Android Permissions

Author:
William Richard
wricha04@tufts.edu

Supervisor:
Ming Chow

May 11, 2012

Contents

Abstract

The Android OS has a unique system by which it limits the abilities of applications. Rather than denying or allowing access to resources via the Linux file permission system, it creates it's own permission system which is more transparent to the user. An application is able to request usage of a certain feature or ability, and this request is presented to the user when the application is installed. Sadly, like every other permission system, it is not foolproof; there are several documented vulnerabilities to this system.

Here we explore much of the security related work done regarding the Android permission system. This includes what sorts of features are can be accessed without permissions that should be hidden, how applications can work together to subvert the permission system, how permissions are used in the real world, and the problem of overprivledged applications.

1 Introduction

The Android operating system, used in mobile devices, is one of the most popular operating systems currently on the market - in quarter 3 of 2011, Android was estimated be powering just over 50% of smartphones sold worldwide [2]. With this in mind, the security features of this operating system are becoming incresingly relevant, as users rely on them more and attackers attempt to break them. The main method by which Android controls applications access to users' devices is the permissions system.

1.1 Purpose of Permissions

In order for an application to use certain phone features or access certain data sources on a device, the application must explicitly request access. These requests are specified before the application is compiled. All of the permission requests of an application are displayed to the user before the application is installed. At this point, the user is supposed to read the permissions requested, and decide if they are willing to grant access to the application of the various features, and abort the installation if the user is uncomfortable with the permissions that the application is requesting. For example, in order to access the phones location, an application must request the ACCESS_FINE_LOCATION permission. [3][4]

1.2 How Permissions Work

When an application attempts to access a system resource that requires a permission, it calls a function in a public API for that feature. Within this public API, the applications permissions are verified - if an application has not requested the necessary permission, an exception is thrown. There is not standardized procedures dictating where and by which way this permission check is performed, and in several cases it is checked multiple times. If the application passes the permission checks, the public API then calls on a private interface, which makes the necessary request of the background system process or service. Note that no permission check occurs in the background system process or service - it is technically possible for an application to directly query the processes, circumventing the permission system, though no evidence of this occurring has been found.

Some system services are also protected via the UNIX file permission system, such as accessing the Internet, writing to the SD card and access to Bluetooth. In this case, when the application is installed, it is added to the relevant Linux group. Thus, in these limited cases the Linux kernel is directly controlling access to these resources. [1]

2 To The Community

This paper is aimed primarily at developers, but also users, of Android applications. Research has shown that developers do not fully understand the

permission system, and often request unnecessary permissions [1]. Also, in revealing some of the flaws in the permission system, this paper may inform developers on how to develop their own applications safely.

The Android permission system is very informative to users, but if the user does not understand what they are enabling, the permission system is meaningless and ineffective. Hopefully users will be able to gain a better understand of how the permission system works, and what sorts of permissions are reasonable to allow for a given application.

3 Action Items

The following are a small selection of problems with the permission system. They include the limitations of permissions, ways to get around the system, how permissions are used nefariously in the real world, and the problem of uninformed developers mis-requesting permissions.

3.1 Zero-permission Application

How effective are Android permissions? What can an application access if it does not have the necessary permission? Paul Brodeur has created an application that does not request any permissions, but is still able to access some revealing information and send it back to a central repository.

Firstly, all android applications are able to access external storage i.e. the SD card and read all non-hidden files. Much security-relevant information is

stored on the SD card - from photos to backups, to whatever files the user wants to store there. Being able to read all of these files can be a huge security vulnerability, depending on which applications are installed, and thus what is stored stored there.

Secondly, his application was able to read which other applications are installed on the device. This information is stored in a central location, which is readable to all applications. From there, his application is able to read any non-hidden files in any of the application-specific directories. This can potentially be used to steal information from other applications (such as password) or find weakly permissioned applications.

Finally, his application was able to read some identity information about the device. The GSM and SIM vendor IDs can be read, along with the kernel version and potentially the name of the installed ROM. Most dire, a zero permission app is able to read the Android ID, which is a random generated number that stays constant throughout the devices lifetime.

Though the app does not have the INTERNET permission, and thus cannot explicitly make Internet connections, it is able to open a browser. His app exports any found data by sending a specific GET request to his server, with all of the pilfered data included. [5]

3.2 Permission Spreading

There are many methods by which Android applications can, legitimately, communicate with other applications. For instance, applications can start

background services that constantly sitting in the background, waiting for a message from another application. Both the ability to start a background service and sending or recieving arbitrary messages are allowed by default to all applications.

Using such a background service, a group from Austria lead by Clemens Orthacker demonstated a method by which two applications can work together to gain the permissions of both rather than the permissions of one or the other [6]. They called this "Permission Spreading".

Essentially, you have one application that requests a set of permissions, and by request, provides the information provided by those permissions to another application. You then have another application that accesses that data source, and in effect is given more permissions than they requested. Since there is no centralized method by which permissions are verified, nor any control over what an applicaion does with that information once it is obtained, this is trivially easy to do, and almost impossible to detect without intelligent, in depth analysis of the code

3.3 How Are Permissions Being (Mis)Used

Enck et al developed an Android decompilier, which they used to decompile and statically analyze 1,100 free applications from the Android market. They found numerous instances were applications were misusing the permissions granted to them. [7]

The first major security-breaking usage of a permission they found re-

garded the phone identifiers, which as we discussed earlier, you can access without requesting the permission. Most app developers do request the permission, and use the phone id numbers to track the phone and individual users across phones (if they get a new phone, some of the id numbers stay the same). In almost all cases, these identification numbers are sent off of the phone, in many cases to advertisers.

Location information is also collected often (in about 50% of the apps analyzed), but in most cases it is used legitimately. If it is collected and sent off the phone, it is most often to advertisers. Many advertiser libraries will probe for location and phone id permissions, calling the functions to see if the parent app has the permission. Since these libraries piggyback on legitimate permission usage, it is often not apparent to the user that their information is being shared in these cases.

For the most part, it does not appear that telephony, audio or video recording permissions are being misused. In other words, applications do not seem to be sending texts, calling phone numbers, or recording the user maliciously.

The real security risks often come from advertisement libraries and developer toolkits that are included with applications. These libraries will often probe for information, as mentioned earlier, try to send it off the phone. Developer toolkits also often reimplement dangerous functionality, creating possible vulnerabilities.

Essentially, as far as permissions go, often applications will request phone

identity permissions and occasionally location permissions for user tracking. Otherwise, permission requests tend to be legitimate.

3.4 Overpermission

Overpermission is the situation where an application requests permissions it does not need. This is inadvisable because of the principle of least privilege, along with the fact that it confuses users. It also makes more applications susceptible to bugs or exploits unnecessarily. Adrienne Felt et al developed a tool that uses the android decompiler mentioned previously [7] to find which API calls are made in an application, and compares them to the API calls enabled by the permissions the application requested [1]. They found that over a third of the applications they analyzed were overprivledged, often because of developer confusion or mistake. For example, there are two similarly name permissions, ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE, which allow applications to read the state of the network controller and wifi controller, respectively. Developers often request these permissions together, when only one is required.

4 Conclusion

The Android Permission system is well intentioned. It is trying to be transparent to the user, and inform them about the abilities of the applications that they are installing. That said, this system has several flaws - from in-

formation sources that can be accessed without the necessary permissions, to a lack of information flow control allowing applications to work together to circumvent the system, to applications misusing legitimately requested permissions.

There are two main issues that stop this security feature from working effectively. Firstly, it relies upon an informed user. If the user does not understand what an application can do with the Bluetooth permission, they don't know if it's reasonable to grant it.

Furthermore, once you include multiple permissions, it is not the individual permissions that matter but their interaction that dictates the application's abilities. Once you start discussing the interaction of permissions, the intent of an application can become much more vague. For example, if an application that requests the fine GPS location permission and access to the Internet might be checking your local weather, or it might be tracking your phone, or both. With the current system, the user may know that these are both possibilities, but there is no way to tell which way the application is using their information.

The problem of overpermission only confounds this issue, because it adds extra information that further confuses and complicates an application's intention.

Ultimately, the Android permission system is well intentioned, but not detailed enough, and relies too heavily on a knowledge base that most users do not possess.

References

[1] Felt, Adrienne Porter et al. *Android Permissions Demystified*. UC Berkley, Berkley, CA. 2011.

[2] Pettey, Christy. *Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent*. https://www.gartner.com/it/page.jsp?id=1848514 Egham, UK. Nov 2011.

[3] *Security and Permissions* https://developer.android.com/guide/topics/security/security.html May 2012.

[4] *Android Security Overview* http://source.android.com/tech/security/index.html Apr 2012.

[5] Brodeur, Paul. *Zero-Permission Android Applications* http://leviathansecurity.com/blog/archives/17-Zero-Permission-Android-Applications.html Apr 2012.

[6] Orthacker, Clemens et al. *Android Security Permissions - Can we trust them?* University of Technology Graz, Austria 2011

[7] Enck, William et al. *A Study of Android Application Security* Pennsylvania State University, 2010