ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2017

VULHUNTER: An Update Based Privilege Escalation Checker for Applications

Ramachandra Reddy Avula*

Sphoorthy Engineering College, Nadargul, Hyderabad, India

E-mail: rcreddy.aavula@gmail.com

Abstract: With the increasing demand for smart phones, the vulnerability to security attacks has also been increased. Our objective is to develop an application that shields or verifies any attacks on the user's device in the form of updates. Google added a new layer of security to the Android Market, dubbed Bouncer that will scan apps for evidence of malware. The effort will automatically scan new and existing apps as well as developer accounts, without disrupting the user experience of Android Market or requiring developers to go through an application approval process. Once an application is uploaded, the service immediately starts analyzing it for known malware, spyware and Trojans. It also looks for behaviors that indicate an application might be misbehaving, and compares it against previously analyzed apps to detect possible red flags. But, there still exist vulnerabilities despite the layer of bouncer, through which an application certified by the bouncer can attack the user's data confidentiality. Our system aims at handling the previously mentioned vulnerability up to a greater extent. Also, the check must be internally done for every new update installed, so that, there is no misuse of the access rights previously assigned to it. The user must initially accept that they have read the declarations and warnings even before the installation of updates. This is a measure to ensure that the user is informed about the details of the usage of the application and the properties that the application is trying to use in the user's device.

Keywords: Security; Vulnerability; Android; Applications

I. INTRODUCTION

Android is a modern and popular software platform for smartphones. It is an open source and Linux-based operating system. Among its predominant features is an advanced security model which is based on application-oriented mandatory access control and sandboxing. This allows developers and users to restrict the execution of an application to the privileges it has at the time of installation. The exploitation of vulnerabilities in program code is hence believed to be confined within the privilege boundaries of an application's sandbox. Concerning security and privacy aspects, Android deploys application sandboxing and a permission framework implemented as a reference monitor at the middleware layer to control access to system resources and mediate application communication. The current Android business and usage model allows developers to upload arbitrary applications to the Android app market1 and involves the end-user in granting permissions to applications at install-time. The Android platform has been designed to allow the installation of potentially untrusted applications. This is different from the iPhone security model, where all applications need to be installed through the Apple store (unless the mobile phone is 'jail broke' to allow the installation from different sources) and are (supposedly) verified concerning their internal behaviour prior to their publication. Therefore, the Android platform implements security mechanisms on different layers: application sandboxing as a high-level concept makes use of file system access control as enforced by the Linux kernel and permissions granted upon installation time to – selectively – pass the boundaries of these sandboxes. Applications are also cryptographically signed, but these signatures only provide some level of auditing and no security-relevant validation procedures before publication. In the following, we will describe these security layers and their potential shortcomings in more detail [1].

The current Android business and usage model allows developers to upload arbitrary applications to the Android app market1 and involves the end-user in granting permissions to applications at install-time. This, however, opens attack surfaces for malicious applications to be installed on users' devices. Since its introduction, a variety of attacks have been reported on Android showing the deficiencies of its security framework. Android or any other smartphones make their way into the mobile market based on the various applications they can support. Considering this fact, another noticeable point is the effects and vulnerabilities these applications cause to the privacy of the users. In android based

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2017

mobiles, customization is of more importance. This feature of android compromises the security and the ability of an operating system to shield the access of certain applications from violating the basic access rights provided to them at the time of installation i.e., privilege escalation. Initially the vendor requests for the basic permissions from the user which may not be harmful for the device [2]. This is checked by the Google bouncer application once any application is introduced into the Google play store. But once the application is installed, the vendor may ask for updates with the upgraded features. These updates at times do not contain only the upgraded features but may also contain some malicious information which may harm the user's device internally.
Furthermore, the application, once updated, may not make use of the same rights it was granted at the beginning. This imposes the serious urge to monitor the permissions that are being granted to the application even at the time of updating.

II. EXISTING SYSTEM

The current android system has a lot of user friendly features, thus making it more vulnerable to attacks. It doesn't provide security towards the malicious data entering it from various sources. This could be explained with the Google's Bouncer application. This application surely checks the access rights of every application in the play store, but once the application is downloaded, the manufacturer can modify the access rights it previously requested for using the updates. There is no restriction on the access of user's data [3-5]. The user is, in fact, unaware of the background functionality of the application. For example, the present system, a simple calculator application does not need any of the user's personal data. But, through the updates, if the application is internally accessing any of the data is unknown to the user. In the existing system, 3rd parties will likely have mechanisms to execute code on the device and read information from the device. For example if Apple wants to delete an application from your IOS device they can do that at any time. On Android you'll see applications provided by handset vendors which can't be removed (without removing the OS and replacing with a 3rd party ROM) and may be auto-updated, essentially giving them a backdoor onto the device.

III. PROPOSED WORK

If vulnerability is discovered in the OS users are totally dependent on the vendor for updates. This is made more of a problem for corporate users by the fact that some vendors do not publish good information on whether a given phone model is vulnerable to a specific issue and also do not have regular security patching cycles as you see with traditional PC platforms. Our system aims at handling the previously mentioned vulnerability up to a greater extent. Also, the check must be internally done for every new update installed, so that, there is no misuse of the access rights previously assigned to it. The user must initially accept that they have read the declarations and warnings even before the installation of updates. This is a measure to ensure that the user is informed about the details of the usage of the application and the properties that the application is trying to use in the user's device [6-8].

IV. PROPOSED MODEL

In the Figure 1 we can see the how the customer's device and the applications are related. Also how the developed scans the general application [9].

STEP 1: Customer's device searches for an application.
STEP 2: Application checks for access.
STEP 3: The access permissions are sent to the application.
STEP 4: The application return the status to the device.
STEP 5: Then the application is installed.
STEP 6: After installation, database is created.
STEP 7: Now the developed application scans the general app.
STEP 8: Also the developed app maintains a Database.
STEP 9: Application gives an update to the device.
STEP 10: Then the developed application creates a temporary list.
STEP 11: The developed app compares the permissions.
STEP 12: Finally after comparisons, the application return the status to the customer's device.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2017

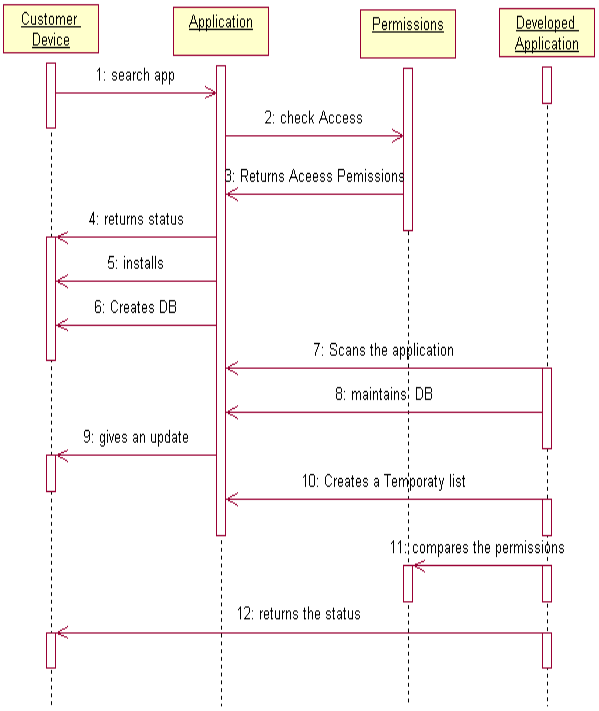

Figure 1: Experimental setup.

V. IMPLEMENTATION

We are implementing using java and running it on an Android 4.4, i.e., KITKAT version with 1.2 GHz of processor and 768 MB ram [10]. The server side script is written in java and database creator and connector used is SQLite. The current system focuses on three main modules.

5.1 Study of the Current Android Architecture

The first module of the project is to thoroughly study the architecture of android. This module gives us an overview of the limitations and benefits of the android base in any smartphone. It also explains the reasons why android phones are preferred to other phones with different operating systems [11].

5.2 Study of the Loops Holes of the Existing System

The loop holes of the android system enable the third party users to exploit the user's device even without the user's consent. Studying these loop holes or vulnerabilities helps to shield the system from external attacks [12].

5.3 Intimating the User about any Escalations

Performing a continuous or periodic check of the permissions accessed by each application. Also, comparing the permissions each time the updating is performed with the initial list of rights granted at the time of installation of the application. Warn the user if there are any escalations observed [13].

VI. CONCLUSION AND FUTURE WORK

In the current system, the application only checks for the access permissions while the application is being installed for the first time and also each time the application is updated. But if the application finds out that a certain app is escalating its permissions, it does not have the liberty to go ahead and block it, nor restrict the accessibility. The application developed can only warn the user about such attack, which indicates its limitation [14].

Future enhancement of the application would contain an opening for the user to control the permissions accessed by the applications each time an application is installed and updated.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2017

VII. ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my project guide Prof. A Rama Chandra Reddy who has in the literal sense, guided and supervised me. I am indebted with a deep sense of gratitude for the constant inspiration and valuable guidance throughout the work. Lastly, I extend my heartfelt thanks to my parents and family for their in time kind support and helping me in doing project.

VIII. REFERENCES

1. F Rafael, B Christian, et al. Android OS Security: Risks and Limitations: A Practical Evaluation. Fraunhofer Research Institution for Applied and Integrated Security 2012; 1-34.
2. http://developer.android.com
3. D Brian, Android Root Access Vulnerability Affecting Most Devices. Threat Post 2014.
4. http://www.androidvulnerabilities.org/
5. X Zhi, Android Installer Hijacking Vulnerability Could Expose Android Users to Malware. Paloalto networks 2015.
6. E William, O Machigar, et al. On lightweight mobile phone application certification. Computer and communications security 2009; 235-245.
7. J Markus, J Karl-Anders, Retroactive detection of malware with applications to mobile platforms. Security 2010.
8. https://www.mylookout.com/
9. W Lei, G Michael, et al. The impact of vendor customizations on android security. Computer and communications security 2013; 623-634.
10. D Lucas, D Alexandra, et al. Privilege Escalation Attacks on Android. International Conference on Information Security 2010; 346-360.
11. http://www.satprnews.com/2015/05/19/worlds-first-android-app-for-installer-hijacking-vulnerability-installer-hijacking-defenderlau nched/
12. B Alexandre, K Jacques, et al. Automatically securing permission-based software by reducing the attack surface: an application to Android. International Conference on Automated Software Engineering 2012; 274-277.
13. C Patrick, CK Lucas, et al. A privilege escalation vulnerability checking system for android applications. IEEE Communication Technology 2011.
14. QD Xiang, Y Ge, Coordinated attack research between Android applications and solutions. IEEE Software Engineering and Service Science 2014.