harmful activities are possible that can harm. A mapping is required between needed permissions and API method. In this paper author shows when static analysis are applied with off-the-shelf components on android then it fails. To extract this mapping author can shows class hierarchy and field-sensitive set of analysis[5].

4.2 Application for malicious application detection

Today's era there is a lot of malicious application that can harm very badly to the devices. There is a big challenge for detecting these malicious apps and keep these malicious away from market apps. Permission control is one of the major security mechanism for android that can restrict the access of apps. In this paper author suggests the permission induced risk on three levels in a planned manner. Firstly author can examine the risk in permissions individually and then in group. According to risk a rank is provided to each permission through three ranking methods i.e .mutual information, T-test, correlation coefficient. Secondly author can use some methods like vector machine, decision tree and random forest for accessing the effectiveness of risky permissions for detection of malicious apps .Thirdly detection results are analyzed [4].

4.3 Composition of Inter-app vulnerabilities

Android is an open source software that are widely used in mobiles. Through inter-app communication system it can share services and data among applications .Android permission system can handle all the access resources but these are not enough to stop such security violations, as permissions may be ruined consciously or unconsciously .In this paper author can proposed a COVERT tool for compositional study of android inter-app vulnerabilities. COVERT's analysis is a step by step analysis of applications as they are installed, updated and removed [6].

4.4 Flow permission for android

In this paper author purposed a flow permission for android which is an extension of permission mechanism. Distant from the existing permission mechanism this mechanism contains semantic information that is form information flows. These flow permissions allow the users to inspect and allow straight forward information flows within the applications as well as explicit information flows across various applications. Authors aim is to offer visibility to the applications that are installed on user's phone [3].

4.5 Formal model to analyze the permission

In this paper a formal model is used to inspect the permissions and also describe the scheme that can specify the entities and relationship. Even the author can also provides a state based model that includes the interaction between application components and the behavior description of permission authorization. The author also shows how the security of the specified system is logically confirmed [1].

4.6 Assessment method for android applications based in permission control

Applications are controlled to access the particular resources by permission based security model of android but malicious applications can occupy easily in user-centric pattern. Through the study of permission features of android applications and permission based security model, author can establish the permission model so that the functional characteristics of the applications are quantified and then visualization techniques and clustering algorithms are used for testing the applications weather they are malicious or not that can help the users to select the applications before installation [2].

4.7 A Novel Approach To Detect Android Malware

Android is world's most popular and largest base of any mobile platform. It has gained a great popularity among Smartphones and is growing very fast because it provide its users a world class platform for creating the applications and games and allows them to be distributed. Secondly, it also offers free third party apps to be downloaded and installed from Google Play, the first marketplace for selling and distributing Android apps. Android openness made has made it favourite for users and developers. Every month user can download many apps from play store. However due to this, various harmful apps in the shape of malwares getting downloaded are also increasing. These malwares executes the various activities behind the scene, such as stealing the sensitive information of the users. As a result of this, users are getting affected and their privatness gets compromised. As developers are also free to develop and publish their own creation in play store without undergoing any security of their apps, they tend to take the benefit of user's inability to examine the threats of such harmful apps. In this paper Author purposed a system which would help the users in studying and removing such harmful apps and thereby

saving their privacy and security. This is achieved by analyzing the permissions of the application that it has requested during installation. The process of analyzing permissions is done by using clustering and classification techniques.

4.8 SUMMARY OF VARIOUS PURPOSED TECHNIQUES BY VARIOUS AUTHORS

Comparison of various purposed techniques that are discussed above along with their findings is given in Table-1 as below:

AUTHOR & YEAR	TECHNIQUE USED	FINDINGS
Wook Shin Shinsaku Kiyomoto Kazuhide Fukushima Toshiaki Tanaka in 2010	A scheme based on entities and relationship.	Build a formal model of the android permission scheme. Specification that express authorization and permission - protected interactions among application components.
Danyang Jiang Xiangling Fu Maoqiang Song Yidong Cui 2012	Encryption/decryption technology. visualization technique and clustering algorithm	Focus on the permissions of the application to detect the malicious applications.
Shashank Holavanalli Don Manuel Vishwas Nanjundaswamy Brian Rosenberg in 2013	Flow permission mechanism, SOOT java optimization framework Tested blue seal	Explicit information flows within application as well as implicit information flows across multiple applications. Detail of MyCalendar , MySpace, Blackmoon File Browser , Gmail apps. CHEX provide tool for detecting highjack enabling flows with an app.
Wei Wang Xing Wang Zhen Han in 2014	Employ algorithms SVM, Decision tree, random forest to detect malapps.	Detect malapps with permission matrix and construct decision rules
Alexandre Bartel Jacques Klein in 2014	Class Hierarchy Analysis on android. COPES tool	COPES tool is used to find a non negligible part of application suffers from permission gaps Analyzing permissions through permission based model.
Hamid Bagheri Alireza Sadeghi Sam Malek in 2015	COVERT tool.	Static analysis to automatically recover models that reflect android apps and interactions among them. Identify vulnerabilities due to interaction of multiple apps.
Shaikh Bushra Almin MadhumitaChatterjee	Android application analyzer, Clustering and classification techniques.	Analyze and remove the harmful apps by using android application analyzer.

Table-1 : Comparison between the purposed Techniques by various Authors

5. CONCLUSION

In this survey we studied various techniques that can enhance security in android applications. A scheme that is based on entities and relationship builds a model of android permission scheme. Encryption decryption technique is to be used that pay attention on the features of applications to find malicious application. The algorithms like SVM, decision tree, random forest to detect malicious apps. The behavior of installed apps are identified with the help of permissions that are granted upon installation. This helps the user to identify the malicious apps. Even we can also identify the strength and weakness of the android apps with respect to its functionality.

6. REFERENCES

[1].Wook Shin, Shinsaku Kiyomoto, Kazuhide Fukushima, and Toshiaki Tanaka," A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework" 978-0-7695-4211-9/10 $26.00 © 2010 IEEE DOI 10.1109/SocialCom.2010.140

[2].Danyang Jiang, Xiangling Fu, Maoqiang Song, Yidong Cui," A Security assessment method for android applications based on permission model" 978-1-4673-1857-0/12/$31.00 ©2012 IEEE.

[3].Shashank Holavanalli, Don Manuel, Vishwas Nanjundaswamy, Brian Rosenberg, Feng Shen," Flow Permissions for Android" 978-1-4799-0215-6/13/$31.00 c 2013 IEEE.

[4].Wei Wang, Xing Wang, Dawei Feng, Jiqiang Liu, Zhen Han, and Xiangliang Zhang, Member, IEEE,"Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection" 1556-6013 © 2014 IEEE.

[5].Alexandre Bartel, Jacques Klein, Member, IEEE Computer Society, Martin Monperrus, Member, IEEE, and Yves Le Traon, Member, IEEE Computer Society," Static Analysis for Extracting Permission Checks of a Large Scale Framework: The Challenges and Solutions for Analyzing Android" 0098-5589 _ 2014 IEEE.

[6].Hamid Bagheri, Alireza Sadeghi, Joshua Garcia and Sam Malek," COVERT: Compositional Analysis of Android Inter-App Permission Leakage" 0098-5589 (c) 2015 IEEE.

[7].Shaikh BushraAlmin, Madhumita Chatterjee ,"A novel approach to detect Android Malware" 1877-0509 © 2015