ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

Application Permission Manager using Smartphones

Mayur Bherade, Pranit Kokate, Sagar Dalvi, Nagraj Marathe, Prof. Bhavani V.

B. E Students, Department of CE, Alard College of Engineering & Management, Marunje, Pune, India

Professor, Department of CE, Alard College of Engineering & Management, Marunje, Pune, India

ABSTRACT: The permission managers of an android application will give choice when installing any new Android application from the play store in our system of either accepting all request or nothing. There will be an android application that can alter the decision of the application after installing any Android application that you have called as authorization manager (permission manager). However, one of the problems with this Android application can access the revoked authorization due to the loss of authorization (permission leaks). To address these problems,defining the level of middleware to defend against the loss of authorization. The middleware is ensuring that it can effectively block all license losses. It requires minimal changes in the authorization manager and also in the Android operating system. The results show that the middleware platform works with low power consumption and less overhead.User get all the information about permission required to that application then user can block some permissions. User can view block the permission and then update that block permissions. Changing that block permission update the status to the system. User can view all the updated permission by the different applications.

KEYWORDS: Permission Manager, Permission Leaks, Middleware, Android

I. INTRODUCTION

A. Background:

Once the vulnerable applications are installed on Android, find a common and serious problem: none of them could defend themselves against loss of authorization, loss of permissions such as vibrating, change of WiFi status, phone call, send SMS and the camera. This means that an application can still access confidential resources even if its user has revoked the corresponding permissions through a certain authorization manager. The result is serious because most users tend to rely on permission managers in order to effectively apply user-defined access control. Motivated to address the above problem, a system-level middleware is proposed for administrators of existing authorizations to protect against authorization losses. To handle a list of blocked permissions, will present the concept of blocked permission.

B. Motivations:

Showing that middleware meets design requirement. The main challenge is achieved by attaching blocked permission instance of each android application. Stopping the permission leaks which are caused by the summation of two application. In that application user can update the block status of installed applications.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

C. Objective and Scope:

The main objectives of this application is,
1.The middleware must be able to defend itself effectively against the loss of authorization at runtime. Any Android permission manager compatible with proposed middleware can so reliably enforce permissions granted or revoked in the presence of vulnerable applications.

2.User can login with authenticate username and password and also by using fingerprint security. If fingerprint is match then user can get access to the application.

3.Only authenticate user can access thisapplication. The middleware is transparent for users and application developers. It provides public interfaces for existing authorization managers to exploit in order to be secure against license losses.

II. RELATED WORK OR LITERATURE SURVEY

[1].P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin(2011) has represents Permission re-delegation: attacks and defenses [1]. The author represents the new delegation of permissions and demonstrates his risk when launching attacks of the real world in applications of the Android system; many of the vulnerabilities have been confirmed as errors. They discussed the possible ways to deal with the new delegation of permits and presented the IPC Inspection, a new OS mechanism to defend against the new delegation of permits. The IPC inspection avoids the possibility of delegating permissions by reducing the permissions of the application after receiving communication from a less privileged application. They have implemented IPC Inspection for a browser and Android and they show that it prevents the attacks that find in the Android system applications.

[2].M. Grace, Y. Zhou, Z. Wang, and X. Jiang(2012) has demonstrate Systematic detection of capability leaks in stock android smartphones [2].The author analyzes eight popular Android smartphones and discovers that the images of the backup phones do not correctly apply the authorization model. Several privileged permissions are inappropriately exposed to other applications that do not need to require them for their actual use. To identify these permissions or leakage capabilities, the author developed a tool called Carpenter. Using them, an unreliable application can delete user data, send SMS messages or record user conversations on affected phones, all without asking permission.

[3].L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang(2013) has represents The impact of vendor customizations on android security[3].The author analyzes ten representative images of the Android actions of five popular smartphone vendors (with two models from each provider). Their objective is to evaluate the scope of the security problems that can be introduced through the personalization of the provider and to determine over time the evolution of the situation. In particular, let's take a three step process: first, provide a backup image of a smartphone, let's analyze the source to classify each application in the image into three categories: AP applications,custom applications or written the provider and the members of third parties that are simply included in the stock image. This analysis of origin allows you to correctly attribute the security problems detected in the Android images examined. Secondly, they analyzed pre-loaded application authorization modes to identify overlapping subjects that unnecessarily require more Android permissions than those actually used. Finally, the vulnerability analysis detects a pre-programmed bug application that can be used to mount new delegate authorization attacks or lose private information.

[4].L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang(2012) has designed Chex: statically vetting android apps forcomponent hijacking vulnerabilities [4]. The author designed CHEX, a static analysis method that automatically controls Android applications for component hijacking vulnerabilities. By modeling these vulnerabilities from a data flow analysis perspective, CHEX analyzes the Android applications and detects any sequence of hijacking enable elements, performing low-range scope tests on custom graphics of system dependency. To address the challenges of analytics imposed by the paradigm of special programming of Android, they use an innovative technique to discover the points

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

of entry of the components in their integrity and introduce the subdivision of the application to model the asynchronous executions of multiple access points in an 'application.

[5].Chin, A. P. Felt, K. Greenwood, and D. Wagner(2011) has represents Analyzing inter-application communication in android [5].Modern operating systems for smartphones support the development of third-party applications with open system APIs. In addition to an open API, the Android operating system also offers a rich messaging system between applications. This encourages collaboration between applications and reduces developer costs by facilitating the reuse of components. Unfortunately, message switching is also an application attack surface. The content of the message may be smelled, modified, stolen or replaced, which may compromise user privacy. In addition, a malicious application can inject false or malicious messages that can cause violations of the user's data and violate the security policies of the applications.

[6].K. Yang, J. Zhuge, Y. Wang, L. Zhou, and H. Duan (2014) has demonstrate Intentfuzzer: Detecting capability leaks of android applications [6].The author proposes a dynamic fuzzing mechanism to detect vulnerable applications in both the Android and closed-source ROM markets. They created a prototype called IntentFuzzer. With having analyzed more than 2000 Android applications on Google Play and hundreds of ROM applications within two closed sources.

[7].L. Li, A. Bartel, J. Klein, and Y. le Traon (2014) has developed Automatically exploiting potential component leaks in android applications [7].The author presents PCLeaks, a tool of vulnerability of intercomponent communication (ICC) to perform data analysis of Android applications to detect leaks of potential components that could be exploited by other components. To evaluate their approach, they run PCLeaks in 2000 randomly selected apps from the Google Play Store. The PCLeaks reports report the loss of 986 potential components in 185 applications. For each PCLeaks leak, PCLeaksValidator automatically generates an Android application that seeks to exploit the loss.

[8].P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner (2011) has represents Android permissions demystified [8].The author studies Android applications to determine if Android developers have less privileges with their permission requests. They created Stowaway, a tool that recognizes the privilege of Android compiled applications. Stowaway determines the set of API calls that an application uses and, therefore, the API call assigns permissions. They use automated test tools in the Android API to create the authorization map needed to detect privileges.

[9].Bartel, J. Klein, Y. Le Traon, and M. Monperrus (2012) has demonstrate Automatically securing permission-based software by reducing the attack surface: An application to android[9].The author presents an approach for locating authorization voids using static analysis. When using tool in a set of Android applications, they discovered that an insignificant part of the applications suffers from permissions, which means that they do not use all the permissions that they declare.

[10].X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos (2012) has represents Permission evolution in the android ecosystem [10].The author probably presents the first long-term study that focuses on the evolution and use of permissions throughout the Android ecosystem (platform, third-party applications and pre-installed applications). First, let's study the Android platform to see how the set of permissions has evolved; They believe that this tends to grow and that growth is not intended to grant permissions to the end user, but rather to access new hardware capabilities; A particular concern is that the set of dangerous permits is increasing. Second, examined third-party Android applications and pre-installed to see if they follow the principle of minimum privilege. They believe that this is not the case, since a growing percentage of popular applications that they are studying is more privileged. In addition, applications tend to use more permissions over time. Third, highlighted some concerns with pre-installed applications, such as applications distributed by distributors with the phone; these applications have access to and use a wider set of privileged privileges that pose security and privacy risks. At the risk of over-application, they affirm that the Android ecosystem is becoming more secure from the user's point of view.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

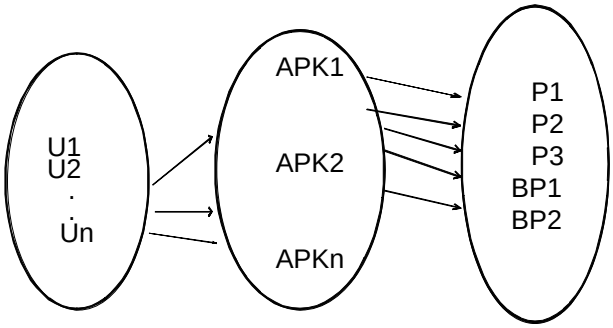
International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

III. MATHEMATICAL MODELING



In a mathematical model, contains
Where.
U1,U2.....Un=No.Of User
APK1,APL2,APK3= No.of APK in Mobile
P1,P2,P3=Permission of application
BP1,BP2=Block permission of application.

Set Theory:-
S={s, e, X, Y,Φ}
Where,
s = Start of the program.
1. Log in System
2. View the install APK List in mobile
e = End of the program.
View Permission of APK Files
X = Input of the program.
Input should be APK file Name
Y = Output of the program.
Using APK file name user can view all the permission of that APK file.
X, Y ∈ U

Let U be the Set of System.
U= {U1,U2,APK1,APK2,P1,P2,BP1,BP2}
Where U1,U2,APK1,APK2,P1,P2,BP1,BP2are the elements of the set.
U1=User1
U2=User2
APK1=Android Application Package 1
APK2= Android Application Package 2
P1=Permission 1
P2=Permission 2
BP1=Block Permission 1
BP2=Block Permission 2

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

Failures:

1. Huge database can lead to more time consumption to get the information.
2. Hardware failure.
3. Software failure.

Success:

1. Got file which want to search.
2. Got accurate file.

Above mathematical model is NP-Complete.

IV. EXISTING SYSTEM AND DISADVANTAGES

The responsibility of permission managers is to granted or revoked permissions. The loss of permissions can be considered as an attack where an application without privileges can access certain confidential resources without declaring the corresponding permissions. This means that an application can still access confidential resources even if its user has revoked the corresponding permissions through a certain permission manager. The result is serious because most users tend to trust authorizations so that they can effectively apply access controls defined by the user.

Disadvantages:

- It is not flexible and does not provide fine-grained access control to user.
- Generate permission leakage problem.

V. PROPOSED SYSTEM AND ADVANTAGES

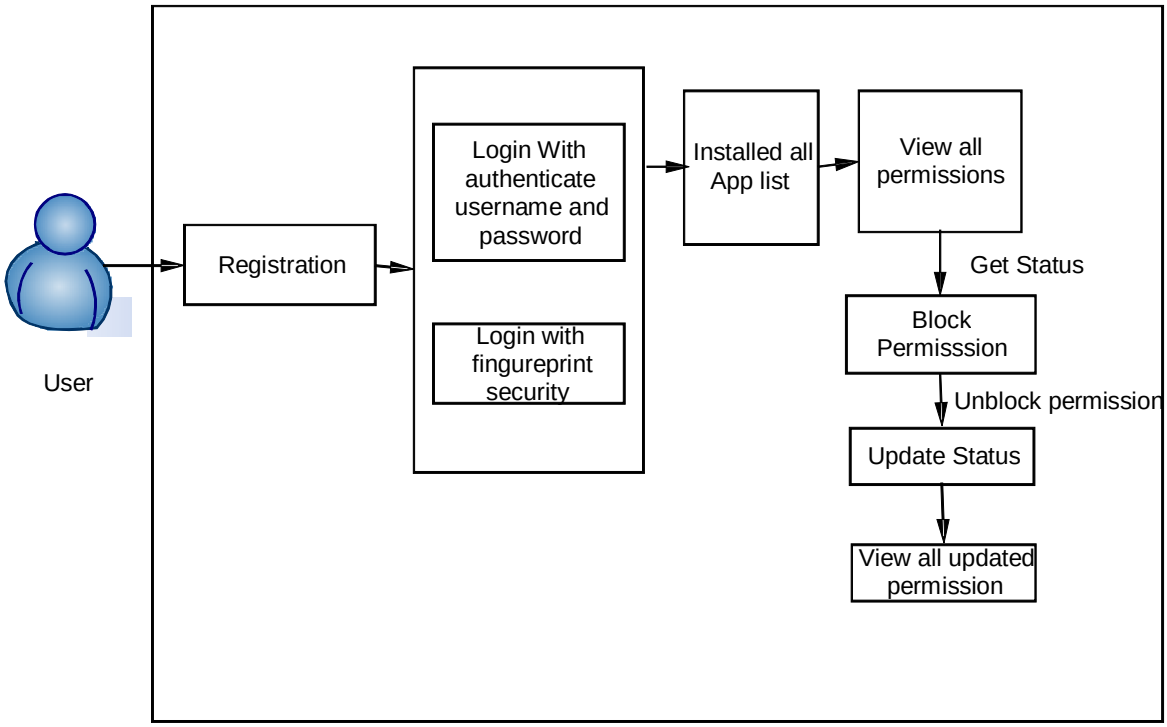


Figure: Proposed System Architecture

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

The proposed system, propose an OS-level middleware for existing permission managers to defend against permission leaks. In a proposed System Architecture User first registration to the system.If registration are get successfully done then user got sms which is entered by the user. Login this system by 2-way first way is with entering authenticate username and password and second by using fingerprint security.If fingerprint is match then user can get access to the system.After login user can view the installed application list which are present in android phone.On click of any application information user get all thepermission required to that application then user can block some permissions.User can view block the permission and then update that block permissions. Changing that block permission update, the status to the system.Finally user can view all the updated permission by the different applications.

Proposed System Advantages:

1.The main problem which is known as permission leaks which occurs due to summation two application blocked permission list which is prevented by this android application.
2.User got SMS on registered mobile number at a registration time.
3.User can view total permission of all installed application and total permission of particular installed application also.
4.User can login with authenticate username and password and also byusing fingerprint security. If fingerprint is match then user can get access to the application.
5.This application can prevent the access of permission which are activated by some other application or revoke the permission which saves android device from accessing the sensitive information.
6.It is more flexible and provides fine-grained access control to user.

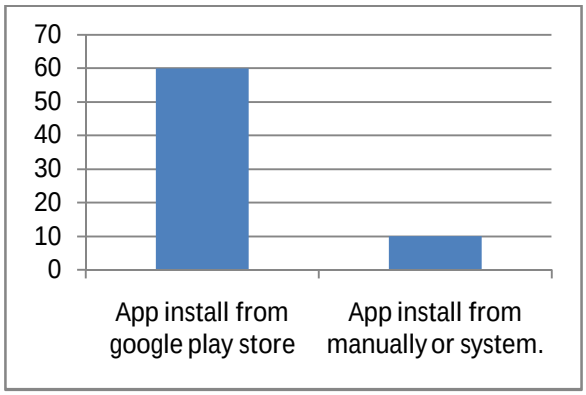
VI. SYSTEM ANALYSIS

In this experimental setup, identified that in user mobile calculate total number of apk files.In a Particular APK file check the required permission of that APK file. This system can have classifiedin 2 categories. First app install from google play store and Second app install from manually or system.

Sr.No	App install from Google play store	App install from manually or system.
1	60	10

Table 1:- Application List in Users Mobile

Table no 1showsthat in usersmobile 60 App install from Google play store and 10App install from manually or system.



ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 3, March 2018

Graph 1:- Application List in Users Mobile

Above graph 1 showing that in users mobile 60 App install from Google play store and 10App install from manually or system.

VII. CONCLUSION

It was a great challenge to provide flexible and refined permissions in Android applications. Although several Android authorization handlers (permission managers) have been developed, found that none of the existing permission managers investigated correctly applied user-defined access control due to the loss of authorization. To solve this problem, thisproposed middleware that can be used by any existing permission manager. The middleware is designed to be (i) protected from any loss of authorization (permission leaks); (ii) highly usable with minimal impact on the use of any running application; (iii) User can login with authenticate username and password and also by using fingerprint security. If fingerprint is match then user can get access to the application.

REFERENCES

[1] P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin,"Permission re-delegation: attacks and defenses," in Proceedings of the 20th USENIX conference on Security, 2011

[2] M. Grace, Y. Zhou, Z. Wang, and X. Jiang, "Systematic detection of capability leaks in stock android smartphones," in Proceedings of the 19th Annual Symposium on Network and Distributed System Security, 2012

[3] L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang, "The impact of vendor customizations on android security," in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 623–634. [Online]. Available: http: //doi.acm.org/10.1145/2508859.2516728

[4] L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang, "Chex: statically vetting android apps for component hijacking vulnerabilities," in Proceedings of the 2012 ACM conference on Computer and communications security, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 229–240

[5] Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in android," in Proceedings of the 9th international conference on Mobile systems, applications, and services, ser. MobiSys '11. ACM, 2011, pp. 239–252.

[6] K. Yang, J. Zhuge, Y. Wang, L. Zhou, and H. Duan, "Intentfuzzer: Detecting capability leaks of android applications," in Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ser. ASIA CCS '14. New York, NY, USA: ACM, 2014, pp. 531–536. [Online]. Available: http://doi.acm.org/10.1145/2590296.2590316

[7] L. Li, A. Bartel, J. Klein, and Y. le Traon, "Automatically exploiting potential component leaks in android applications," in Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on, Sept 2014, pp. 388–397.

[8] P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security, ser. CCS '11. ACM, 2011, pp. 627–638.

[9] Bartel, J. Klein, Y. Le Traon, and M. Monperrus, "Automatically securing permission-based software by reducing the attack surface: An application to android," in Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on. IEEE, 2012, pp. 274–277.

[10] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the android ecosystem," in Proceedings of the 28th Annual Computer Security Applications Conference, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 31–40.