Android Forensics

Investigation, Analysis, and Mobile Security for Google Android

Andrew Hoog

John McCash, Technical Editor

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Syngress is an imprint of Elsevier

Acquiring Editor: Angelina Ward
Development Editor: Heather Scherer
Project Manager: Danielle S. Miller
Designer: Russell Purdy

Syngress is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods or professional practices may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information or methods described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
Application submitted

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 978-1-59749-651-3

Printed in the United States of America

11 12 13 14 15 10 9 8 7 6 5 4 3 2 1

Dedication

To my beautiful spouse who has endured my extended absenteeism as I wrote this book. She is my motivation, my friend, my partner, and the root of my happiness. This book is dedicated to her.

And to my wonderful daughters. You light up our lives and know more about Android forensics than any other 6-year-olds. May your lives be full of learning, success, and happiness.

Acknowledgements

I now understand that the phrase “It takes a village…” applies equally to writing a book as it does to raising children. As such, I wish to acknowledge the village:

- My family (see Dedication).
- Lee Haas, for excellent editing and attempts to keep me on schedule
- Ted Eull, who coined to term “deHOOGification,” which provides an immense service to you, the reader, as the ideas bouncing around in my head don’t always come out that clear when I persist them to words. Ted is also a great friend and all around swell guy. Many thanks to his better half for her patience in putting up with the long hours racked up by motivated geeks at a tech start-up.
- Chris Triplett, for diving head first into Android and doing an amazing job at it. Chris is also excellent at patching drywall and providing some comic relief by applying farm English to digital forensics.
- Katie Strzempka, for generally taking care of that other book (“iPhone and iOS Forensics”). Please buy that one too, seriously.
- My parents, Stevie and Al, who set me on the correct path from the start and were always there to remind me if I swerved off a bit.
- To Harmonee and Hadabogee, whose help with our daughters, dinner, and other areas is immensely appreciated.
- To the men and women who bravely serve the public interest in Local, State, and Federal law enforcement and other government agencies. We appreciate all that you do to protect and serve our communities and countries.
- To Google, for seeing the value in Android and creating a new paradigm of openness for mobile devices.
- To Apple, for providing the opposite paradigm.
- And finally to the reader. I hope that you find this book useful and certainly do appreciate your support.

Introduction

The Android mobile platform has quickly risen from its first phone in October 2008 to the most popular mobile operating system in the world by early 2011. The explosive growth of the platform has been a significant win for consumers with respect to competition and features. However, forensic analysts and security engineers have struggled as there is a lack of knowledge and supported tools for investigating these devices. This book seeks to address issues not only by providing in-depth insights into Android hardware, software, and file systems but also by sharing techniques for the forensic acquisition and subsequent analysis of these devices. For readers with limited forensic experience, this book creates step-by-step examples that use free, open source utilities so the reader can directly participate in the examples. As the free Android software development kit provides a full Android emulator, readers do not even need to possess an Android device.

As Android devices grow in numbers, an increased awareness of the data they possess will equally grow. Unfortunately, much of that interest will come from cyber criminal organizations who realize that successful attacks against the platform will yield significant results as the devices contain enormous quantities of personal and business information. The solution to this threat requires a deep understanding of the platform not only from core Android developers and manufacturers but also from app developers and corporate security officers. More secure apps will prevent loss of sensitive information as well as strong policies that can be put in place by IT security managers.

Although most of the discussed statistics about Android focus on smartphones and now tablets, there are many more devices that currently or in the near future will run Android. Some examples include vehicles, televisions, GPS, gaming devices, netbooks, and a wide variety of other consumer devices. Android will be present in an increasingly significant percentage of investigations for both forensic analysts and security engineers. Finally, the appeal of Android is not specific to any particular country or region and as such will impact individuals, corporations, and agencies throughout the world.

The following paragraphs contain a brief summary of each of the chapters.

CHAPTER 1

This chapter provides not only a history of the Android platform but also discusses the Android Open Source Project (AOSP), the internationalization of the platform, the Android Market, a brief Linux tutorial, and a quick fb-non-chapter to Android forensics. It also provides a step-by-step tutorial for creating an Ubuntu-based virtual machine (VM), which will be used throughout the book in examples. The Ubuntu VM is a highly recommended component of this book and can also be used outside of the book for Android forensic cases.

CHAPTER 2

In this chapter, a wide array of Android-supported hardware and device types is covered. Although the hardware compatibility is great for manufacturers, wireless providers, and ultimately consumers, this diversity poses challenges for forensic analysts and security engineers. Understanding the hardware components, device types, and boot process for Android will aid in your overall understanding of Android and assist in both forensic and security investigations.

CHAPTER 3

This chapter covers the various Android releases, the Android software development kit (SDK), the Davlik virtual machine, key components of Android security, and several other concepts core to Android forensics such as the Android debug bridge (adb) and the USB debugging setting. Step-by-step examples include installing the SDK on Linux, OS X, and Windows as well as creating an Android virtual device that can be used to test forensic techniques.

CHAPTER 4

This chapter covers the information needed to understand how data are stored on an Android device. This includes reviewing the methods in which data are stored (shared preferences, files, SQLite, and network) as well as the types of memory used in an Android device such as RAM and the all important NAND flash. The various file systems the reader might encounter in an Android device are also covered in great detail including the YAFFS2, EXT, FAT32/FAT16, and a variety of low-level file systems.

CHAPTER 5

This chapter covers the security of Android devices, data, and apps. A review not only of how data can be exfiltrated from an Android device is covered but also of how an Android device can be used as an active attack vector. After discussing several overarching security concepts, this chapter provides specific advice for three primary audiences: individuals, corporate security, and app developers. As the growth of Android continues, issues of data security will be increasingly important and this chapter provides a thorough and practical fb-non-chapter to this important topic.

CHAPTER 6

This chapter covers specific techniques that are useful in the forensic acquisition of Android devices. After clarifying the different types of acquisitions and providing procedures for handling an Android device, seven different strategies for circumventing a pass code are discussed. Next, techniques and a specific script for acquiring an SD card and, if present, the Embedded MultiMediaCard (eMMC) are covered. Logical acquisition techniques are then covered including ones built into Android and the SDK, a solution free to law enforcement and government agencies called AFLogical, and finally a review of six commercial forensic software packages. Finally, techniques for acquiring a physical image of the NAND flash are described in detail including six strategies for gaining root privileges and the AFPhysical technique developed by viaForensics.

CHAPTER 7

In this final chapter, strategies and specific utilities are provided, which enable a forensic analyst or security engineer to analyze an acquired Android device. Although many of the techniques used in traditional forensic investigations are applicable in Android forensics analysis, the new file system and the underlying hardware characteristics require new techniques. Without these new techniques, little content and value can be extracted from an Android physical acquisition. Beyond providing the background and actual utilities, an overview of Android's directory structure as well as an in-depth analysis of 11 important applications that provide significant data about the device are given. Armed with this knowledge, a forensic analyst or security engineer can investigate any Android device they encounter.

WEBSITE

For companion material including code, programs and updates please visit: http://viaforensics.com/education/android-forensics-mobile-security-book/

About the Author

Andrew Hoog is a computer scientist, certified forensic analyst (GCFA and CCE), computer and mobile forensics researcher, former adjunct professor (assembly language), and cofounder of viaForensics, an innovative digital forensic and security firm. He divides his energies between investigations, forensic software development, and research in digital forensics and security. He also has two patents pending in the areas of forensics and data recovery. He lives in Oak Park, IL, where he enjoys spending time with his family, traveling, great wine, science fiction, and tinkering with geeky gadgets.

About the Technical Editor

John McCash (CompTIA Sec+, GCIH, GAWN, GCFA, EnCE, GREM, SANS Lethal Forensicator) is a 23-year IT veteran. He has specialized in Security for the last 15 years, and Forensics for the last 4 years. McCash has extensive experience in digital forensics, security/system/network administration, and incident response on diverse platforms in very heterogeneous environments. He obtained his BS and MS in CS at Bradley University in 1988. Currently John works for a major telecommunications equipment provider, and is a semiregular contributor to the SANS Forensic Blog.

About the Author

Andrew Hoog is a computer scientist, certified forensic analyst (GCFA and CCE), computer and mobile forensics researcher, former adjunct professor (assembly language), and cofounder of viaForensics, an innovative digital forensic and security firm. He divides his energies between investigations, forensic software development, and research in digital forensics and security. He also has two patents pending in the areas of forensics and data recovery. He lives in Oak Park, IL, where he enjoys spending time with his family, traveling, great wine, science fiction, and tinkering with geeky gadgets.

About the Technical Editor

John McCash (CompTIA Sec+, GCIH, GAWN, GCFA, EnCE, GREM, SANS Lethal Forensicator) is a 23-year IT veteran. He has specialized in Security for the last 15 years, and Forensics for the last 4 years. McCash has extensive experience in digital forensics, security/system/network administration, and incident response on diverse platforms in very heterogeneous environments. He obtained his BS and MS in CS at Bradley University in 1988. Currently John works for a major telecommunications equipment provider, and is a semiregular contributor to the SANS Forensic Blog.

CHAPTER

Android and mobile forensics 1

INFORMATION IN THIS CHAPTER

- Android platform
- Linux, Open source software and forensics
- Android Open Source Project
- Internationalization
- Android Market
- Android forensics

INTRODUCTION

Digital forensics is an exciting, fast-paced field that can have a powerful impact on a variety of situations including internal corporate investigations, civil litigation, criminal investigations, intelligence gathering, and matters involving national security. Mobile forensics, arguably the fastest growing and evolving digital forensic discipline, offers significant opportunities as well as many challenges. While the interesting part of Android forensics involves the acquisition and analysis of data from devices, it is important to have a broad understanding of both the platform and the tools that will be used throughout the investigation. A thorough understanding will assist a forensic examiner or security engineer through the successful investigation and analysis of an Android device.

TIP

Book corrections, updates, and software

All corrections, updates, and even software samples for this book will be maintained online at the following web page:

http://viaforensics.com/education/android-forensics-mobile-security-book/

Please check the web site as over time it will evolve and provide significant and increasing value to the reader. Beyond corrections and updates, some of the software referenced in the book will be available for download.

ANDROID PLATFORM

Android is an open source mobile device platform based on the Linux 2.6 kernel and managed by the Open Handset Alliance, a group of carriers, mobile device and component manufacturers, and software vendors.

Android Forensics. DOI: 10.1016/B978-1-59749-651-3.10001-9

Table 1.1 Total US Smartphone Subscribers, Ages 13+, November 2010

Platform	Share (%) of Smartphone Subscribers
RIM	33.5
Google	26.0
Apple	25.0
Microsoft	9.0
Palm	3.9

Android has made a significant impact on the smartphone market and, consequently, in the area of forensics. Two years and one month after the first Android device was introduced (October 2008), Android became the second largest smartphone platform capturing 26.0% of the 61.5 million US smartphone subscribers (comScore reports, n.d.). Table 1.1 shows the top smartphone platforms as of November 2010, according to comScore, Inc.

But Android's influence extends well beyond the US market. According to Gartner, Inc., the Android operating system (OS) was the second most popular during the third quarter of 2010 and accounted for 25.5% of worldwide smartphone sales (Gartner says, n.d.), as shown in Table 1.2.

According to the web site Google Investor, Google CEO Eric Schmidt reported that over 350,000 Android devices were being activated each day as of February 2011 (Google investor, n.d.). These statistics focus on the smartphone market, which is only one of the many types of Android devices available in the market.

The open source nature of Android has not only established a new direction for the industry, but also has enabled developers, code savvy forensic analysts, and

Table 1.2 Worldwide Smartphone Sales to End Users by Operating System in Third Quarter of 2009–2010 (in Thousands of Units)

Company	Units—3rd Qtr 2010	Market Share (%)—3rd Qtr 2010	Units—3rd Qtr 2009	Market Share (%)—3rd Qtr 2009
Symbian	29,480.1	36.6	18,314.8	44.6
Android	20,500.0	25.5	1424.5	3.5
iOS	13,484.4	16.7	7040.4	17.1
Research in motion	11,908.3	14.8	8522.7	20.7
Microsoft Windows mobile	2247.9	2.8	3259.9	7.9
Linus	1697.1	2.1	1918.5	4.7
Other OS	1214.8	1.5	612.5	1.5
Total	80,532.6	100.0	41,093.3	100.0

(unfortunately) sophisticated criminals to understand the device at the most fundamental level. As the core platform quickly matures and continues to be provided free of charge, carriers and hardware vendors alike can focus their efforts on customizations intended to retain their customers.

History of Android

For over three decades, companies have invested significant resources into research and development of handheld computing devices in the hopes that they would open new markets. As with traditional computers, the hardware components central to building such devices have advanced significantly and now provide a small, though powerful, mobile platform for handheld computers.

A central figure in the development of Android is Andy Rubin whose past employers include robotics firms, Apple, WebTV, and Danger Inc. His previous company, Danger Inc., developed a smartphone and support OS most recognized from the T-Mobile Sidekick. This mobile operating system, DangerOS, was built using Java. It provided a software development kit and had some of the features found in current smartphones. In 2004, Rubin left Danger and tinkered with several new ideas. He again returned to smartphone development and teamed with several engineers from past companies. The company Rubin formed in 2003 was called Android, Inc.

While the team began development, Rubin was actively marketing Android to both potential investors and wireless carriers. One of the companies he spoke with was Google, who subsequently acquired Android in July 2005. The acquisition, combined with new patents and services involving mobile and a large bid for wireless spectrum, fueled significant speculation that Google was developing their own smartphone and perhaps was aiming to be a full wireless carrier.

However, on November 5, 2007, Andy Rubin announced a more ambitious plan on the official Google blog (Official Google blog, n.d.):

> *Android is the first truly open and comprehensive platform for mobile devices. It includes an operating system, user-interface and applications—all of the software to run a mobile phone, but without the proprietary obstacles that have hindered mobile innovation. We have developed Android in cooperation with the Open Handset Alliance, which consists of more than 30 technology and mobile leaders including Motorola, Qualcomm, HTC and T-Mobile. Through deep partnerships with carriers, device manufacturers, developers, and others, we hope to enable an open ecosystem for the mobile world by creating a standard, open mobile software platform. We think the result will ultimately be a better and faster pace for innovation that will give mobile customers unforeseen applications and capabilities.*

One week later, Google released an early look at the Android software development kit (SDK) to developers. This allowed Google to create the first Android Developer Challenge, which ran from January 2008 through April 2008. Google set

aside $1,000,000 to reward the most innovative Android apps. The top 50 apps are available for review at http://code.google.com/android/adc/adc_gallery/.

In August 2008, Google announced the availability of the Android Market where developers could upload their apps for mobile device owners to browse and install. The initial release did not support paid apps. However, that feature was added in early 2009. Finally, October 2008 marked both the official release of the Android Open Source Project (AOSP) (Bort, n.d.) and the first publicly available Android smartphone, the T-Mobile G1.

Since inception, the Android ecosystem has grown significantly and is comprised of diverse groups of contributors. Table 1.3 summarizes significant milestones for the Android platform.

Open Handset Alliance

The Open Handset Alliance (OHA) is a collaboration among mobile technology companies including wireless carriers, handset and component manufacturers, software developers, and other support and integration companies. The alliance, established on November 5, 2007, originally had 34 members. However, by January 2011 there were nearly 80 members.

The OHA is committed "to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience" (Alliance FAQ, n.d.) with the primary focus on the coordination, development, and release of Android devices. Google is the driving force behind both the OHA and AOSP. Some have complained that the alliance is simply a marketing technique that offers little value to the members or consumers. However, new members have joined throughout 2010 and the OHA will undoubtedly continue well into the future. The members, as of

Table 1.3 Android Milestones

Date	Event
July 1, 2005	Google acquires Android, Inc.
November 12, 2007	Android launched
August 28, 2008	Android Market announced
September 23, 2008	Android 1.0 platform released
October 21, 2008	Android released as open source software
February 13, 2009	Android Market: USA takes paid apps
March 12, 2009	Android Market: UK takes paid apps
April 15, 2009	Android 1.5 (Cupcake) platform released
September 16, 2009	Android 1.6 (Donut) platform released
October 5, 2009	Android 2.0/2.1 (Eclair) platform released
May 20, 2010	Android 2.2 (Froyo) platform released
May 23, 2010	Android 2.2. for Nexus One phones released
December 6, 2010	Android 2.3 (Gingerbread) platform released
February 2, 2011	Android 3.0 (Honeycomb) preview released

February 3, 2011, listed in Table 1.4, are grouped by mobile operators, handset manufacturers, semiconductor companies, software companies, and commercialization companies (Alliance members, n.d.).

Android Features

While we explore the various Android device types more in the next chapter, there are several features common to most Android devices that we can discuss here.

First, Android was engineered from the beginning to be online, whether using cellular networks such as Global System for Mobile Communications and Code Division Multiple Access (GSM/CDMA) or wireless networks (Wi-Fi). Regardless of the venue, the ability to be online is a core feature of any Android device. Many of the devices are indeed smartphones and thus support sending and receiving phone calls, text messages, and other services found on cellular networks. Interacting with the device is typically via a touch screen, but many devices also allow for keyboards or other buttons, which support user interaction.

A second core feature of Android devices is the ability to download and install applications (apps) from the Android Market. This is a primary feature to many users because it allows them to extend the functionality of the device. These apps also typically happen to be a rich source of information for forensic analysts.

The final core feature is the ability for users to store their data on the devices. This, of course, is the basis for the forensics work covered in detail in this book. Most Android devices come with some on-device storage using flash (NAND) memory as well as an external SD card that is portable and intended to store larger amounts of data. Some recent HTC devices are now shipping with an emulated SD card which is a separate USB device ID mapped to the NAND and presented as an SD card. The emulated SD cards are typically formatted with Microsoft's FAT32 file system.

Supported Cellular Networks

As smartphones are the largest category of Android devices, it is important to understand the various cellular technologies Android currently supports.

The first Android device, the HTC DREA100 or T-Mobile G1, was a Global System for Mobile Communications (GSM) phone. GSM is the most widely used and supported cellular system with excellent support throughout the world. Major wireless providers in the United States that support GSM include AT&T and T-Mobile. The GSM system leverages a subscriber identity module (SIM) or universal subscriber identity module (USIM) to identify the user to the cellular network.

The next cellular system supported by Android is the Code Division Multiple Access, often referred to as CDMA. CDMA is the technique used to encode and send the voice, data, and control signals used by a CDMA phone. It is popular in the United States, but less so around the world. In the United States, the primary technology standard used is called CDMA2000. Major carriers include Verizon Wireless, Sprint, U.S. Cellular, and Cricket Communications.

The final cellular system supported by Android is the Integrated Digital Enhanced Network, or iDEN, whose primary attraction is its support of the

Table 1.4 Open Handset Alliance Members

Company Type	Companies
Mobile operators	• Bouygues Telecom • China Mobile Communications Corporation • China Telecommunications Corporation • China Unicom • KDDI Corporation • NTT DoCoMo, Inc. • Softbank Mobile Corp. • Sprint Nextel • T-Mobile • Telecom Italia • Telefónica • Telus • Vodafone
Handset manufacturers	• Acer Inc. • Alcatel Mobile Phones • ASUSTeK Computer Inc. • CCI • Dell • FIH • Garmin • Haier Telecom (Qingdao) Co., Ltd • HTC Corporation • Huawei Technologies • Kyocera • Lenovo Mobile Communication Technology Ltd • LG • Motorola • NEC Corporation • Samsung Electronics • Sharp Corporation • Sony Ericsson • Toshiba Corporation • ZTE Corporation
Semiconductor companies	• AKM Semiconductor Inc. • Audience • ARM • Atheros Communications • Audience • Broadcom Corporation • CSR Plc. • Cypress Semiconductor Corp. • Freescale Semiconductor • Gemalto • Intel Corporation • Marvell Semiconductor, Inc. • MediaTek, Inc. • MIPS Technologies, Inc.

Table 1.4 Open Handset Alliance Members *(Continued)*

Company Type	Companies
	• Nvidia Corporation • Qualcomm • Renesas Electronics Corp. • ST-Ericsson • Synaptics, Inc. • Texas Instruments Inc. • Via Telecom
Software companies	• Access Co., Ltd • Ascender Corp. • Cooliris, Inc. • eBay Inc. • Google Inc. • LivingImage Ltd • Myriad • Motoya Co., Ltd • Nuance Communications, Inc. • NXP Software • OMRON Software Co., Ltd • PacketVideo (PV) • SkyPop • SONiVOX • SVOX • VisualOn Inc.
Commercialization companies	• Accenture • Aplix Corp. • Borqs • L&T Infotech • Noser Engineering Inc. • Sasken Communication Technologies Limited • SQL Start International Inc. • TAT The Astonishing Tribe AB • Teleca AB • Wind River Systems • Wipro Technologies

popular push-to-talk (PTT) feature. In the United States, the only large carrier supporting iDEN is Sprint Nextel (who also owns Boost Mobile). Motorola, the developer of iDEN, also developed the Motorola i1, the first Android phone supporting iDEN.

Google's Strategy

Android is clearly a powerful mobile device platform which costs an enormous amount in development. So why did Google give Android away for free?

The answer starts with Google's clearly defined mission (Corporate information: about, n.d.):

> *Google's mission is to organize the world's information and make it universally accessible and useful.*

Cell phones are the most popular consumer device, numbering over 4 billion, so by providing an advanced mobile stack at no cost, Google believes they are fulfilling the universally accessible portion of their mission. But, obviously there must still be some benefit for Google. When more people are online, more people use search, which ultimately drives ad revenue—Google's primary source of income. In a March 2009 interview, Andy Rubin explained:

> *Google has a great business model around advertising, and there's a natural connection between open source and the advertising business model. Open source is basically a distribution strategy, it's completely eliminating the barrier to entry for adoption.*
>
> **(Krazit, n.d.)**

One of the criticisms of Android is that the market is now highly fragmented with different versions and variations of Android—a direct result of how Google releases Android to the manufacturers. This is in contrast to other devices, such as the iPhone where Apple has total control over the hardware and OS and significant influence over third-party application. Rubin defends this model, however. In the same interview, Rubin further commented on this aspect (Krazit, n.d.):

> *Controlling the whole device is great, (but) we're talking about 4 billion handsets. When you control the whole device the ability to innovate rapidly is pretty limited when it's coming from a single vendor. You can have spurts of innovation. You can nail the enterprise, nail certain interface techniques, or you can nail the Web-in-the-handset business, but you can't do everything. You're always going to be in some niche. What we're talking about is getting out of a niche and giving people access to the Internet in the way they expect the Internet to be accessed. I don't want to create some derivative of the Internet, I don't want to just take a slice of the Internet, I don't want to be in the corner somewhere with some dumbed-down version of the Internet, I want to be on the Internet.*

So by creating a mobile OS that meets the demands of the consumer as well as the needs of the manufacturers and wireless carriers, Google has an excellent distribution platform for their revenue-generating search and advertising business.

Apps

One important way by which Android supports innovation beyond the core mobile stack is by enabling the development and distribution of third-party apps on Android. As of January 2011, over 200,000 Android apps have been developed. This, of course, is similar to the strategy Apple developed. However, there are key differences in their approach. Apple maintains tight control over their App Store,

requiring developers to submit to a sometimes lengthy review process and providing Apple with the final approval for an app. Apps can be denied based on a number of criteria, most notably if they contain any content Apple feels is objectionable. Google, on the other hand, requires very little review to publish an app in the Android Market. While Google has the ability to ban a developer, remove an app from the Android Market, and even remotely uninstall apps from Android devices, in general their approach to app management is hands off.

Nexus Phones

In January 2010, Google released its own smartphone, the Nexus One (N1) shown in Fig. 1.1. The N1 was developed by HTC and, by all accounts, was an ideal model for how manufacturers should develop their phones. The processor was extremely fast (1 GHz), it was running the latest version of Android, and it had innovations such as three microphones which survey background noise and blend your voice to create the most clear conversation possible.

The N1 was sold directly by Google and was sold unlocked—a move many analysts saw as a direct challenge to the carrier lock-in model where customers must sign a two-year agreement to get a discount on the device. The N1 was also available through T-Mobile for a reduced price, provided the user signs an extended contract. In the end, the sales for the N1 were not overwhelming and there was speculation that Google failed in their implementation (Fig. 1.1).

FIGURE 1.1

Google Nexus One by HTC.

FIGURE 1.2

Google Nexus S by Samsung.

However, at the time, Google was also trying to demonstrate how they believed an Android phone should be released and maintained. To the surprise of many, one year later Google released the Nexus S manufactured by Samsung, shown in Fig. 1.2. One interesting feature of the Nexus S was that it ran on Android 2.3 that allowed the native ability to make Voice over IP (VoIP) phone calls. If a device has a data connection, whether it is Wi-Fi.com or some other network, then it can send and receive phone calls using any number of popular VoIP services. In the United States, the phone was sold only through Best Buy stores and service was available through T-Mobile (Fig. 1.2).

It is unclear what Google's overall goals are with the Nexus line of smartphones. However, it is clear they intend to release Google phones and eventually may offer consumers a new flexibility in how they purchase and use smartphones.

LINUX, OPEN SOURCE SOFTWARE, AND FORENSICS

Open source software has had a tremendous impact on the digital forensics discipline. Forensic tools that are released as free open source software have tremendous advantages over closed source solutions including the following:

- The ability to review source code and understand exact steps taken
- The ability to improve the software and share enhancements with entire community
- The price

While many of the free, open source software packages do not offer a commercial support model, some companies specialize in providing support. For example, Red Hat has built a significant business providing support and services for the Linux OS. In addition, the maintainers of many free, open source software packages are generally very accessible and responsive to inquiries and can often provide far superior support as they directly maintain the software.

The most significant and important example of free, open source software is the Linux OS. Linux is not only a critical component of Android but can also be used as a powerful forensic tool.

Brief History of Linux

There have been many books written about Linux and dedicating only one section to such an important OS is difficult. There are also many fantastic online resources for Linux some of which focus on Linux as a forensic tool.

In 1991, Linus Torvalds was a University of Helsinki student when he decided to develop a terminal emulator that he could use to connect to the University's systems. The code was developed specifically for his computer, which had an Intel 386 processor. After he completed the initial development, he realized that code could actually form the basis of an OS and he posted the following famous messages on the Usenet newsgroup comp.os.minix (Torvalds, 1991):

```
Path: gmdzi!unido!mcsun!news.funet.fi!hydra!klaava!torvalds
From: torva...@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Free minix-like kernel sources for 386-AT
Keywords: 386, preliminary version
Message-ID: <1991Oct5.054106.4647@klaava.Helsinki.FI>
Date: 5 Oct 91 05:41:06 GMT
Organization: University of Helsinki
Lines: 55

Do you pine for the nice days of minix-1.1, when men were men and wrote
their own device drivers? Are you without a nice project and just dying
to cut your teeth on a OS you can try to modify for your needs? Are you
finding it frustrating when everything works on minix? No more all-
nighters to get a nifty program working? Then this post might be just
for you :-)

As I mentioned a month(?) ago, I'm working on a free version of a
minix-lookalike for AT-386 computers.  It has finally reached the stage
where it's even usable (though may not be depending on what you want),
and I am willing to put out the sources for wider distribution.  It is
just version 0.02 (+1 (very small) patch already), but I've successfully
run bash/gcc/gnu-make/gnu-sed/compress etc under it.

<snip>

I can (well, almost) hear you asking yourselves "why?".  Hurd will be
out in a year (or two, or next month, who knows), and I've already got
minix.  This is a program for hackers by a hacker.  I've enjouyed doing
it, and somebody might enjoy looking at it and even modifying it for
their own needs.  It is still small enough to understand, use and
modify, and I'm looking forward to any comments you might have.

<snip>
```

Reading this post, the mentality of many avid Linux users is captured in the desire to understand, modify, create, and otherwise tinker with complex systems (often referred to as a hacker mentality). The newsgroup Linus posted on was for the Minix OS, which at the time was the OS of choice for many people wanting to test and develop a Unix-like OS. However, there were licensing restrictions as well as technical limitations of Minix that Linus wanted to overcome.

Over nearly 20 years, Linux has matured significantly and is used on many PCs, servers, and now mobile devices. There are literally thousands of powerful tools available as well as complete development environments for many programming languages. The are many distributions that focus on different needs including servers, workstations, laptops, embedded devices, security suites, and many more.

Installing Linux in VirtualBox

Linux is a truly amazing OS and we will use its power throughout this book in examples intended for the reader to follow along and complete. All examples in this book are performed on an Ubuntu 10.10 64-bit desktop install running as a virtual machine (VM). While the virtual machine software from several vendors is compatible (including VMWare Fusion running on Mac OS X), this book is focused on options that are free, open, or both. In this instance, VirtualBox is both open source software and freely available.

NOTE

This Ubuntu VM will be used extensively through the book for all examples. Subsequent chapters will build upon this base install by adding more tools and scripts. Readers are encouraged to create this Ubuntu VM and follow along with all examples to maximize knowledge. The Ubuntu VM can be used directly for Android forensic cases.

VirtualBox is now owned by Oracle and is distributed under the GPLv2 license. There is a section on Oracle's web site that addresses frequently asked questions about licensing.

You can download VirtualBox for many operating systems including Microsoft Windows, Mac OS X, and Linux (2.4 and 2.6) from http://www.virtualbox.org/. After you install VirtualBox, you will see the Oracle VM VirtualBox Manager, shown in Fig. 1.3, where you create and manage new VMs.

When you create the new VM, make sure you have enough hard drive space (at least 20 GB is recommended) and as much RAM as you can spare. For the Android build, Google recommends at least 1536 MB (1.5 GB) (Get Android source code, n.d.).

Using the VirtualBox Manager graphical user interface (GUI) to set the new virtual machine is straightforward. However, if you have access an Ubuntu Linux 64-bit workstation or server, but do not have the ability to run desktop

FIGURE 1.3

Oracle VM VirtualBox Manager for OS X.

applications, here are the steps you can follow to setup, configure, and run the new VM (VirtualBox 3.2.10).

From an ssh session, it is best to use the program "screen" so that if you lose connection to the server, your VM remains active. Then, follow these steps:

```
mkdir -p ~/vbox
cd ~/vbox
wget http://ubuntu.mirrors.pair.com/releases/maverick/ubuntu-10.10 -desktop
-amd64.iso

VBoxManage createvm -name af-book-vm -ostype Ubuntu -register

VBoxManage modifyvm af-book-vm --memory 1536 --acpi on --boot1 dvd \
--nic1 bridged --usb on --usbehci on --vrdp on --vrdpport 3392 \
--clipboard bidirectional --pae on --hwvirtex on --hwvirtexexcl on
  --vtxvpid on \
--nestedpaging on --largepages on
```

```
VBoxManage modifyvm af-book-vm --bridgeadapter1 eth0

VBoxManage storagectl af-book-vm --name "IDE Controller" --add ide

VBoxManage createvdi --filename ~/vbox/af-book-vm.vdi \
--size 20000 --register

VBoxManage storageattach af-book-vm --storagectl "IDE Controller" \
--port 0 --device 0 --type hdd --medium ~/vbox/af-book-vm.vdi

VBoxManage storageattach af-book-vm --storagectl "IDE Controller" \
--port 1 --device 0 --type dvddrive --medium ~/vbox/ubuntu-10.10-desktop
-i386.iso

VBoxHeadless -startvm af-book-vm -p 3392 &

#need to eject DVD, the restart
VBoxManage storageattach af-book-vm --storagectl "IDE Controller" --port 1 \
--device 0 --type dvddrive --medium none

#restart the virtual machine
VBoxHeadless -startvm af-book-vm -p 3392
```

At this point, the VM will start up and you can access the install using any Remote Desktop Protocol (RDP) viewer such as Remote Desktop Connection on Windows, rdesktop on Linux, or Microsoft's Remote Desktop Connection Client for Mac. To access the above session, you would connect to <host server's IP:3392>. From there, follow the install until it is time to reboot.

If you shutdown or reboot the VBoxHeadless session ends; you can simply issue the command again to start the server backup. Then, RDP back into the machine and install openssh server so that we can use ssh instead of the less efficient RDP:

```
sudo apt-get install openssh-server
```

Now you can find the virtual machine's IP address by running ifconfig and looking at the "inet addr" for eth0. You can use your favorite ssh program (if on Windows, try Putty for a great, free client) and ssh into the virtual machine.

The Sleuth Kit (TSK)

Brian Carrier has an excellent open source forensic toolkit called The Sleuth Kit (TSK), which will be discussed in this section. Examples throughout this book will leverage TSK extensively. Brian developed and continues to maintain TSK and provides an enormous service to our industry. If you are not familiar with TSK, visit the web site at http://sleuthkit.org/ and consider using the programs. There is quite a bit of information on TSK's web site as well as many forensic blogs and books. If you are going to follow the examples in this book, you should install TSK on the Linux workstation with the following command:

```
sudo apt-get install sleuthkit
```

Hopefully others can follow in Brian's footsteps and provide such important toolkits and service to the forensic community.

Disable Automount

It is critical that forensic workstations do not have automount enabled which, as the name infers, will automatically mount a file system when one is found on a device connected. The option to disable automount in Ubuntu is done per user, so if the workstation will have more than one user account, please make sure you change each of them:

```
gconf-editor
```

Then navigate to apps > nautilus > preferences and ensure the "media_automount" and "media_automount_open" options are unchecked as illustrated in Fig. 1.4.

You can then close the Gnome Configuration editor. Now, automount is disabled. For typical users, this is more work. However, for a forensic analyst, it is an absolute necessity (as is the use of hardware write blockers).

Linux and Forensics—Basic Commands

Before we setup and configure a Linux forensic workstation, it is helpful to provide an overview of Linux's relevance to forensics. A Linux workstation is a powerful tool for forensic investigation due to the wide support for many file systems, the advanced tools available, and the ability to develop and compile source code. However, since many examiners are not familiar with Linux, the following sections provide a breakdown of some of the more common Linux commands including a description of the command, its general usage, and one or more examples of how the command can be applied.

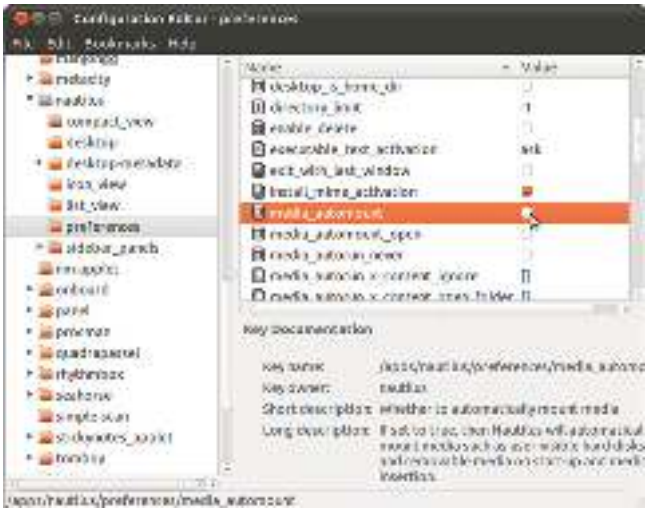

FIGURE 1.4

Disable automount on Ubuntu.