

Create apps that are safe from hacking,  
attacks, and security breaches



# Android Apps Security

Sheran A. Gunasekera



Apress®

# Android Apps Security



Sheran Gunasekera

Apress®

## Android Apps Security

Copyright © 2012 by Sheran Gunasekera

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN 978-1-4302-4062-4

ISBN 978-1-4302-4063-1 (eBook)

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The images of the Android Robot (01 / Android Robot) are reproduced from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License. Android and all Android and Google-based marks are trademarks or registered trademarks of Google, Inc., in the U.S. and other countries. Apress Media, L.L.C. is not affiliated with Google, Inc., and this book was written without endorsement from Google, Inc.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editor: Steve Anglin

Development Editor: Tom Welsh, Douglas Pundick

Technical Reviewer: Michael Thomas

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Brigid Duffy

Copy Editor: Jill Steinberg

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://www.springeronline.com).

For information on translations, please e-mail [rights@apress.com](mailto:rights@apress.com), or visit [www.apress.com](http://www.apress.com).

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at [www.apress.com/bulk-sales](http://www.apress.com/bulk-sales).

Any source code or other supplementary materials referenced by the author in this text is available to readers at [www.apress.com](http://www.apress.com). For detailed information about how to locate your book's source code, go to [www.apress.com/source-code/](http://www.apress.com/source-code/).

*For Tess and Shana*

*—Sheran*



# Contents at a Glance

**About the Author ..... xiii**

**About the Technical Reviewer ..... xv**

**Acknowledgments ..... xvii**

**■ Chapter 1: Android Architecture..... 1**

**■ Chapter 2: Information: The Foundation of an App..... 13**

**■ Chapter 3: Android Security Architecture ..... 31**

**■ Chapter 4: Concepts in Action – Part 1 ..... 47**

**■ Chapter 5: Data Storage and Cryptography..... 55**

**■ Chapter 6: Talking to Web Apps..... 87**

**■ Chapter 7: Security in the Enterprise ..... 121**

**■ Chapter 8: Concepts in Action: Part 2..... 137**

**■ Chapter 9: Publishing and Selling Your Apps ..... 163**

**■ Chapter 10: Malware and Spyware ..... 203**

**Appendix A: Android Permission Constants ..... 213**

**Index..... 223**



# Contents

**About the Author ..... xiii**

**About the Technical Reviewer ..... xv**

**Acknowledgments ..... xvii**

**■ Chapter 1: Android Architecture ..... 1**

    Components of the Android Architecture.....2

        The Kernel .....3

        The Libraries.....4

        The Dalvik Virtual Machine .....4

        The Application Framework.....5

        The Applications .....7

What This Book Is About.....7

Security .....8

    Protect Your User .....8

    Security Risks.....8

Android Security Architecture .....9

    Privilege Separation .....9

    Permissions .....10

    Application Code Signing.....11

Summary.....12

■ <b>Chapter 2: Information: The Foundation of an App</b>	<b>13</b>
Securing Your Application from Attacks	13
Indirect Attacks	13
Direct Attacks	15
Project 1: “Proxim” and Data Storage	15
Classification of Information	22
What Is Personal Information?	23
What Is Sensitive Information?	23
Analysis of Code	23
Where to Implement Encryption	24
Results of Encryption	26
Reworked Project 1	26
Exercise	28
Summary	29
■ <b>Chapter 3: Android Security Architecture</b>	<b>31</b>
Revisiting the System Architecture	31
Understanding the Permissions Architecture	33
Content Providers	34
Intents	38
Checking Permissions	38
Using Self-Defined Permissions	39
Protection Levels	40
Sample Code for Custom Permissions	41
Summary	44
■ <b>Chapter 4: Concepts in Action – Part 1</b>	<b>47</b>
The Proxim Application	47
Summary	54
■ <b>Chapter 5: Data Storage and Cryptography</b>	<b>55</b>
Public Key Infrastructure	56
Terms Used in Cryptography	59

---

Cryptography in Mobile Applications.....	59
Symmetric Key Algorithms .....	60
Key Generation .....	60
Data Padding .....	62
Modes of Operation for Block Ciphers .....	62
Data Storage in Android .....	66
Shared Preferences .....	67
Internal Storage .....	70
SQLite Databases .....	73
Combining Data Storage with Encryption .....	78
Summary .....	85
<b>Chapter 6: Talking to Web Apps.....</b>	<b>87</b>
Preparing Our Environment .....	88
HTML, Web Applications, and Web Services.....	94
Components in a Web Application .....	96
Web App Technology .....	98
OWASP and Web Attacks .....	104
Authentication Techniques .....	105
Self-Signed Certificates.....	110
Man-in-the-Middle Attack .....	111
OAuth .....	113
Challenge/Response with Cryptography.....	118
Summary .....	119
<b>Chapter 7: Security in the Enterprise .....</b>	<b>121</b>
Connectivity.....	121
Enterprise Applications .....	122
Mobile Middleware.....	123
Database Access .....	124
Data Representation .....	130
Summary .....	136



<b>Chapter 8: Concepts in Action: Part 2.....</b>	<b>137</b>
OAuth.....	137
Retrieving the Token .....	137
Handling Authorization .....	139
Challenge Response .....	149
Summary .....	162
<b>Chapter 9: Publishing and Selling Your Apps .....</b>	<b>163</b>
Developer Registration .....	163
Your Apps—Exposed.....	165
Available for Download.....	165
Reverse Engineering.....	169
Should You License? .....	174
Android License Verification Library.....	175
Download the Google API Add-On.....	180
Copy LVL Sources to a Separate Directory .....	182
Import LVL Source As a Library Project.....	182
Building and Including LVL in our app .....	188
Licensing Policy.....	194
Effective Use of LVL.....	196
Obfuscation .....	198
Summary .....	201
<b>Chapter 10: Malware and Spyware .....</b>	<b>203</b>
Four Stages of Malware .....	204
Infection.....	204
Compromise .....	204
Spread .....	204
Exfiltration .....	205
Case Study 1: Government Sanctioned Malware .....	205
Infection.....	206
Compromise .....	206

---

Spread .....	206
Exfiltration .....	206
Detection .....	207
Case Study 2: Retail Malware—FlexiSPY .....	209
Anti-Forensics .....	210
Summary .....	212
<b>Appendix A: Android Permission Constants .....</b>	<b>213</b>
Content Provider Classes .....	218
<b>Index.....</b>	<b>223</b>

---

# About the Author



**Sheran Gunasekera** is a security researcher and software developer with more than 13 years of information security experience. He is director of research and development for ZenConsult Pte. Ltd., where he oversees security research in both the personal computer and mobile device platforms. Sheran has been very active in BlackBerry and Mobile Java security research and was the author of the whitepaper that revealed the inner workings of the first corporate-sanctioned malware application deployed to its subscribers by the UAE telecommunications operator Etisalat. He has spoken at many security conferences in the Middle East, Europe and Asia Pacific regions and also provides training on malware analysis for mobile devices and secure

software development for both Web and mobile devices. He also writes articles and publishes research on his security-related blog, <http://chirashi.zenconsult.net>.

---

# About the Technical Reviewer



**Michael Thomas** has worked in software development for over 20 years as an individual contributor, team lead, program manager, and Vice President of Engineering. Michael has over 10 years experience working with mobile devices. His current focus is in the medical sector using mobile devices to accelerate information transfer between patients and health care providers.



# Acknowledgments

I'd like to thank the editors, reviewers, and staff at Apress who worked tirelessly to help get this book published. They were the driving force behind this book in more ways than one. They put up with more than they should have. I am not a model author.

I'd also like to thank my dear friends and colleagues, Michael Harrington and Shafik Punja, without whom I would not have had the opportunity to publish this book. Thanks guys, this has been a great experience.

—Sheran Gunasekera

# Android Architecture

Google entered the mobile phone market in a style that only multibillion-dollar companies can afford: it bought a company. In 2005, Google, Inc. purchased Android, Inc. At the time, Android was relatively unknown, despite having four very successful people as its creators. Founded by Andy Rubin, Rich Miner, Chris White, and Nick Sears in 2003, Android flew under the radar, developing an operating system for mobile phones. With a quest to develop a smarter mobile phone that was more aware of its owner's preferences, the team behind the Android operating system toiled away in secrecy. Admitting only that they were developing software for mobile phones, the team remained quiet about the true nature of the Android operating system until the acquisition in 2005.

With the full might of Google's resources behind it, Android development increased at a rapid pace. By the second quarter of 2011, Android had already captured nearly a 50% market share in mobile phone operating systems shipped to end users. The four founders stayed on after the acquisition, with Rubin taking the lead as Senior Vice President of Mobile. The official launch of version 1.0 of Android took place on September 23, 2008, and the first device to run it was the HTC Dream (see Figure 1-1).



**Figure 1-1.** An HTC Dream (Courtesy Michael Oryl)

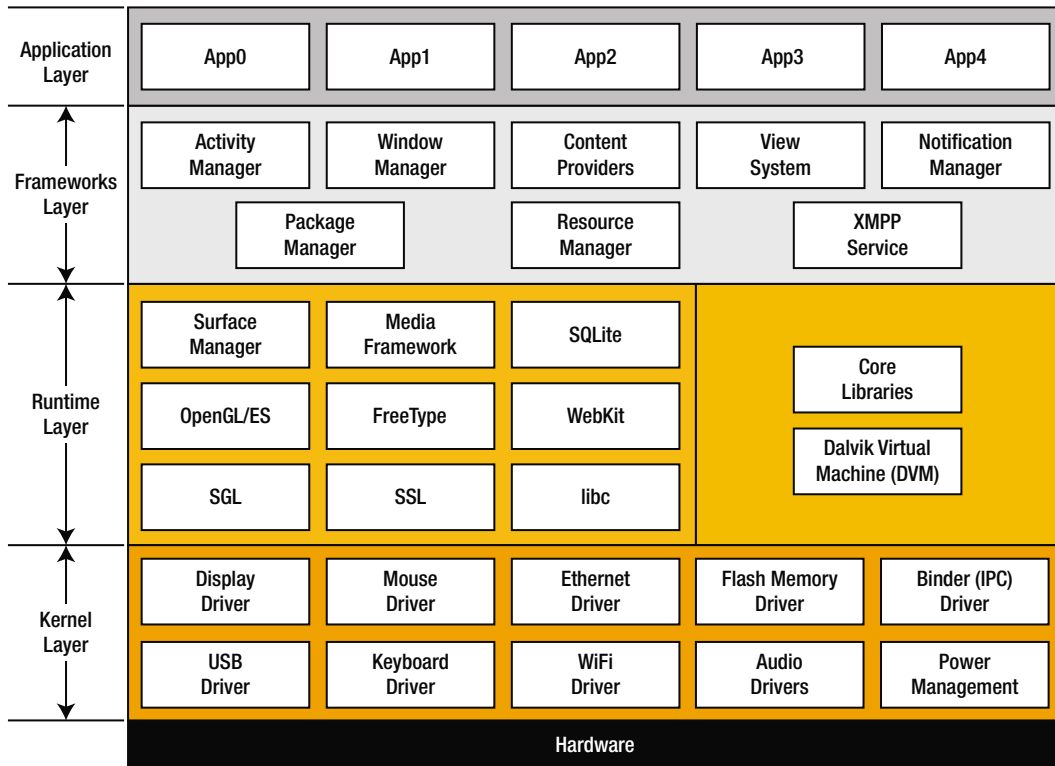
One of the unique features of the Android operating system that has allowed it to grow rapidly has been that the binaries and source code are released as open source software. You can download the entire source code of the Android operating system, and it takes up approximately 2.6 GB of disk space. In theory, this allows anyone to design and build a phone that runs Android. The idea of keeping the software open source was followed until version 3.0. Versions of Android including and higher than 3.0 are still closed source. In an interview given to *Bloomberg Businessweek*, Rubin said that the version 3.x code base took many shortcuts to ensure it was released to market quickly and worked with very specific hardware. If other hardware vendors adopted this version of Android, then the chances for a negative user experience would be a possibility, and Google wished to avoid this.<sup>1</sup>






## Components of the Android Architecture

The Android architecture is divided into the following four main components (see Figure 1-2):

1. The kernel
2. The libraries and Dalvik virtual machine
3. The application framework
4. The applications

<sup>1</sup> *Bloomberg Businessweek*, "Google Holds Honeycomb Tight," Ashlee Vance and Brad Stone, [www.businessweek.com/technology/content/mar2011/tc20110324\\_269784.htm](http://www.businessweek.com/technology/content/mar2011/tc20110324_269784.htm), March 24, 2011.



C, C++, Native Code	Java
 = Linux Kernel	 = Android Frameworks
 = Libraries	 = Applications
 = Android Runtime	

**Figure 1-2.** The Android architecture

## The Kernel

Android runs on top of a Linux 2.6 kernel. The kernel is the first layer of software that interacts with the device hardware. Similar to a desktop computer running Linux, the Android kernel will take care of power and memory management, device drivers, process management, networking, and security. The Android kernel is available at <http://android.git.kernel.org/>.

Modifying and building a new kernel is not something you will want to consider as an application developer. Generally, only hardware or device manufacturers will want to modify the kernel to ensure that the operating system works with their particular type of hardware.



## The Libraries

The libraries component also shares its space with the runtime component. The libraries component acts as a translation layer between the kernel and the application framework. The libraries are written in C/C++ but are exposed to developers through a Java API. Developers can use the Java application framework to access the underlying core C/C++ libraries. Some of the core libraries include the following:

- *LibWebCore*: Allows access to the web browser.
- *Media libraries*: Allows access to popular audio- and video-recording and playback functions.
- *Graphics libraries*: Allows access to 2D and 3D graphics drawing engines.

The runtime component consists of the Dalvik virtual machine that will interact with and run applications. The virtual machine is an important part of the Android operating system and executes system and third-party applications.

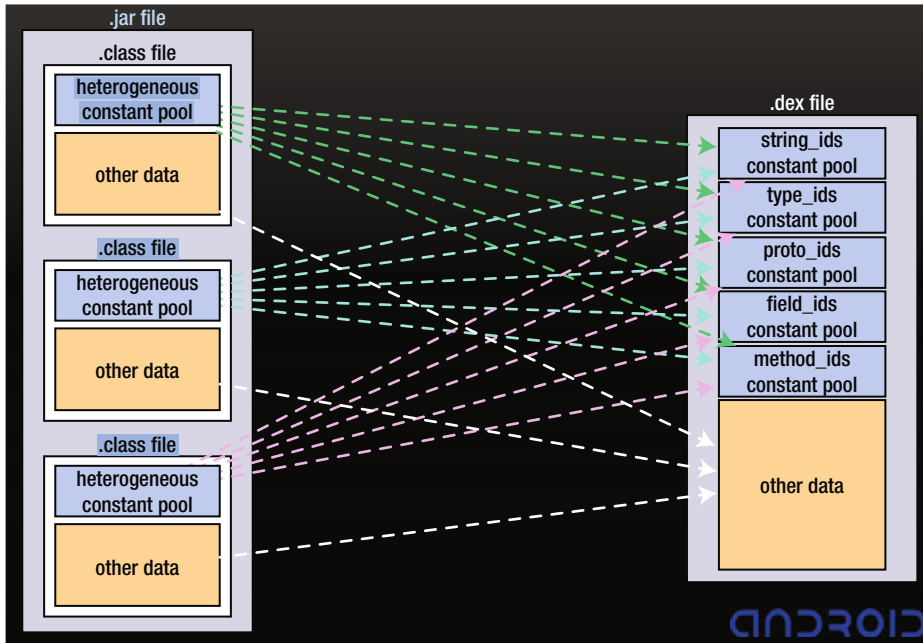
## The Dalvik Virtual Machine

Dan Bornstein originally wrote the Dalvik virtual machine. He named it after a small fishing village in Iceland where he believed one of his ancestors once originated. The Dalvik VM was written primarily to allow application execution on devices with very limited resources. Typically, mobile phones will fall into this category because they are limited by processing power, the amount of memory available, and a short battery life.

### WHAT IS A VIRTUAL MACHINE?

A virtual machine is an isolated, guest operating system running within another host operating system. A virtual machine will execute applications as if they were running on a physical machine. One of the main advantages of a virtual machine is portability. Regardless of the underlying hardware, the code that you write will work on the VM. To you as a developer, this means that you write your code only once and can execute it on any hardware platform that runs a compatible VM.

The Dalvik VM executes `.dex` files. A `.dex` file is made by taking the compiled Java `.class` or `.jar` files and consolidating all the constants and data within each `.class` file into a shared constant pool (see Figure 1-3). The `dx` tool, included in the Android SDK, performs this conversion. After conversion, `.dex` files have a significantly smaller file size, as shown in Table 1-1.



**Figure 1-3.** Conversion of a .jar file to a .dex file

**Table 1-1.** A File Size Comparison (in Bytes) of .jar and .dex Files

Application	Uncompressed .jar	Compressed .jar	Uncompressed .dex
Common system libraries	21445320 = 100%	10662048 = 50%	10311972 = 48%
Web browser app	470312 = 100%	232065 = 49%	209248 = 44%
Alarm clock app	119200 = 100%	61658 = 52%	53020 = 44%

## The Application Framework

The application framework is one of the building blocks for the final system or end-user applications. The framework provides a suite of services or systems that a developer will find useful when writing applications. Commonly referred to as the API (application programming interface) component, this framework will provide a developer with access to user interface components such as buttons and text boxes, common content providers so that apps may share data between them, a notification manager so that device owners can be alerted of events, and an activity manager for managing the lifecycle of applications.

As a developer, you will write code and use the APIs in the Java programming language. Listing 1-1, taken from Google's sample API demos (<http://developer.android.com/resources/samples/ApiDemos/index.html>), demonstrates how to use the application framework to play a video file. The `import` statements in bold allow access to the core C/C++ libraries through a Java API.

*Listing 1-1. A Video Player Demo (Courtesy Google, Inc.)*

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.apis.media;

import com.example.android.apis.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;

public class VideoViewDemo extends Activity {

    /**
     * TODO: Set the path variable to a streaming video URL or a local media
     * file path.
     */
    private String path = "";
    private VideoView mVideoView;

    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.videoview);
        mVideoView = (VideoView) findViewById(R.id.surface_view);

        if (path == "") {
            // Tell the user to provide a media file URL/path.
            Toast.makeText(
                VideoViewDemo.this,
                "Please edit VideoViewDemo Activity, and set path"
                    + " variable to your media file URL/path",
                Toast.LENGTH_LONG).show();
        } else {
```

```
/*
 * Alternatively, for streaming media you can use
 * mVideoView.setVideoURI(Uri.parse(URLstring));
 */
mVideoView.setVideoPath(path);
mVideoView.setMediaController(new MediaController(this));
mVideoView.requestFocus();
    }
}
}
```

## The Applications

The application component of the Android operating system is the closest to the end user. This is where the Contacts, Phone, Messaging, and Angry Birds apps live. As a developer, your finished product will execute in this space by using the API libraries and the Dalvik VM. In this book, we will extensively look at this component of the Android operating system.

Even though every component of the Android operating system can be modified, you will only have direct control over your own application's security. This does not, however, give you free rein to ignore what happens if the device is compromised with a kernel or VM exploit. Ensuring your application does not fall victim to an attack because of an unrelated exploit is also your responsibility.

## What This Book Is About

Now that you've got an overall understanding of the Android architecture, let's turn to what you will *not* learn in this book. First, you are not going to learn how to develop Android apps from scratch in this book. You will see many examples and source code listings; and while I will explain each section of code, you might have additional questions that you might not find answered in this book. You are required to have a certain degree of experience and skill at writing Java applications for the Android platform. I also assume that you have already setup your Android development environment using the Eclipse IDE. In this book, I will focus on how you can develop more secure applications for the Android operating system.

Android has had its fair share of security setbacks and a burgeoning list of malware that is worth examining and learning from. Armed with where to look and how to tackle security aspects of developing for Android will not necessarily make you a better coder, but it will start you on your way to becoming more responsible with your end users' privacy and security.

I've tried to write this book in a manner that will help you understand the concepts of security in relation to the applications you develop. In most cases, the best way I find I can achieve this is by teaching through example. Therefore, you will usually find me asking you to write and execute source code listings first. I will then follow up with an explanation of the specific concept that we are covering. With this in mind, let's take a look at some of the security controls available on the Android operating system.

# Security

*Security isn't a dirty word, Blackadder!*

—General Melchett, *Blackadder IV*

Security is a vast subject and is applicable to many areas depending on what context it is taken in. I wrote this book to cover a small component of a small component of security. It is written to give you a good understanding of Android application security. However, what does that really mean? What are we trying to secure? Who will benefit from this? Why is it important? Let's try to answer those questions and possibly come up with a few new ones.

First, let's identify who you really are. Are you a developer? Maybe you're a security practitioner conducting research. Alternatively, maybe you're an end user interested in safeguarding yourself from an attack. I'd like to think that I fit into each of these categories. No doubt, you will fit into one or more of them. The vast majority, however, will fit into one category: an end user who wants to use the features of a well-written application in a manner that does not compromise her privacy and security. If you're a developer, and I'm guessing you are if you've picked this book up, this is your target audience: the end user. You write applications to distribute to your users. You may choose to sell them or give them away for free. Either way, you are writing applications that will end up installed on someone else's device, possibly thousands of miles away.

## Protect Your User

Your application should strive to provide the best functionality possible while taking care to protect your users' data. This means thinking about security before you begin development.

Your user might not always know about the security practices you employ “under the hood” of your application, but one breach in your application is all it will take to ensure that all his Twitter and Facebook followers find out. Planning and thinking about security prior to the development phase of your application can save you the embarrassment of bad reviews and the loss of paying customers. The end user is almost never quick to forgive or forget.

As we go along, you will learn principles and techniques to identify sensitive user data and create a plan to protect this data. The goal is to eliminate or vastly reduce any unintentional harm your application could cause. So, what are you really protecting the end user from?

## Security Risks

Mobile device users face some unique risks when compared with desktop computer users. Aside from the higher possibility of losing or having their device stolen, mobile device users risk losing sensitive data or having their privacy compromised. Why would this be different from desktop users? First, the quality of data stored on a user's mobile device tends to be more personal. Apart from e-mail, there are instant messages, SMS/MMS, contacts, photos, and voicemail. “So what?” you say. “Some of these things exist on a desktop computer.” True, but consider this: The data on your mobile device is most likely going to be of higher value than that

on your desktop because you carry it around with you all the time. It is a converged platform of both your computer and mobile phone that contains a richer collection of personal data. Because the level of user interaction is higher on the smartphone, the data is always newer than on your desktop computer. Even if you have configured real-time sync to a remote location, that still only protects you from a loss of data and not a loss of privacy.

Consider also that the format of data stored on mobile devices is fixed. Every phone will have SMS/MMS, contacts, and voicemail. Phones that are more powerful will have photos, videos, GPS locations, and e-mail, but all of it is common regardless of the operating system. Now consider how important all of this information is to an end user. To a user who has no backups, losing data of this nature can be unthinkable. Losing important phone numbers, precious moments of her daughter's first steps caught on video, or important SMS messages can be catastrophic to the everyday phone user.

What about the user who combines both business and personal activities on his phone? What would you do if someone copied an entire file of passwords for your office server farm from your phone? Or if an e-mail containing trade secrets and confidential pricing for proposals leaked out onto the Internet? What if you lost the address of your child's school? Consider a stalker gaining access to this information and more, such as your home address and phone number.

It is clear when you think about it that the data stored on the phone is, in most cases, far more valuable than that of the device itself. The most dangerous type of attack is the one that takes place silently and remotely; an attacker does not need physical access to your phone. These types of attacks can happen at any time and can often happen because of weak security elsewhere on the device. These lapses in security might not be because your application is insecure. They could be due to a bug in the kernel or web browser. The question is this: can your application protect its data from attackers even when they gain access to the device through different routes?

## Android Security Architecture

As we discussed previously, Android runs on top of the Linux 2.6 kernel. We also learned that the Android Linux kernel handles security management for the operating system. Let's take a look at the Android Security Architecture.

### Privilege Separation

The Android kernel implements a privilege separation model when it comes to executing applications. This means that, like on a UNIX system, the Android operating system requires every application to run with its own user identifier (uid) and group identifier (gid).

Parts of the system architecture themselves are separated in this fashion. This ensures that applications or processes have no permissions to access other applications or processes.

## WHAT IS PRIVILEGE SEPARATION?

*Privilege separation* is an important security feature because it denies one of the more common types of attacks. In many cases, the first attack that is performed is not the most effective one. It is usually the stepping-stone or gateway to a bigger attack. Often, attackers will exploit one component of a system first; and once there, they will try to attack a more important component in the system. If both these components are running with the same privileges, then it is a very trivial task for the attacker to hop from one component to the next. By separating privileges, the attacker's task becomes more difficult. He has to be able to escalate or change his privileges to that of the component he wishes to attack. In this manner, the attack is stopped, if not slowed.

Because the kernel implements privilege separation, it is one of the core design features of Android. The philosophy behind this design is to ensure that no application can read or write to code or data of other applications, the device user, or the operating system itself. Thus, an application might not be able to arbitrarily use the device's networking stack to connect to remote servers. One application might not read directly from the device's contact list or calendar. This feature is also known as *sandboxing*. After two processes have run in their own sandboxes, the only way they have to communicate with each other is to explicitly request permission to access data.

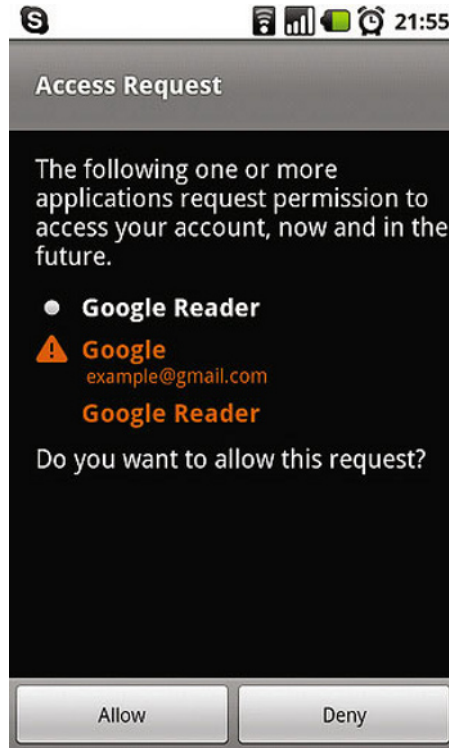
## Permissions

Let's take a simple example. We have an application that records audio from the built-in microphone of the device. For this application to work correctly, the developer has to make sure to add a request for the `RECORD_AUDIO` permission in the application's `AndroidManifest.xml` file. This allows our application to request permission to use the system component that handles audio recording. But who decides whether to grant or deny access? Android allows the end user to perform this final approval process. When the user installs our application, he is prompted with the screen shown in Figure 1-4. It is worthwhile to note that no prompt for permissions will take place when the application is executing. Instead, the permission will need to be granted at install time.

If we do not explicitly set our need for the `RECORD_AUDIO` permission, or if the device owner does not grant us the permission after we request it, then an exception will be thrown by the VM and the application will fail. It is up to the developer to know to request the permission and handle the scenario where permission is not granted by catching the relevant exception. To request this permission, the following tag must be included in the `AndroidManifest.xml` file of the project:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

The full list of permissions is given in this book's appendix.



*Figure 1-4. The Android permissions request screen*

## Application Code Signing

Any application that is to run on the Android operating system must be signed. Android uses the certificate of individual developers in order to identify them and establish trust relationships among the various applications running in the operating system. The operating system will not allow an unsigned application to execute. The use of a certification authority to sign the certificate is not required, and Android will happily run any application that has been signed with a self-signed certificate.

Like permissions checks, the certificate check is done only during installation of the application. Therefore, if your developer certificate expires after your application is installed on the device, then the application will continue to execute. The only difference at this point would be that you would need to generate a new certificate before you could sign any new applications. Android requires two separate certificates for debug versions of your application and release versions of your application. Generally, the Eclipse environment running the Android Development Tools (ADT) is already setup to help you generate your keys and install your certificate, so that your applications can be automatically packaged and signed. The Android emulator behaves identically to the physical device. Like the physical device, it will only execute signed applications. We will cover application code signing in detail, as well as publishing and selling your applications online.