Quick answers to common problems

Android Security Cookbook

Practical recipes to delve into Android's security mechanisms by troubleshooting common vulnerabilities in applications and Android OS versions

Keith Makan
Scott Alexander-Bown

[PACKT] PUBLISHING open source community experience distilled

Android Security Cookbook

Practical recipes to delve into Android's security mechanisms by troubleshooting common vulnerabilities in applications and Android OS versions

Keith Makan

Scott Alexander-Bown

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

Android Security Cookbook

First published: December 2013

Production Reference: 1171213

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78216-716-7

www.packtpub.com

Cover Image by Joseph Kiny (joseph.kiny@gmail.com)

Credits

Authors

Keith Makan

Scott Alexander-Bown

Reviewers

Miguel Catalan Bañuls

Seyton Bradford

Nick Glynn

Rui Gonçalo

Elliot Long

Chris Pick

Acquisition Editors

Pramila Balan

Llewellyn F. Rozario

Lead Technical Editor

Ritika Dewani

Copy Editors

Alisha Aranha

Roshni Banerjee

Tanvi Gaitonde

Aditya Nair

Karuna Narayanan

Shambhavi Pai

Laxmi Subramanian

Technical Editors

Zainab Fatakdawala

Hardik B. Soni

Sebastian Rodrigues

Project Coordinator

Akash Poojary

Proofreaders

Lauren Harkins

Amy Johnson

Indexer

Rekha Nair

Production Coordinator

Kirtee Shingan

Cover Work

Kirtee Shingan

About the Authors

Keith Makan is a former computer science and physics student, and a passionate hobbyist and security researcher. He spends most of his free time reading source code, performing reverse engineering and fuzz testing, and developing exploits for web application technology.

Keith works professionally as an IT security assessment specialist. His personal research has won him spots on the Google Application Security Hall of Fame numerous times. He has developed exploits against Google Chrome's WebKit XSSAuditor, Firefox's NoScript Add-on, and has often reported security flaws and developed exploits for WordPress plugins.

I would like to thank my mom, dad, and other family members for supporting my crazy ideas and always being a great motivation to me.

Scott Alexander-Bown is an accomplished developer with experience in financial services, software development, and mobile app agencies. He lives and breathes Android, and has a passion for mobile app security.

In his current role as senior developer, Scott specializes in mobile app development, reverse engineering, and app hardening. He also enjoys speaking about app security and has presented at various conferences for mobile app developers internationally.

Most importantly, I'd like to thank my wife Ruth. Your love and encouragement make everything I do possible. High five to my son Jake who keeps me going with his laughter and cute smiles.

Additionally, I would like to thank the following people:

Keith, Barbara and Kirk Bown, and Mhairi and Robert Alexander for your love and support.

Andrew Hoog and the viaForensics team for their support, insight, and expertise in mobile security.

Mark Murphy, Nikolay Elenkov, Daniel Abraham, Eric Lafortune, Roberto Tyley, Yanick Fratantonio, Moxie Marlinspike, the Guardian Project, and the Android Security team whose blog articles, papers, presentations, and/or code samples have been interesting and extremely useful when learning about Android security.

Keith Makan for his enthusiasm, guidance, and for welcoming me aboard the *Android Security Cookbook* ship.

The technical reviewers for their attention to detail and valuable feedback.

Finally, thanks to you, the reader—I hope you find this book useful and it allows you to create more secure apps.

About the Reviewers

Miguel Catalan Bañuls is a young engineer whose only purpose is to try and make his little contribution to changing the world. He is mainly a software developer, but is actually a team leader.

He holds a degree in Industrial Engineering and is a partner at Geeky Theory. Also, he is the vice president of the IEEE Student Branch of the Miguel Hernandez University (UMH in Spanish).

I want to thank both my spouse and my parents for their patience and understanding as they have to share me with my work.

Seyton Bradford is a software developer and an engineer with over 10 years experience in mobile device security and forensics.

He currently works at viaForensics as a Senior Software Engineer focusing on app and mobile device security.

He has presented his work across the globe and acted as a reviewer for academic journals.

I'd like to thank my family and friends for their support for my career and work.

Nick Glynn is currently employed as a technical trainer and consultant delivering courses and expertise on Android, Python, and Linux at home in the UK and across the globe. He has a broad range of experience, from board bring-up, Linux driver development, and systems development through to full-stack deployments, web app development, and security hardening for both the Linux and Android platforms.

I would like to thank my family for their love, and my beautiful baby girl Inara for always brightening my day.

Rui Gonçalo is finishing his Masters thesis at University of Minho, Braga, Portugal, in the field of Android security. He is developing a new feature that aims at providing users with fine-grained control over Internet connections. His passion for mobile security arose from attending lectures on both cryptography and information systems security at the same university, and from several events held by the most important companies in the field in Portugal. He provides the point of view of an Android security beginner who sees this book as a must read for those keen to become security experts.

I would like to thank the staff at Packt Publishing in charge of this book for making me absolutely sure that mobile security will fulfill my needs in the world of software.

Elliot Long grew up in Silicon Valley and has been creating mobile apps since 2005. He is the co-founder of the mobile travel guide producer mycitymate SL/GmbH. Since 2009, he has worked as Lead Android and BlackBerry Developer for Intohand Ltd.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

Android has quickly become one of the most popular mobile operating systems, not only to users but also developers and companies of all kinds. Of course, because of this, it's also become quite a popular platform to malicious adversaries.

Android has been around in the public domain since 2005 and has seen massive growth in capability and complexity. Mobile smart phones in general now harbor very sensitive information about their users as well as access to their e-mails, text messages, and social and professional networking services. As with any software, this rise in capability and complexity also brings about a rise in security risk; the more powerful and more complex the software becomes, the harder they are to manage and adapt to the big bad world.

This applies especially to software on mobile smart phones. These hot beds of personal and sensitive information present an interesting security context in which solve problems. From one perspective, the mobile smart phone security context is very difficult to compare to the servers on a network or in the "cloud" because, by their very nature, they are not mobile. They cannot be moved or stolen very easily; we can enforce both software and physical security measures to protect unauthorized access to them. We can also monitor them constantly and rapidly respond to the security incidents autonomously. For the devices we carry around in our pockets and handbags, and forget in taxi cabs, the playing field is quite different!

Android users and developers express a need to be constantly aware of their mobile security risks and, because of this need, mobile security and risk assessment specialists and security engineers are in high demand. This book aims to smoothen the learning curve for budding Android security assessment specialists and acts as a tool for experienced Android security professionals with which to hack away at common Android security problems.

What this book covers

Chapter 1, *Android Development Tools*, introduces us to setting up and running the tools developers use to cook up Android applications and native-level components on the Android platform. This chapter also serves as an introduction to those who are new to Android and would like to know what goes into setting up the common development environments and tools.

Chapter 2, *Engaging with Application Security*, introduces us to the components offered by the Android operating system, dedicated to protecting the applications. This chapter covers the manual inspection and usage of some of the security-relevant tools and services used to protect applications and their interaction with the operating system.

Chapter 3, *Android Security Assessment Tools*, introduces some of the popular as well as new and upcoming security tools and frameworks used by Android security specialists to gauge the technical risks that applications expose their users to. Here you will learn to set up, run, and extend the hacking and reverse engineering tools that will be used in later chapters.

Chapter 4, *Exploiting Applications*, covers the casing exploitation techniques that target the Android applications. The content in this chapter spans all the Android application component types and details how to examine them for security risks, both from a source code and inter-application context. It also introduces more advanced usage of the tools introduced in *Chapter 3*, *Android Security Assessment Tools*.

Chapter 5, *Protecting Applications*, is designed to be the complete opposite of *Chapter 4*, *Exploiting Applications*. Instead of talking purely about application flaws, this chapter talks about application fixes. It walks readers through the useful techniques that developers can use to protect the applications from some of the attacks, which are detailed in *Chapter 4*, Exploiting Applications.

Chapter 6, *Reverse Engineering Applications*, helps the readers to learn to crack open the applications and teaches them the techniques that Android reverse engineers use to examine and analyze applications. You learn about the Dex file format in great detail, as well as how to interpret Dex bytecode into useful representations that make reverse engineering easier. The chapter also covers the novel methods that reverse engineers can use to dynamically analyze applications and native components while they are running on an Android operating system.

Chapter 7, *Secure Networking*, helps the readers to delve into the practical methods that application developers can follow to protect data while in transit across the network. With these techniques, you will be able to add stronger validation to the Secure Sockets Layer (SSL) communications.

Chapter 8, *Native Exploitation and Analysis*, is dedicated to covering the security assessment and testing techniques focused on the native context of the Android platform. Readers will learn to look for security flaws that can be used to root phones and escalate privileges on the Android systems as well as perform low-level attacks against native services, including memory corruption and race condition exploitation.

Chapter 9, *Encryption and Developing Device Administration Policies*, is focused heavily on how to use encryption correctly and avoid some of the common anti-patterns to keep data within your application secure. It recommends several robust and timesaving third-party libraries to quickly yet securely enhance the security of your applications. To wrap up, we will cover how to use the Android Device Administration API to implement and enforce enterprise security policies.

What you need for this book

Though there are some software requirements for the book, many of the walkthroughs in the book discuss downloading and installing the required software before actually getting down to using them to contribute to the topic being discussed.

That being said, here is a list of the software you will probably need to have before starting with the walkthroughs:

- The Android Software Development Kit (SDK)
- The Android Native Development Kit (NDK)
- The GNU C/C++ Compiler (GCC)
- The GNU Debugger (GDB)
- Python, preferably 2.7 but 3.0 should work fine
- Virtual box
- Ettercap (for Windows or Linux/Unix systems)
- Dex2Jar
- Objdump
- Radamsa
- JD-GUI
- The Java Development Kit (JDK)
- drozer, an Android security assessment framework
- The OpenSSL command-line tool
- The keytool command-line tool

Who this book is for

With some chapters dedicated to exploiting Android applications and others focused on hardening them, this book aims to show the two sides of the coin, the attacker and the defender.

Security researchers, analysts, and penetration testers will enjoy the specifics of how to exploit the Android apps. Application developers with an appetite to learn more about security will gain practical advice on how to protect their applications from attacks.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The system image ID you selected from the previous step must be specified using the `-t` switch."

A block of code is set as follows:

```
from drozer import android
from drozer.modules import common, Module
class AttackSurface(Module,common.Filters, common.PackageManager):
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold as follows:

```
from drozer import android
from drozer.modules import common, Module
class AttackSurface(Module,common.Filters, common.PackageManager):
```

Any command-line input or output is written as follows:

```
sudo aptitude update //If you have aptitude installed
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus, or dialog boxes, for example, appear in the text like this: "Once you've accepted the licenses, you can collect your documentation and APIs by clicking on **Install**".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Android Development Tools

In this chapter, we will cover the following recipes:

- Installing the Android Development Tools (ADT)
- Installing the Java Development Kit (JDK)
- Updating the API sources
- Alternative installation of the ADT
- Installing the Native Development Kit (NDK)
- Emulating Android
- Creating Android Virtual Devices (AVDs)
- Using the Android Debug Bridge (ADB) to interact with the AVDs
- Copying files off/onto an AVD
- Installing applications on the AVDs via ADB

Introduction

A very clever person once said that, "you should keep your friends close but your enemies closer". Being a security professional means keeping an eye on what developers are doing, have done, and are likely to do. This is because the decisions they make greatly affect the security landscape; after all, if no one wrote bad software, no one would exploit it!

Given that this book is aimed at anyone interested in analyzing, hacking, or developing the Android platform, the *know thy enemy* concept applies to you too! Android developers need to stay somewhat up to date with what Android hackers are up to if they hope to catch security vulnerabilities before they negatively affect the users. Conversely, Android hackers need to stay up to date with what Android developers are doing.

The upcoming chapters will walk you through getting the latest and greatest development and hacking tools and will get you to interact directly with the Android security architecture, both by breaking applications and securing them.

This chapter focuses on getting the **Android Development Tools** (**ADT**) up and running and discusses how to troubleshoot an installation and keep them up to date. If you feel you are already well-acquainted with the Android development environment and tool chains, feel free to skip this chapter.

Without further ado, let's talk about grabbing and installing the latest Android Development Tools.

Installing the Android Development Tools (ADT)

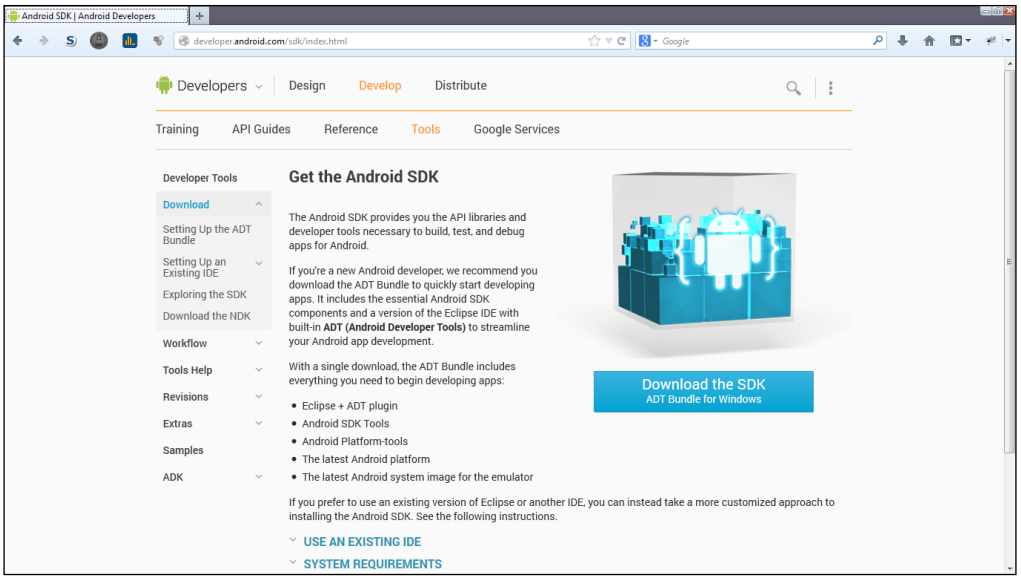
Given that there are many versions of the Android framework already deployed on mobile platforms and a variety of handsets that support it, Android developers need tools that give them access to many device- and operating system-specific Application Programming Interfaces (**APIs**) available on the Android platform.

We're talking about not just the Android APIs but also handset-specific APIs. Each handset manufacturer likes to invest in the developer mindshare in their own way by providing exclusive APIs and services to their developers, for example, the HTC OpenSense APIs. The ADT consolidates access to these APIs; provides all the necessary tools to debug, develop, and deploy your Android apps; and makes it easy for you to download them and keep them up to date.

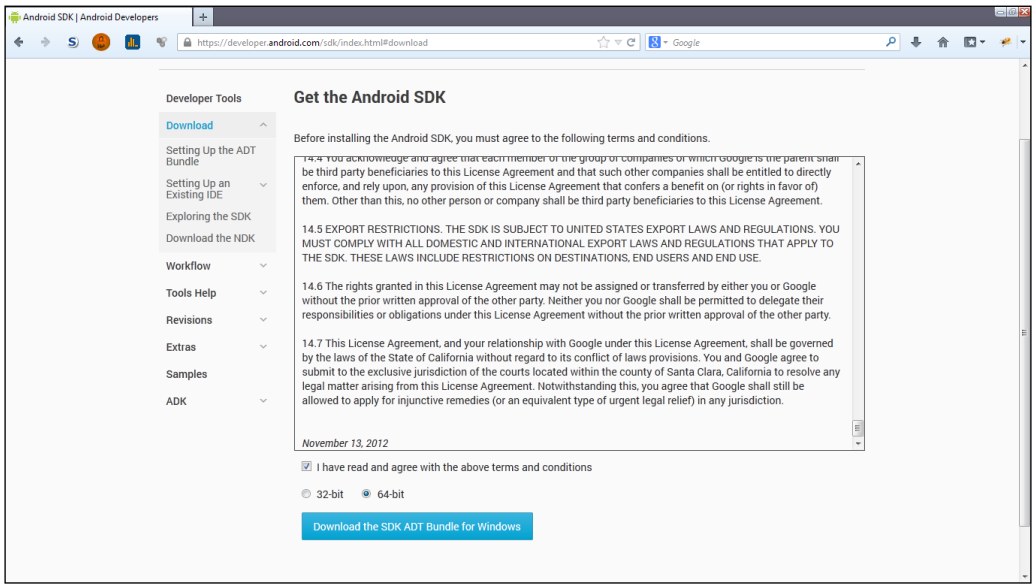
How to do it...

The following steps will walk you through the process of downloading the ADT and getting them up and running:

1. You'll need to head over to `https://developer.android.com` and navigate to the ADT **Download** page or just visit `https://developer.android.com/sdk/index.html#download`. You should see a page like the one in the following screenshot:



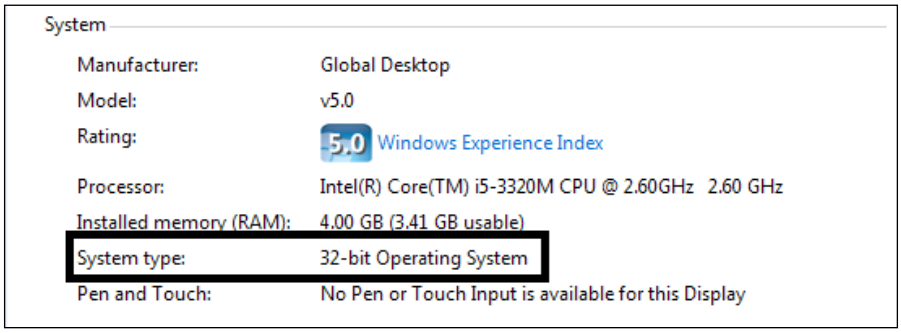
2. Once you're there, click on **Download the SDK** and the following screen should appear:



3. Of course, you will need to accept the license agreement before downloading and select the appropriate CPU type, or register size if you're not sure how to check your CPU type.

   On Windows, you need to complete the following steps:

   1. Click on **Start**.
   2. Right-click on **My Computer**.
   3. Select **Properties**.
   4. A window with your computer's system-specific information should pop up. The information you are looking for should be under the **System** section, labeled **System type**.

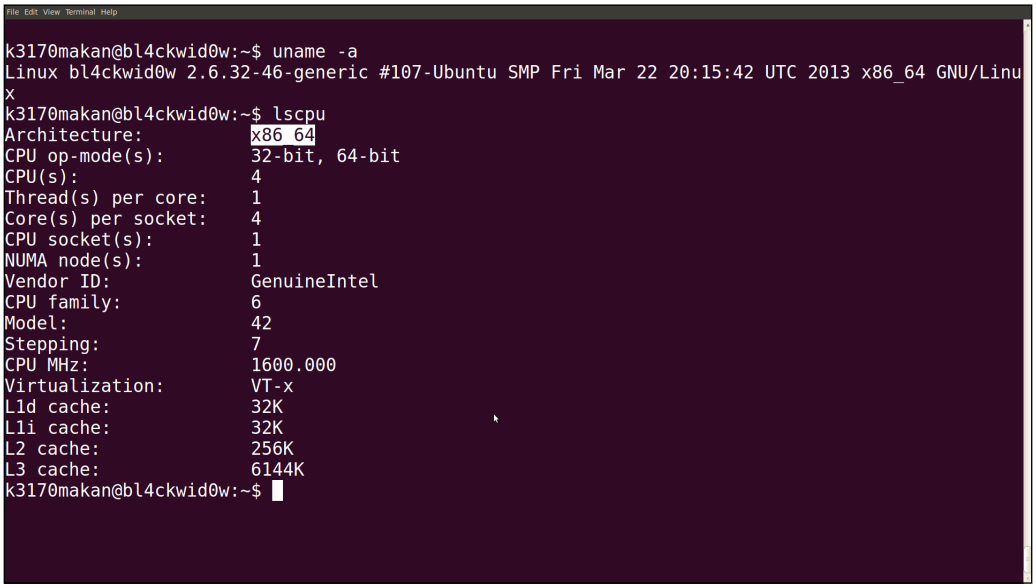


   To check your system type on Ubuntu, Debian, or Unix-based distributions, perform the following steps:

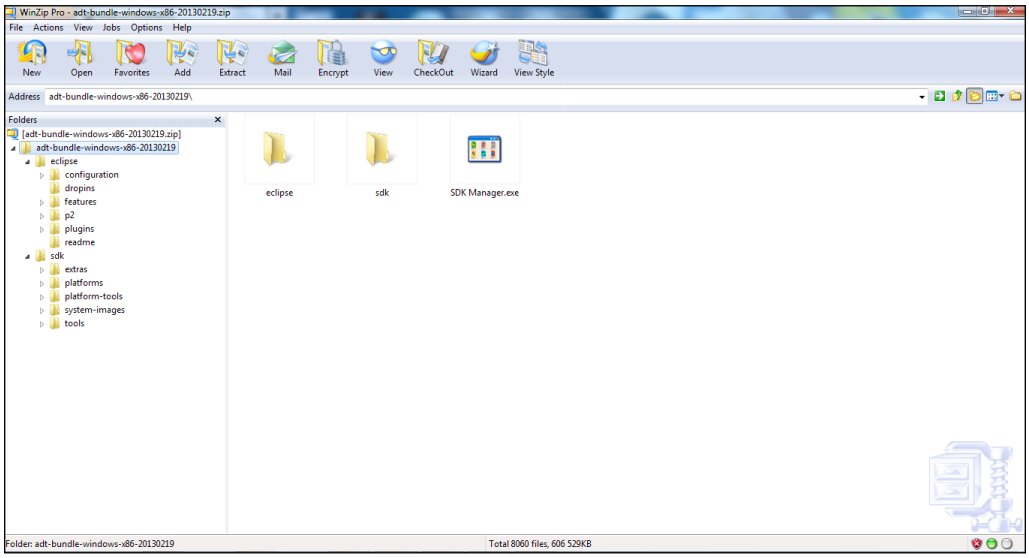
   1. Open a terminal either by pressing *Ctrl* + *Alt* + *T* or simply launching it using the graphical interface.
   2. Execute the following command:

      ```
      uname -a
      ```

3. Alternatively, you could use `lscpu` that should show you something like the following screenshot:



4. When you're happy with the license agreement and you've selected the correct system type, click on **Download** in the ADT **Download** page. Once the ZIP file has been downloaded, it should look like the following screenshot on Windows:



The archive will have the same structure on the Linux- or Unix-based distributions.

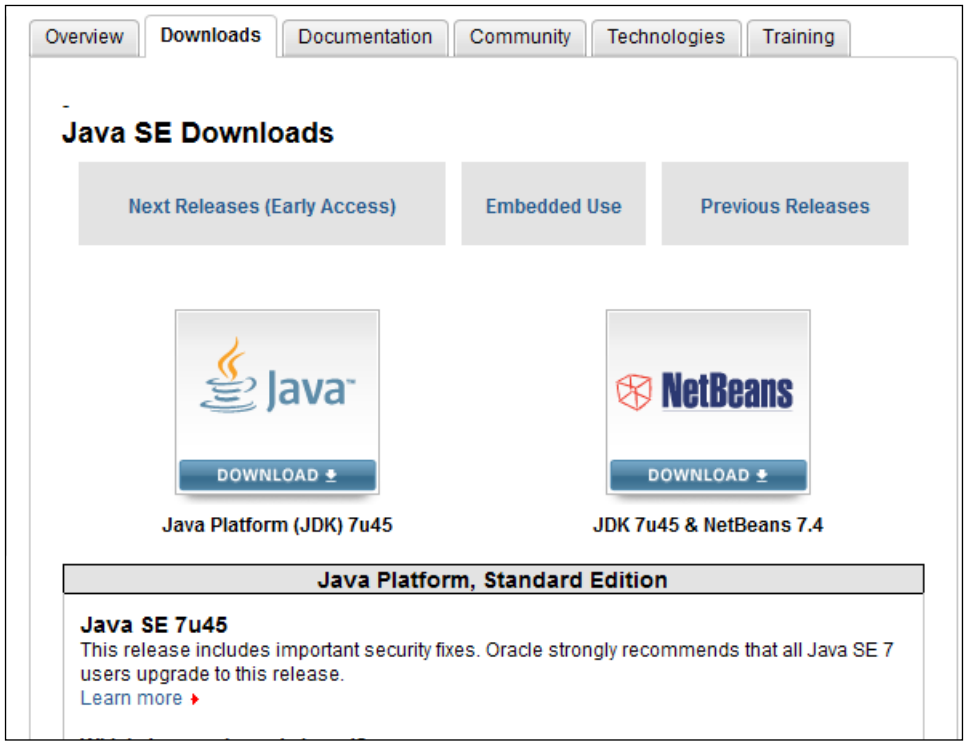
Installing the Java Development Kit (JDK)

Android uses a customized version of the Java runtime to support its applications. This means, before we can get going with Eclipse and developing Android applications, we actually need to install the Java runtime and development tools. These are available in the **Java Development Kit** (**JDK**).

How to do it...

Installing the JDK on Windows works as follows:

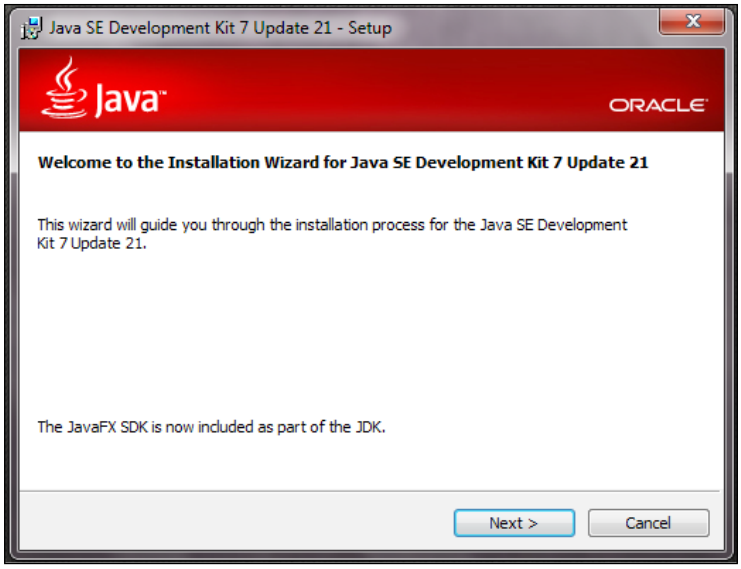
1. Grab a copy of the JDK from Oracle's **Downloads** page, `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. Click on **DOWNLOAD**. The following screenshot shows the **Downloads** page:



2. Make sure to select the appropriate version for your system type; see the previous walkthrough to find out how to check your system type. The following screenshot highlights the Windows system types supported by the Oracle Java JDK:

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.67 MB	jdk-7u45-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u45-linux-arm-vfp-sflt.tar.gz
Linux x86	115.62 MB	jdk-7u45-linux-i586.rpm
Linux x86	132.9 MB	jdk-7u45-linux-i586.tar.gz
Linux x64	116.91 MB	jdk-7u45-linux-x64.rpm
Linux x64	131.7 MB	jdk-7u45-linux-x64.tar.gz
Mac OS X x64	183.84 MB	jdk-7u45-macosx-x64.dmg
Solaris x86 (SVR4 package)	139.93 MB	jdk-7u45-solaris-i586.tar.Z
Solaris x86	95.02 MB	jdk-7u45-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.6 MB	jdk-7u45-solaris-x64.tar.Z
Solaris x64	16.23 MB	jdk-7u45-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.38 MB	jdk-7u45-solaris-sparc.tar.Z
Solaris SPARC	98.17 MB	jdk-7u45-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.91 MB	jdk-7u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.26 MB	jdk-7u45-solaris-sparcv9.tar.gz
Windows x86	123.49 MB	jdk-7u45-windows-i586.exe
Windows x64	125.31 MB	jdk-7u45-windows-x64.exe

3. After downloading the JDK, run the `jdk-[version]-[platform version].exe` file. For instance, you could have an EXE file named something like `jdk-7u21-windows-i586.exe`. All you need to do now is follow the prompts until the installation of all the setups is completed. The following screenshot is what the install wizard should look like once it's launched:



Once the install wizard has done its job, you should see a fresh install of your JDK and JRE under `C:\Program Files\Java\jdk[version]` and should now be able to launch Eclipse.

There's more...

Installing the Java Runtime and Development tools on Ubuntu Linux is somewhat simpler. Seeing that Ubuntu has a sophisticated package and repository manager, all you need to do is make use of it by firing off a few simple commands from the terminal window. You need to execute the following steps:

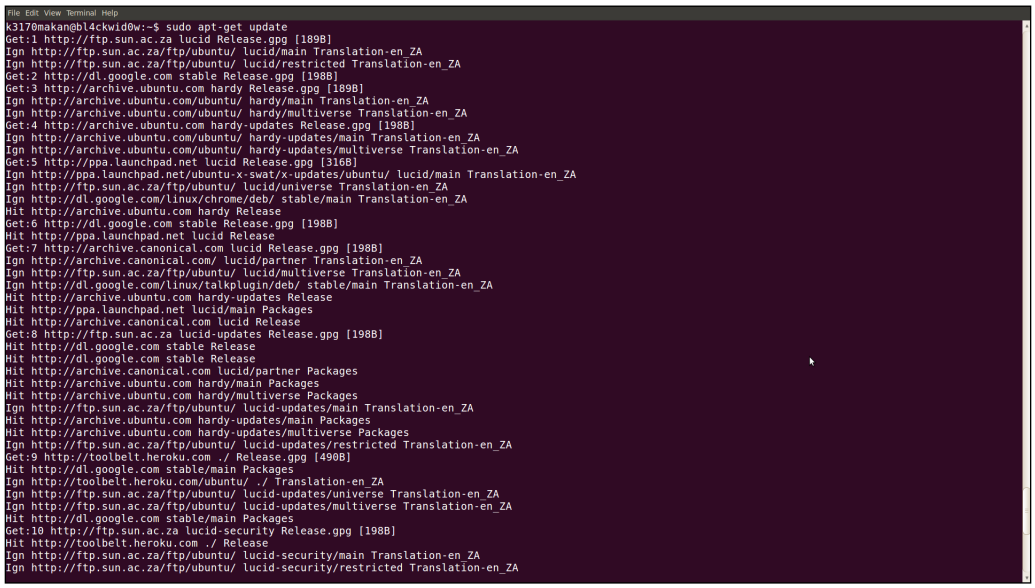
1. Open a terminal, either by searching for the terminal application via your Unity, KDE, or Gnome desktop or by pressing *Ctrl* + *Alt* + *T*.
2. You may need to update your package list before installation, unless you've already done that a couple of minutes ago. You can do this by executing either of the following commands:

   ```
   sudo aptitude update   //If you have aptitude installed
   ```

   Or:

   ```
   sudo apt-get update
   ```

   You should see your terminal print out all the downloads it's performing from your repositories as shown in the following screenshot:



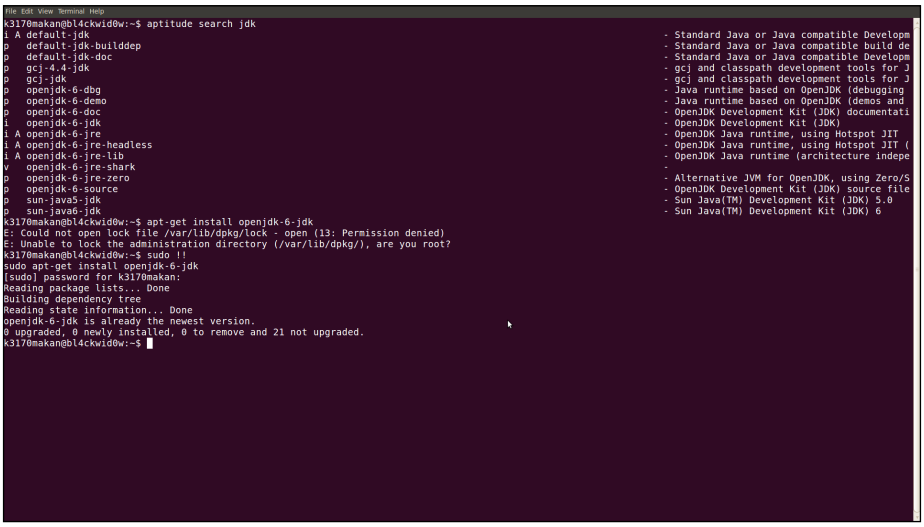
3. Once that's done, execute the following command:

   ```
   sudo apt-get install openjdk-[version]-jdk apt-get
   ```

   You will need to enter your password if you have been added to your `sudoers` file correctly. Alternatively, you could borrow root privileges to do this by executing the following command, assuming that you have the root user's password:

   ```
   su root
   ```

   This is displayed in the following screenshot:



Once your JDK is installed properly, you should be able to launch Eclipse and get going with your Android development. When you launch Eclipse, you should see the following screenshot:

After successful installation, the toolbar in your Eclipse installation should look something like the one in the following screenshot:

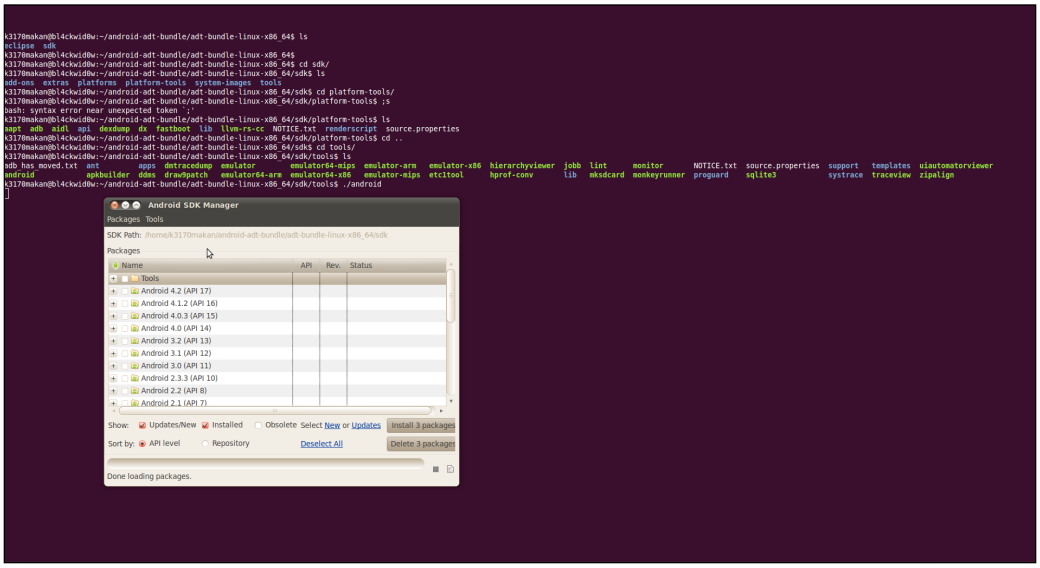
Updating the API sources

The SDK manager and related tools come bundled with the ADT package; they provide access to the latest and most stable APIs, Android emulator images, and various debugging and application testing tools. The following walkthrough shows you how to update your APIs and other Android development-related resources.

How to do it...

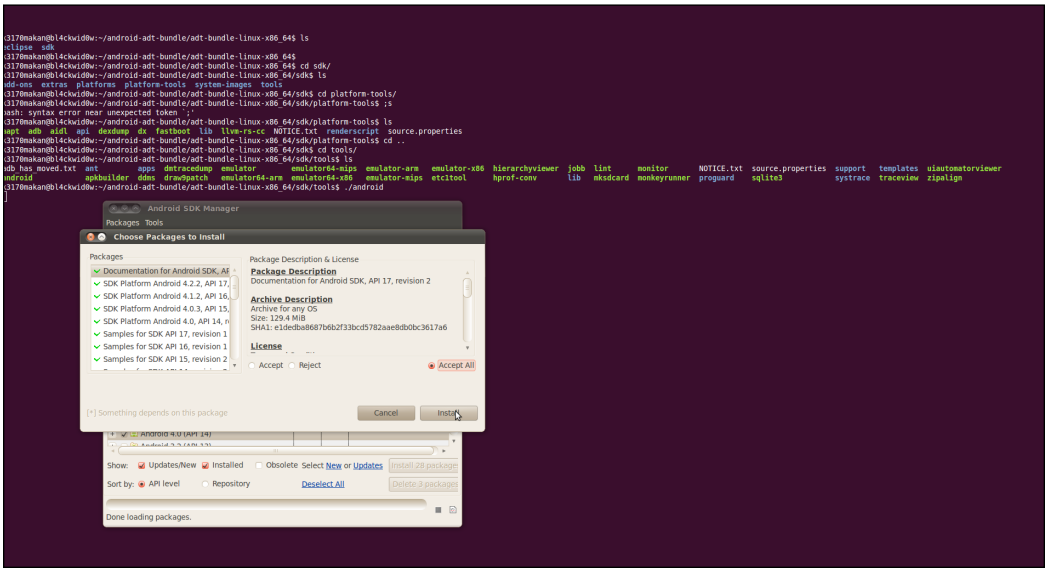
Updating the APIs for your ADT works as follows:

1. Navigate to the SDK manager. If you're doing this all from Windows, you should find it in the root of the `ADT-bundle` folder called `SDK Manager.exe`. Ubuntu users will find it at `[path to ADT-bundle]/sdk/tools/android`.
2. All you need to do is launch the SDK manager. It should start up and begin retrieving a fresh list of the available API and documentation packages.



3. You will need to make sure that you select the **Tools** package; of course, you could also select any other additional packages. A good idea would be to download the last two versions. Android is very backward compatible so you don't really need to worry too much about the older APIs and documentation, unless you're using it to support really old Android devices.
4. You will need to indicate that you accept the license agreement. You can either do this for every single object being installed or you can click on **Accept All**.

5. Once you've accepted the licenses, you can collect your documentation and APIs by clicking on **Install** as shown in the following screenshot:



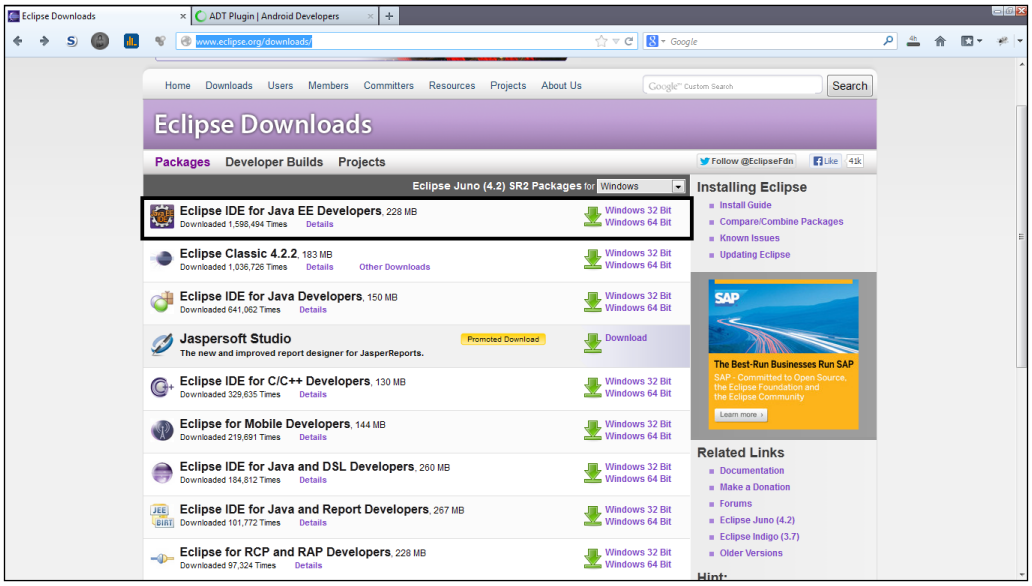
Alternative installation of the ADT

If the preceding methods for installing Eclipse and the ADT plugin don't work for some reason, you could always take the old school route and download your own copy of Eclipse and install the ADT plugin manually via Eclipse.

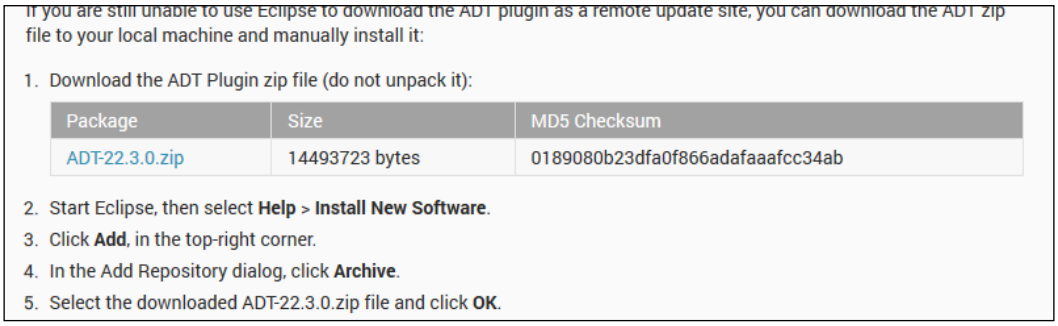
How to do it...

Downloading and plugging in the ADT works as follows:

1. Download Eclipse—Helios or a later version—from `http://www.eclipse.org/downloads/`. Please make sure to select the appropriate version for your operating system. You should see a page that looks like the following screenshot:

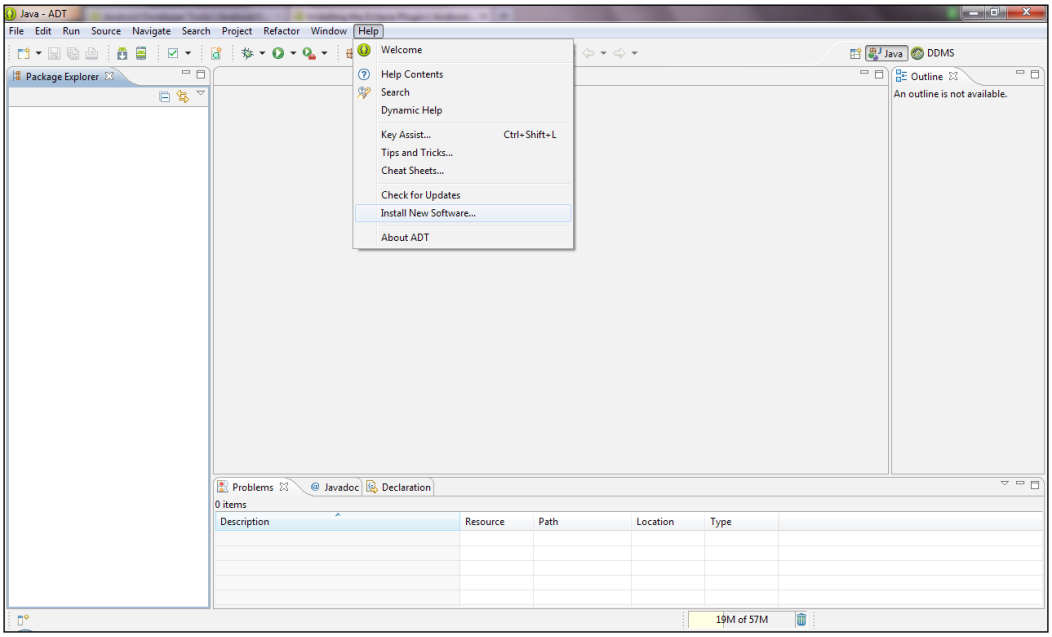


2. Download the ADT bundle for your platform version from the Android website, `http://developer.android.com/sdk/installing/installing-adt.html`. The following screenshot displays a part of the page on this website:



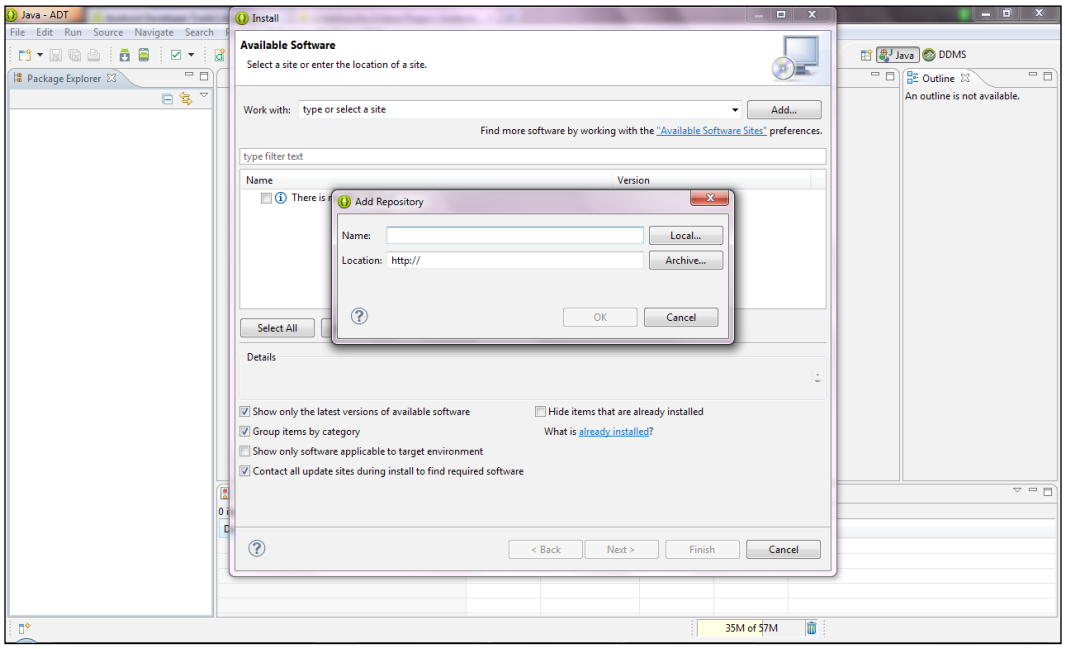
3. Make sure you have the Java JDK installed.

4. If your JDK installation is good to go, run the Eclipse installer you downloaded in step 1.
5. Once Eclipse is installed and ready to go, plugin your ADT.
6. Open Eclipse and click on the **Help** button in the menu bar.

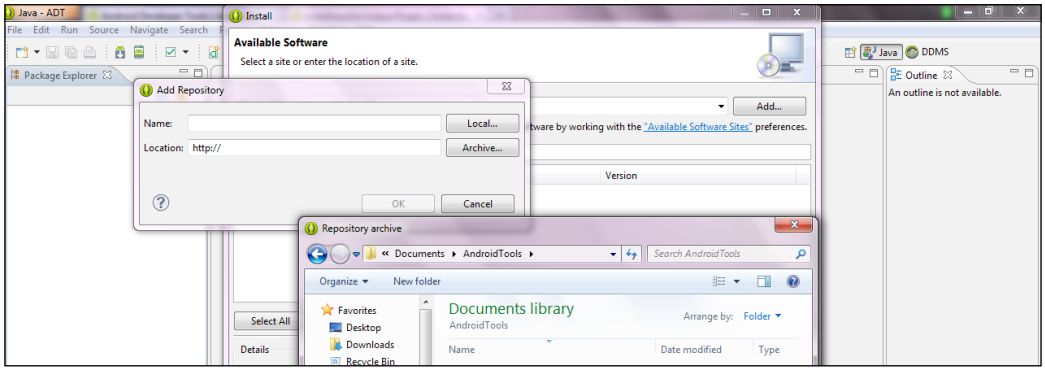


7. Click on **Install New Software...**.

8. The **Available Software** dialog box will pop up. You need to click on **Add...**.

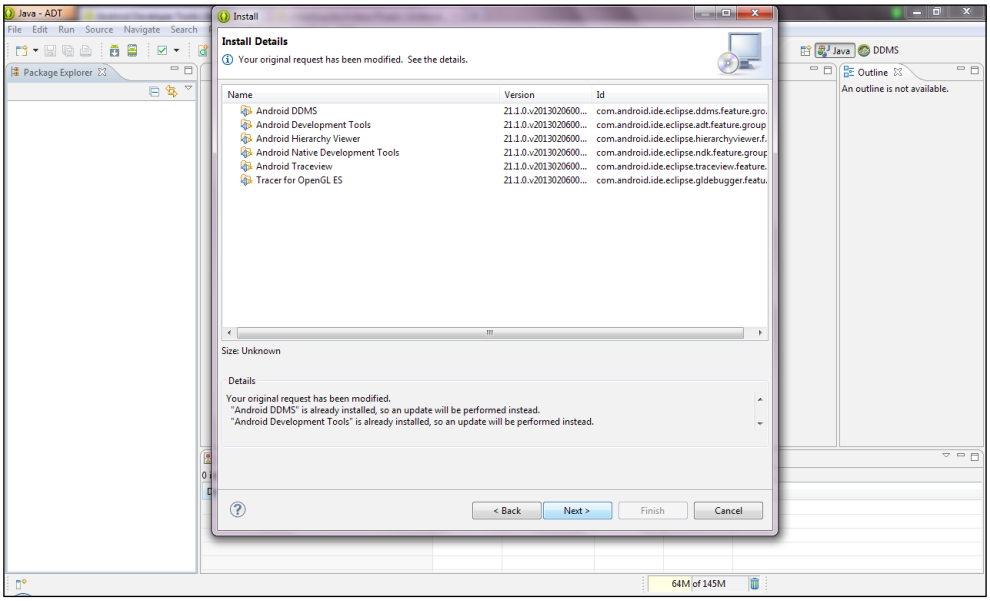


9. The **Add Repository** dialog box will show up. You need to click on the **Archive...** button.
10. A file browser should pop up. At this point, you will need to navigate to the ADT ZIP file that you downloaded in the previous steps.

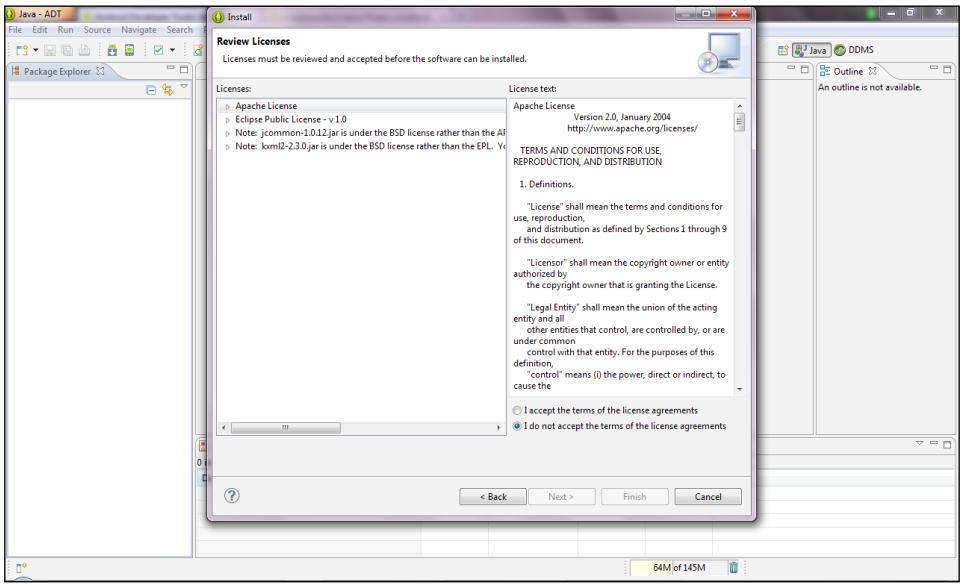


11. After finding the ADT file, click on **Open**.
12. Then click on **OK**.

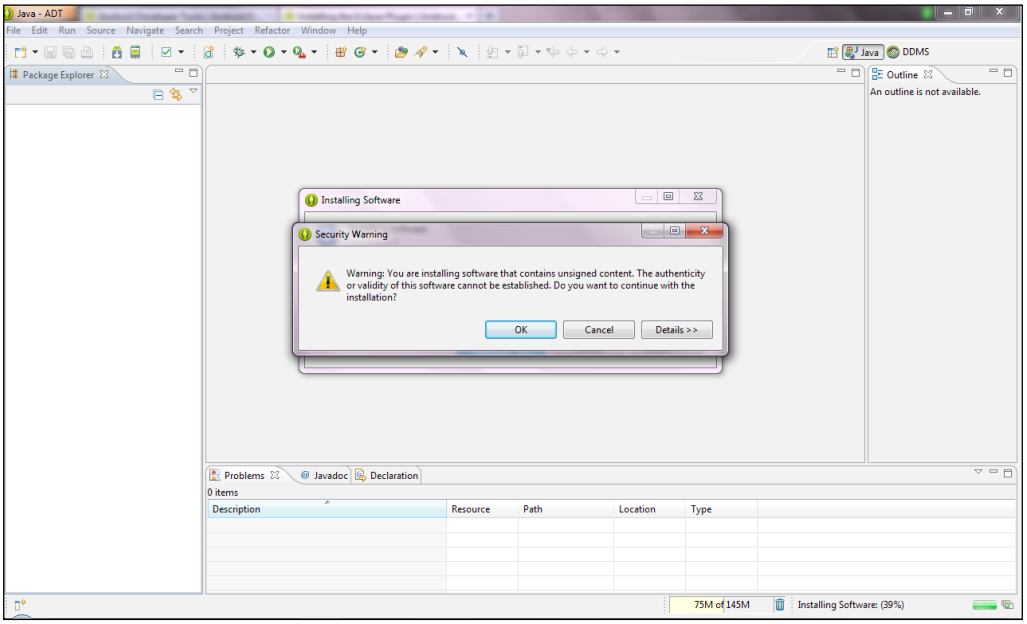
13. You will be shown the available packages in the `.zip` archive. Click on **Select All** and then on **Next**.



14. You will now need to accept the license agreement; of course, you reserve the right not to. It's always a good idea to give it a read. If you're happy, select the **I accept the terms of the license agreements** option and then click on **Finish**.



15. The software installation will now begin. You may get a warning stating that the content is unsigned and the authenticity cannot be verified. Click on **OK**.



16. Restart Eclipse.

The Android SDK, the device emulator, and the supporting Eclipse functionality should be ready to go now. See your Eclipse toolbar. It should have some new icons.

Installing the Native Development Kit (NDK)

If you want to do any low-level exploitation or development on your Android device, you will need to make sure that you can write applications at a lower level on the Android platform. Low level means development in languages like C/C++ using compilers that are built to suit the embedded platform and its various nuances.

What's the difference between Java and the native/low-level programming languages? Well, this topic alone could fill an entire book. But to state just the bare surface-level differences, Java code is compiled and statically—meaning the source code is analyzed—checked before being run in a virtual machine. For Android Java, this virtual machine is called the Dalvik—more on this later. The natively developed components of Android run verbatim—as their source code specifies—on the embedded Linux-like operating system that comes shipped with the Android devices. There is no extra layer of interpretation and checking—besides the odd compiler extensions and optimizations—that goes into getting the native code to run.