Institute of Computer Science 4
universitätbonn Communication and Distributed Systems

Rheinische
Friedrich-Wilhelms-Universität Bonn,
Germany

Lab Course: Selected Topics in Communication Management

# Android Security - Common attack vectors

**Author:** Patrick Schulz
**Advisor:** Daniel Plohmann
**Seminar:** Selected Topics in Communication Management
**Semester:** Winter term 2011/12
**Date:** August 11, 2012

# Contents

Android [1] is an operating system for mobile devices and runs on more than half of the smartphones sold worldwide [2]. These devices are used for nearly every kind of communication and store therefore a lot of personal and sensitive data like contacts, emails, credentials for online services, pictures, and chat logs. Finally, there are at least three aspects which make those devices a valuable target for attacks. First, those devices are almost always online and in the presence of their owners. Second, they store notable amounts of personal data. Third, they have enough computational power to perform almost all of the tasks accomplished by current desktop malware.
This paper describes the general Android security mechanisms and shows which common attack vectors, known from the desktop computer world, are applicable and which are prevented by the security concept.

# 1 Introduction

Smartphone sales are rising rapidly, with the Android [1] platform developed by the "Android Open Source Project" running on more than half of the worldwide sold smartphones [2]. A reason for this popularity is the wide range of use cases for smartphones and the possibility to be always online in order to almost instantly communicate with friends, business partner or browsing the web for information. Besides this, smartphones are used to organize personal data like contacts, email, pictures and so on. Due to the fact, that they are always online it is common to store also credentials for online services, such as social networks or even financial data on them. Therefore, smartphones are an interesting and valuable target for attacks, which are known in the computer world since years.

The Android platform runs with good performance on today's smartphones, which are as powerful as computer some years ago. The platform is based on Linux [3], a widely-used kernel for server and desktop systems. However, the main differences between Android smartphones and desktop systems are the user interface and the underlaying security concept. On the one hand, the user interface is smarter and easier to handle with a responsive touchscreen than normal desktop interfaces. On the other hand, this interface does not expose direct access to deep and detailed aspects of the system like the process management. The system wide security concept is much stricter than on desktop systems in order to ensure integrity and data security of all system components and applications running on the smartphone. This concept isolates different applications and restricts their ability to access data as well as introduces a different permission management. Unlike with common desktop operating systems, the owner of the smartphone has usually no administrative rights within the system. Besides these differences, the Android platform is designed in a way, that it is easy to install new applications through application markets.

The popularity and the similarity to desktop system raise the question on how well common attacks from the desktop world are opposed by the security concept of the Android platform and which new attacks are possible.

This paper is organized as follows. In section 2 we discuss related papers about attacks and security concepts. Section 3 describes the Android architecture and its security model. Attack vectors and their impact on Android smartphones are discussed and evaluated in section 4. In the final section we conclude and summarize the results.

## 2 Related Work

In this section we present related work on this topic. First, general surveys on the Android security concept itself are presented. This is followed by analyses of benign and malicious third party applications.

The Android platform and its security concept have been covered thoroughly by researchers. The implemented concept has been evaluated and compared with other concepts. A good description of the basic concept is given in "Google Android: A State-of-the-Art Review of Security Mechanisms" [4] by Shabtai et al. Besides different approaches and mechanisms which shall ensure the security of the system, the authors analyzed different attack vectors e.g. browser exploitation and SQL-injection and clustered them along the characteristics "Likelihood of occurrence" and "impact" of this threat. The security concept has been addressed in other survey papers as well, focusing on security flaws like trusted USB connections and long patch cycles. In the paper "All Your Droid Are Belong To Us: A Survey of Current Android Attacks" [5] they categorized attacks by the need of remote and physical access as well as privileged and unprivileged access.

Enck et al. [6] investigated Android specific vulnerabilities within applications. They worked out methods which can be used to circumvent security mechanisms and to abuse the system services.

Malware targeting the Android platform has been covered with analyses and case studies as well [7][8]. A discussion about the evolution of Android malware in combination with their distribution techniques shows that the security concept has influenced the techniques used by malware in comparison to other architectures [7]. On the one hand the research shows also there are still some security flaws within the concept which are abused by malware, but on the other hand we can deduct from this result best practices that can reduce the effects of malware on the Android platform [8].

# 3 Android Security Concept

The Android platform is the most popular platform for mobile devices [2]. It is designed in a way, that applications can be easily installed through online application markets. In order to secure such a system, Android implements a security concept which protects the system as well as the applications by isolating the application context and permission management.
In the following subsection we discuss the Android architecture. Based on this, we describe in subsection 3.2 the security features and extensions within the underlaying kernel. Followed by the sandboxing concept in subsection 3.3 and the design of the Application Framework in subsection 3.4, which provides the runtime environment for Android applications.

## 3.1 Architecture

In this section we describe the main components of the Android architecture. It is based on a Linux Kernel and adopts many concepts, which are common on unix systems. So, Android uses different system users and applies file system permissions in order to isolate applications. In order to allow communication and interaction beyond this isolation, Android provides interfaces, which are protected by a permission management.

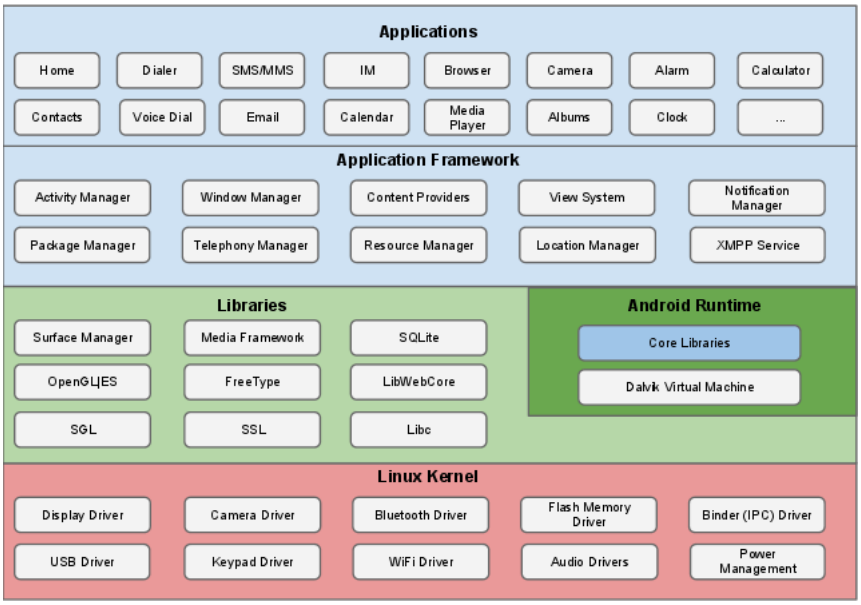


Figure 1: The Android software stack is based on five main component groups [9]

The Android architecture consists of five main component groups, as shown in figure 1. The underlaying Linux kernel manages any hardware access and provides interfaces for device specific drivers, which were distributed by hardware vendors. The kernel configuration on Android has been adjusted for special proposes of mobile devices. Therefore, many features, which are enabled on desktop and server systems, have been disabled in order to reduce memory and energy consumption.
The second component group contains native libraries, which provide different services

e.g. for graphic rendering, cryptographic functions and database access. These components are executed natively and can be used within Android applications by loading them into their process context. So, these components are stored as shared objects. Such a process of an application also contains a copy of the Dalvik Virtual Machine (DVM), as part of the third component group shown in figure 1. The DVM executes Android applications, which are delivered in form of Dalvik bytecode. Within the DVM, an application can make use of services, which are provided by the Android Application Framework and native libraries. By this, the Android Application Framework can ensure compatibility between different systems and restrict data access by permission checking, as we will discuss in section 3.4.

## 3.2 Kernel

The Android system uses a Linux kernel and in consequence the security concept of Android is build on top of that. Therefore, the kernel has to be hardened, which is done by disabling unused features in order to reduce attackable code surface and to minimize the power consumption. But also security relevant features like the SELinux extension have been disabled, because of their complexity and power consumption. Besides this, the kernel has also been extended by an interprocess communication manager called "Binder (IPC) Driver", which is shown in figure 1. This extension is necessary in order to secure communications between different processes. The Binder can be used by a process to send messages to other processes. While processing these messages, the Binder performs permission checks and ensures the desired process isolation level. Permission checks will also be performed in case of hardware access. Therefore, applications must hold the particular permission in order to access the hardware e.g. requesting GPS information. This check is implemented as system group memberships, so the particular application must be within a specific system group. The kernel has been modified in order to implement these checks. This also includes some hard coded group assignments within the code, which is very unusual for Linux systems. Another important feature of the kernel comes with the multiuser paradigm. This means that applications, which run under different users, are not able to access resource of the other application and vice versa [9]. This isolation scheme is directly used to implement the application Sandbox, which will be discussed in the next section.

## 3.3 Sandbox

Every application on the Android platform runs within its own sandbox by default. By this, the application is isolated from other applications except well-defined and system supervised interfaces like the IPC, as discussed in section 3.2. In order to achieve this, the multiuser paradigm of the kernel is used by assigning individual user accounts to the installed applications. As a result, every application is executed as a different user and therefore its resources are protected and only accessible by the owning application itself. Data and files, which are stored by an application are also protected by default file system permissions in a way, that an application must explicitly grant permissions in order to allow other applications to get access to this data.
This approach is well-known and used within unix server installations for years.

## 3.4 Application Framework

The Application Framework of the Android platform provides services, like "Windows Manager" and "Content Provider" as shown in figure 1, to applications in order to have a uniform interface. This abstraction layer, implemented by these services, is necessary in order to control the access to those information. This is achieved by permission checks within the interprocess communication by the Binder service, as discussed in section 3.2. The evaluation of this checks is based on the properties of the particular application. At installation time of an application a list of permissions, which are needed to run this application, is shown to the user. In this situation, a user can decide to grant these permissions to this application or to abort the installation. In this way, the system knowns what a particular application is allowed to do and which information should be accessible, because of the permission assignment. This approach ensures that the user knows about the capabilities of an application.
Besides this, the Application Framework also takes care about the installation process of new applications. This is done in order to control the permission assignment process and to do some further verification and optimization of the application itself. This includes the assignment of a unique system user, so that the sandbox mechanism as discussed in section 3.3 can be applied.

To conclude, the Android security concept ensures application isolation and takes control over interprocess communication between different applications as well as communication with other services. This is achieved by a combination of different mechanisms, as discussed in the last sections. Furthermore, the concept ensures that the user gets information about what an application is capable of.

# 4 Attack Vectors

Attack vectors are paths through a security concept that can be used in order to elevate permission levels, gain access to the system or data, which are stored on the system. In this section we evaluate the security concept of the Android platform against common attack vectors, known from desktop systems, and against new vectors, which are specific to mobile systems.

Attacks on Android devices are motivated by different attack goals. Android devices have significant value to serve as bot in a botnet due to the fact that these devices are always on and most of the time connected to the internet. This is one of the most important threats for today's computer systems and has also been found on Android [10]. Another goal could be to steal credentials for online services or private data like emails, contacts and pictures. Compared to desktop systems it is easier to gather and find such information due to the fact that the platform provides functionalities to access them and that in most cases it is well known where these information are stored. Furthermore, it is very likely that these information are up to date and therefore more valuable, because these devices are designed and mainly used to communicate which e.g. needs up to date contact information. Another new attack goal, that comes with smartphones, is to capture mTans, which are send via the short messages service (SMS). These mTans are used by online banking systems, when the user initiate a new bank transaction. Then an SMS with a mTan is send in order to have a two-way authentication. So an attacker is interested in capturing this mTans.

In the following sections we discuss different attack vectors and evaluate them against the Android security concept.

## 4.1 Social Engineering

Social Engineering does not use technical aspects to attack a system in the first place, but instead targets the user and his capabilities of making decisions. This method tries to manipulate the user in a way, that he helps the attacker to perform his attack. This e.g. can be done by faking information, so that the user assumes the attacker is some authorized person like an administrator. Another example would be to prepare a website, that looks like the website of an manufacturer, and send a link to this site to an user with the message that a new update is available.

This kind of attacks are well-known under desktop systems for years. As a famous example, the ILOVEYOU virus e.g. was send by email and tricked the user by suggesting to be a love letter, so that the receiver opens and executes it by himself.
In order to transfer this scheme to Android, we have to send our malicious application or an information about how to get it to the user, so that he we install and executes it. This can be done via email with an attachment or by sending a link to a website via some communication channel like email, SMS, twitter and so on. This is the same on desktop systems. But, due to the security concept of Android the installation process is different on Android. Whereas on the desktop system the user is not able to get information about

what the downloaded application will do or which data it is able to access, he has to rely on anti malware software or similar approaches. This is different on Android, before an application gets installed, the user is informed about what permissions this application is using. This enables the user to get an idea about what this application is going to do and can compare this with what he though the application should do. So e.g. if he downloads an application for painting, but this one wants to have access to the internet and the contacts, the user is able to abort the installation because this looks suspicious. Finally, the security concept, which includes to inform the user about an application when it gets installed, helps the user to detect such attacks easier, but does not prevent all of them. Another advantage compared to desktop systems is that the user get notified about that an application is going to be installed, which is not the case on common desktop systems. By this the user knowns about all installed applications.

## 4.2 Drive-by Exploitation

Drive-by Exploitation is an attack method that uses bugs within software, running on the device and processing external data. In most cases this attack vector is done using bugs within the web browser. While a user is surfing the web, the web browser gets data from server, processes this data, and displays it, so that the user can see the website. In case of a bug within the processing of this data, the attacker is able to execute code, also called payload, on the device. These bugs could be e.g. buffer or heap overflows.

This kind of attacks have a big impact on the security of a system, because the attacker can execute malicious code without users knowledge. They have been seen a lot on desktop systems and therefore especially browser software have been hardened by further security mechanisms like sandboxing and memory protection mechanisms. But still, there are many bugs within browser and also contests are organized where the attendees have to find bugs and exploit them [11].
On the Android platform the situation is a little bit different, due to the fact that on desktop systems most of the software is implemented as native executable, which are prone to such bugs. On Android most of the software is implemented in Java and therefore executed within the Dalvik Virtual Machine (DVM). So, the combination of the Java language, which protects data structures by boundary checks, and the DVM protects against the exploitation of such bugs. However, also on the Android platform applications can use native libraries, as discussed in section 3.1, which are implemented as native code. So, these parts of an application are still vulnerable for such attacks. This e.g. apply to the Android default browser, which is distributed within every Android device. A proof of concept attack has been shown at the RSA conference [12].
On the one hand, the attack space for Drive-by Exploitation has been reduced, by using Java and DVM as a base for the major part of Android applications. On the other hand, some parts are implemented as native libraries and so they become vulnerable for such memory management bugs. Furthermore, a huge amount of Android devices are still running under older versions [13] and wound get updates in order to eliminate well-known bugs.

## 4.3 Phishing

A Phishing attack is used in order to gather information, especially credentials from a user by masquerading itself as an official instance. This can be done by sending faked emails or serving similar looking website e.g. for an online banking system. This kind of attack includes aspects of social engineering, as discussed in section 4.1. Due to this, there are no good technical solution for this kind of attacks and so they are still common on desktop system. On the Android platform this also holds. But the situation here is different, because for most online services, which are used on mobile devices, there are also applications available to have a better user experience while using the service. Therefore, most users do not use the website directly, but use an application for this service. So in this case the attack vector can not be applied that easy because the application can not be tricked as easy as the user. Finally, for other use cases this attack vector is still applicable.

## 4.4 Network Services

Network services are provided by programs running on the local system. They are used to communicate with other systems. In order to achieve this they listen on a network interface, receiving packets and process them. An example for typical services running on most computer are Samba and NFS, two network file system services. In this case all involved system have to run such a program.

Network services are always in the risk of being attacked due to their nature that they are reachable from the network. In most cases it is not necessary to interact with the user. If these services are vulnerable they can be attacked remotely. On desktop systems it is very usual to run such services especially network file systems and printing services. For smartphones this is not the case. At the default configuration there are no network services running or installed. So we have a reduced attack vector space on these devices.

## 4.5 WebView

WebView is a new technique, specific to mobile devices, that enables applications to integrate a browser component and to have a well defined communication interface between the displayed website and the application. The WebView feature comes with the WebKit framework and has been integrated into Android, because many applications want to integrate and display the content of a particular website, like e.g. social networks. Furthermore, these applications want to enrich the displayed content with locally available content like contacts. In this case, WebView can be used, which provides an interface so that a JavaScript component, integrated into the website, and a handler function within the application can communicate and exchange such data. Finally, this website can access data stored on the Android devices, if the application uses WebView. The same holds for other mobile platforms like iOS, where this technique is called UIWebView [14].

The attack vector can be described as the following. The user of an Android device is using an application, which integrates a WebView, in order to use an online service like a instant messenger which uses phone numbers to identify the communication partner.

So, this service must have access to the locally stored contacts, which can be done using the WebView feature. So this service, which is implemented as a website, can access the contacts using JavaScript. In the case of an attacker can modify the website and integrate further JavaScript into it, which e.g. can be done using cross site scripting (XSS), he is able to use these functionalities of WebView to get access to contacts.
This attack vector is specific to mobile platforms and is not possible on desktop system.

## 4.6 Android Market

The Android Market is an online service by Google Inc. which can be used to easily download and install new applications to the Android device. Due to the fact that this service only provides applications for the Android platform, attacks using this service are specific to Android and not applicable to desktop systems. Also on desktop systems it is unlikely to have a central software repository like on mobile devices.

The Android Market is an easy way to install new applications and also to distribute own applications, by uploading them to the Market. This can also be done by maleware authors. So this service has been massively used to distribute maleware and therefore Google Inc. tries to reduce the amount of malicious applications within the marked by first checking against some malware databases before distributing these applications. This checking service started in February 2012 and is called Bouncer.
Most of the distributed Android malware samples are repacked applications with some malicious code injected. This can easily be done by disassembling the application and modifying the resulting program code. After the malicious code has been injected the application is repacked and uploaded to the market. At this point the market contains the original as well as the modified version of the application and the user, which is going to install it, has to figure out which of these two applications is the original. Due to the fact that the modified version looks the same and also has the same description within the market, it is very hard to see any differences. The only way for the user to detect this, besides downloading both and comparing their code by hand which is not applicable for most users, is to rely on comments for other market users.
Finally, on Android malware can be uploaded and distributed via the Android market, which is used by maleware authors. Therefore, the user has to decide before installing a new application if this is a malicious one which can be hard.

## 4.7 Physical Attacks

This attack vector is about the security consequences in the case where an attacker has physical access to the device. We exclude attacks like opening the device, but focus on privilege escalation attacks. Many desktop systems are equipped with a Firewire connector and their systems are not hardened against attacks abusing the DMA feature of Firewire. By having physical access to a system with firewire enabled we have random read and write access and therefore can control the system. On Android this is not possible because these devices are not equipped with Firewire or other techniques which provides DMA.

In Android there is a feature called Android Debug Bridge (ADB) which is used within the development process of applications for this platform. ADB provides functionalities which can circumvent some aspects of the Android security concept. It is possible to install Android applications without prompting the user asking for permission assignment by using ADB over an USB connection. In order to do this the ADB feature must be activated which is not the case by default. But, if the user actives it the device is vulnerable for such attacks. This attack is limited to the Android platform and thus not applicable to desktop systems.

# 5 Conclusion

The Android platform for mobile devices has implemented a new security architecture in order to fulfill new requirements which come with the use cases of mobile devices. The security architecture combines many successfully concepts from unix server systems. They have been adapted in order to improve the security of the system, protect users data, and let the system still be usable for less security-aware users.
The new security architecture of Android minimizes the effect of a lot of common attacks, which are well known and still in use under desktop systems. From this we can conclude that the architecture is more effective in the sense of the system security. Furthermore, the system is still usable for users without knowledge of the technical inner aspects, which is important for such an popular platform. The possibility to be more informed and have more control over installed software by the user is a further improvement compared to security architectures of common desktop systems.
On the one hand many well known attack vectors have been eliminated by the implemented architecture, but on the other hand new features have been implemented within the Android platform, which made new attack vectors possible.

We conclude that the Android system is more secure than desktop system relating to common attack vectors, but this might change when new attack vectors are exploited more often.

# References


[1] Android Open Source Project. Android sources. Visited: May, 2012. [Online]. Available: http://source.android.com

[2] Gartner. Worldwide smartphone sales soared in fourth quarter of 2011 with 47 percent growth. Visited: May, 2012. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1924314

[3] L. Torvalds. The linux kernel archives. Visited: June, 2012. [Online]. Available: http://www.kernel.org/

[4] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, and S. Dolev, "Google android: A state-of-the-art review of security mechanisms." CoRR, 2009.

[5] T. Vidas, D. Votipka, and N. Christin, "All your droid are belong to us: A survey of current android attacks." USENIX Association, 2011.

[6] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," 2011.

[7] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution." IEEE Symposium on Security and Privacy, 2012.

[8] V. Manjunath, "Reverse engineering of malware on android." SANS Institute, 2012.

[9] Google Inc. Android security overview. Visited: Jun, 2012. [Online]. Available: http://source.android.com/tech/security/index.html

[10] X. Jiang. New rootsmart android malware utilizes the gingerbreak root exploit. Visited: Jun, 2012. [Online]. Available: http://www.cs.ncsu.edu/faculty/jiang/RootSmart/

[11] The pwnie awards. Visited: Jun, 2012. [Online]. Available: http://pwnies.com

[12] D. Alperovitch and G. Kurtz. Rsac us 2012 – hacking exposed: Mobile rat edition. Visited: May, 2012. [Online]. Available: http://365.rsaconference.com/community/archive/usa/blog/2012/03/15/video-rsac-us-2012-hacking-exposed-mobile-rat-edition--dmitri-alperovitch-george-kurtz

[13] Google Inc. Current distribution. Visited: May, 2012. [Online]. Available: http://developer.android.com/about/dashboards/index.html

[14] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin, "Attacks on webview in the android system," 2011.