Formal Modeling of Security Concerns in Android

Khadija Javed¹, Maria Tariq²

¹ Department of Computer Science, GC University Lahore

² Department of Computer Science, Lahore College University for Women Lahore
khadijajaved17@yahoo.com

Abstract:

The need of providing a secure environment to the users of technology is necessary to keep it going. Android devices are used by most of the population worldwide, to keep it working and developing it should be secure for the users. Applications are installed on the device by the user for specific purposes. Different applications interact with each other to perform some specific functions e.g. an application that doesn't have its built-in Calendar functionality asks for the permission to access it externally from another application/s installed on the device and this inter-application communication can result in data theft vulnerabilities because of communication with a malicious application directly or indirectly. We present a defense mechanism model named PBAD (Permission Based Attack Defense) Model, which protects the applications from interacting with malicious applications and protecting the permission protected interfaces of the innocuous applications. Our main focus is on the permission related security measures because the permission model of the Android OS is coarse-grained and it is vulnerable to attacks. The presented model is a PROMELA based model.

Keywords: PROMELA, PBAD Model, Formal Methods, Permission-based attacks, Android OS.

1. INTRODUCTION

1.1. *Android:*

Android is a software platform for mobile devices that includes an OS, middleware and key applications. It is a mobile OS based on Linux Kernel. It was first developed by Android Inc. but later bought by Google in 2005. More than a million devices are activated all over the world every day. Initial versions of android applications supported java, C/C plus or .NET but now Android applications are developed using Java as-well. There are different versions of Android from its birth till now. They have improved graphically, functionally and conceptually. Android has a layered architecture consisting of Linux Kernel, Hardware Abstraction Layer, Native Libraries, Java API Framework and System Applications.

1.2. *Formal Methods:*

For the verification of Safety-Critical Systems, Formal methods are widely used. The system is modeled into a language for

specification and then verification is done using verification language. For this purpose, there are different verification and specification languages. The Failure of the system could result in loss of life and these systems refer to the safety-critical system. Hence, if the risks are kept at a level of acceptance a system is in a safe state. They can detect the issues at the earlier stages which results in the protection of the system from future disasters. To improve the quality of the system and to reduce the cost of its design many formal methods have been introduced.

1.3. *Formal Methods in Android Security:*

Android is an open-source mobile software, it's vulnerable to external and internal attacks. Being an open-source platform it needs solid security architecture to prevent any kind of attack. It has multi-layer architecture; every layer has its own security properties. Formal methods are used in the security analysis and improvement of Android devices as well. Android is open-source mobile software, it's vulnerable to external and internal attacks.

Being an open-source platform it needs solid security architecture to prevent any kind of attack. It has multi-layer architecture; every layer has its own security properties.

Android provides security controls to help developers. Security controls that it provides, include thorough security at the level of the operating system through the Linux kernel, application sandbox for applications, signing feature for applications, and permissions granting by users and defined by the application.

Linux Kernel is the core of Android security. It provides a user-based permissions model, process isolation, a mechanism for secure Interprocess communication. Access of user data to applications is provided by some protected APIs. Basically, Overtime Android devices are going to store user data in the applications that are downloaded on them. Some applications can choose to share this data and can use the Android OS permission checks to protect it from other parties. Users should be clear what kind of permission an application is trying to access before granting it to protect data from unauthorized access. Applications can share data through permissions applied to the secure IPC mechanisms used by the OS.

Android OS faces many security issues. Some of those security issues are discussed here; Android apps that are uploaded on its market has no security scan. Applications can exploit the services of other applications without permission requests. The Open Source nature of Android is available to hackers as well, which makes it vulnerable to security threats. Android does not provide security for external storage.

Even after so much thorough security measures taken by Android, it is still not as secure as it seems. If a user installs a suspicious application or allows permission to the application without thorough knowledge about it, it will always be risky. Many researchers have worked in the field of Android security. In [1], Android application security has been studied. The ded decompiler has been introduced by the researchers, in which the installation image is generated from the Android application source code. In [2], researchers did a survey 'Android Security Framework'. Security risk is increasing because of the increased open-source nature of the smart phone. To provide security to the phone, Android provides some set of permissions. In [3], improvement in the security of android based systems is presented. In [4], the Android Security case study was presented with

Bauhaus. On Android OS-based devices a case study was made, in which collaboration with security experts was made and a reverse engineering tool suite was employed for security assessment. In this work, some inefficiency was found in the implementation of concepts related to Android Security. Different research topics were proposed based on the case study mentioned in the paper discussed for security assessments to help the security analysts. In [5], researchers studied linguistically rich Security in Android which was centered on Application, in which enhancements were made to the existing android OS with a framework to meet security requirements. In [6], study is done on improving the security of Linux based Android devices. In [7], study is done on the permission-based security model and collusion attack and its inference for modern life smartphone systems. In [8], a generic OS framework is presented that provides security in smartphones without the need for hardware extensions. A smartphone OS was encapsulated in a virtual machine, which allows applications to run in a parallel manner securely with the virtual machine. In [9], a discussion has been done on improving the android permission model and in the improvement of the quality of it.

Android users need to know if their data is leakage proof or not, for that purpose researcher created AndroidLeaks, which reduces the number of applications and manual verification of paths by security auditor. In recent years formal methods have been used in the analysis, detection, and deletion of malware. By the use of model checking, work on the identification of malicious payload in repackaged applications is done [10]. In [11], authors have introduced CTPL (Computation Tree Predicate Logic) which is an extension of CTL (Computation Tree Logic), and a well-worked model checking algorithm is also provided.

Windows malware variants is also confirmed by the help of a dataset of worms dating from 2002-2004. Quite a work has been done on malware detection and prevention of informal methods in [12].

Permission-based security threats and breaches are one of the most critical issues that have quite an impact on Android security. Our aim is to detect these attacks and how they can be avoided to keep applications secure from getting infected by the maliciousness of malicious applications.

We are going to provide defense against these attacks in Inter Application working mechanism. We call this system as PBAD (Permission Based Attack Defense). Previously not much work is done on detection and protection of permission induced attacks. Especially, Formal aspects for detection and prevention are not discussed which can secure the spreading of maliciousness from one application to another. According to [13], permission-based defects that have been identified in Android can result in security breaches and disturbance in the privacy of the user. Details of the problem are discussed in the section below for better understanding.

Our focus is on Android Inter Application Security because previously many works have been done on single application security techniques but not much on applications that are communicating with each other legally or illegally. Android has the most sophisticated communication scheme that is the reason our focus is on the Android security model, even though other security models are similar to Android.

2. PROBLEM STATEMENT

In this section, the problem is discussed in detail. Attacks that are discussed in this work are permission-based so before explaining those attacks we need to get a look into what is Permission. The objective of permissions is to provide protection and security to Android users. To access the sensitive data of the user an application is required to request permission. After verifying the reason for the permission request, the system grants it.

The attacks that are going to be addressed in this paper include, similar custom permission attack, privilege escalation attack. These attacks reach out to the vulnerabilities of other applications and threaten their security.

Permissions are either Built-in i.e. predefined or custom in nature. Built-in permissions include Location, SMS, etc., while custom permissions are defined by Android applications and are requested by other applications. Custom permissions are being requested from other applications, therefore at this point lays the vulnerability that can threaten the security of the application. This can be cleared via the statement that states: Among two Android applications, the one that is installed first on the system is the one whose permission

definition is used in case define the same Custom Permission [14].

2.1. Similar Custom Permission Attack:

If an application is protected by a Custom Permission and malicious application tries to access the interface of some other application illegally then Similar Custom Permission attack vulnerability is misused. TOCTOU vulnerability is also caused by the naming collision of permissions [15].

Two applications in which one of them is malicious while the other one is victim have defined the same custom permission that is named as CP1 and CP2 in manifest file under permission tag. In this scenario, malicious application is installed before victim application so, the permission definition of malicious application is used i.e. CP2 and this is defined at a Normal protection level. Therefore, the interface that is defined by the victim application is accessible by the malicious application, which should only be accessible to those requesting the custom permission CP1, such as benign application. This attack can happen even if the protection levels of the two permissions are different but have the same name.

2.2. Privilege Escalation:

This attack clearly shows Android malware is not limited to an application sandbox because it allows a malicious application to join forces with other applications to access resources that are critical without explicitly requesting permission [16].

When a less privileged application is allowed to access the components of more privileged application, then this attack occurs [17] the permission-protected interface of the privileged application is reached by the malicious application indirectly, by exploiting the vulnerability of the Victim Application.

3. METHODOLOGY: MODEL CHECKING OF SECURITY THREATS

In this section, a model is presented to secure the applications from getting attacked. Model Checking approach is provided to create a secure mechanism. For model checking, code is written in ProMela which uses Atomic propositions.

Atomic propositions used in the code are explained below as: appeal: malicious application, appvic: victim application, reqp: permission requested by an application, permgrant: permission granted to the application and attacked: application is attacked with maliciousness.

The first Attack that we discussed was Similar Custom Permission Attack. In this attack, we see if a malicious application is installed on device with the same custom permission as the one which another application is trying to declare, then we'll simply not declare the permission to that application because if it'll be installed then it can access the data of the other applicants because of the naming collusion. We'll only declare custom permission 1 to another application if it's not declared by the malicious application already to ensure the protection of data. PROmela code for it is shown below:

```
activeproctype permission () {
2 SCustP:
3
4 If
5 :: app1 = malapp && app2 != malapp->
6 goto SCustP;
7
8 :: app1 != malapp && app2 != malapp;
9 goto acceptcp;
10
11 fI;
12
13 acceptcp:
14 cp2->cp1;
15 }
```

Second attack that we discussed is Privilege Escalation Attack. As discussed in previous section, this attack occurs if an application with less privilege tries to access the one with more privilege. For this we'll see if the application (malapp) that's trying to access the more privileged application (vicapp) is active or not, then we'll see if the vicapp has requested for the permission or not. If malapp is active and vicapp hasn't requested the access then simply not grant the permission access to the application. For vicapp to be granted permission access make sure the malapp is unactive and vicapp has requested the access to the permission. PROmela code for this attack is shown below:

```
3 activeproctype prvesc () {
4 Pesc:
5 If
6 :: appmal == actapp->
7 goto Pesc;
8 :: appvic != reqp->
9 goto Pesc;
10 :: appvic == permgrant->
11 goto Pesc;
12 :: appmal != actapp;
13 :: appvic == reqp->
14 appvic = permgrant;
15 goto accept;
16 fI;
17
18 accept:
19 printf("Permission Granted");
20 }
```

4. CONCLUSION

Permissions are used to protect the device from getting unwanted access but even after the use of permissions the device can be infected by an international application which can be a result of inter-application communication. We designed a model named PBAD to identify Android applications of permission related vulnerabilities and then presented the solution to prevent the applications to get infected by the maliciousness of other applications or to prevent the passing of malware from one application to another and in result getting access to sensitive resources of that application.

In the future, we can extend our work to prevent and identify more vulnerability related to permissions.

References

- [1] Zhou, Y., & Jiang, X. (2012, May). Dissecting android malware: Characterization and evolution. In 2012 IEEE symposium on security and privacy (pp. 95-109). IEEE.
- [2] Powar, S., & Meshram, B. B. (2013). Survey on Android Security Framework. International Journal of Engineering Research and Applications, 3(2), 907-911.
- [3] Kaur, S., & Kaur, M. (2013). Review Paper on Implementing Security on Android Application. Journal of Environmental Sciences, Computer Science and Engineering &

Technology, 2(3).

[4] Berger, B. J., Bunke, M., & Sohr, K. (2011, October). An Android security case study with bauhaus. In 2011 18th Working Conference on Reverse Engineering (pp. 179-183). IEEE.

[5] Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P. (2012). Semantically rich application-centric security in Android. *Security and Communication Networks*, 5(6), 658-673.

[6] Schmidt, A. D., Schmidt, H. G., Clausen, J., Yuksel, K. A., Kiraz, O., Camtepe, A., & Albayrak, S. (2008, October). Enhancing security of linux-based android devices. In *Proceedings of 15th International Linux Kongress* (pp. 1-16). Lehmann.

[7] Marforio, C., Francillon, A., & Capkun, S. (2011). Application collusion attack on the permission-based security model and its implications for modern smartphone systems. *ETH Zurich*. Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A., & Peter, M. (2011, October). L4Android: a generic operating system framework for secure smartphones. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (pp. 39-50). ACM.

[8] Barrera, D., Kayacik, H. G., Van Oorschot, P. C., & Somayaji, A. (2010, October). 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* (pp. 73-84). ACM.

[9] Battista, P., Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2016, February). Identification of Android Malware Families with Model Checking. In *ICISSP* (pp. 542-547).

[10] Kinder, J., Katzenbeisser, S., Schallhart, C., & Veith, H. (2005, July). Detecting malicious code by model checking. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 174-187). Springer, Berlin, Heidelberg.

[11] Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2016, June). Ransomware steals your phone. formal methods rescue it. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems* (pp. 212-221). Springer, Cham.

[12] Song, F., & Touili, T. (2012, August). Efficient malware detection using model-checking. In *International Symposium on Formal Methods* (pp. 418-433). Springer, Berlin, Heidelberg.

[13] Fang, Z., Han, W., & Li, Y. (2014). Permission based Android security: Issues and countermeasures. *computers & security*, 43, 205-218.

[14] Bagheri, H., Kang, E., Malek, S., & Jackson, D. (2015, June). Detection of design flaws in the android permission protocol through bounded verification. In *International Symposium on Formal Methods* (pp. 73-89). Springer, Cham.

[15] Shin, W., Kwak, S., Kiyomoto, S., Fukushima, K., & Tanaka, T. (2010, July). A small but non-negligible flaw in the android permission scheme. In *2010 IEEE International Symposium on Policies for Distributed Systems and Networks* (pp. 107-110). IEEE.

[16] Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., & Sadeghi, A. R. (2011). Xmandroid: A new android evolution to mitigate privilege escalation attacks. *Technische Universität Darmstadt, Technical Report TR-2011-04*.

[17] Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A. R., & Shastri, B. (2012, February). Towards Taming Privilege-Escalation Attacks on Android. In *NDSS* (Vol. 17, p. 19).