

iOS应用安全攻防 (影印版)



Hacking and Securing iOS Applications

O'REILLY®

东南大学出版社

Jonathan Zdziarski 著

iOS应用安全攻防 (影印版)

Hacking and Securing iOS Applications

*Jonathan Zdziarski*著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目 (CIP) 数据

iOS 应用安全攻防: 英文/(美)扎德尔斯基 (Zdziarski, J.)
著. —影印本. —南京: 东南大学出版社, 2012.6
书名原文: Hacking and Securing iOS Applications
ISBN 978-7-5641-3446-4

I. ① i… II. ① 扎… III. ① C 语言—程序设计—英文
② 计算机网络—安全技术—英文 IV. ① TP312 ② TP393.08

中国版本图书馆 CIP 数据核字 (2012) 第 089056 号

江苏省版权局著作权合同登记

图字: 10-2012-157 号

©2012 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2012. Authorized reprint of the original English edition, 2012 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2012。

英文影印版由东南大学出版社出版 2012。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

iOS 应用安全攻防 (影印版)

出版发行: 东南大学出版社

地 址: 南京四牌楼 2 号 邮编: 210096

出 版 人: 江建中

网 址: <http://www.seupress.com>

电子邮件: press@seupress.com

印 刷: 扬中市印刷有限公司

开 本: 787 毫米 × 980 毫米 16 开本

印 张: 22.25

字 数: 436 千字

版 次: 2012 年 6 月第 1 版

印 次: 2012 年 6 月第 1 次印刷

书 号: ISBN 978-7-5641-3446-4

定 价: 59.00 元 (册)

本社图书若有印装质量问题, 请直接与营销部联系。电话 (传真): 025-83791830

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是书籍出版、在线服务还是面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔视野，并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

Steve: The coolest cat. We loved the chase!
- Hackers and tinkerers everywhere

Preface

Data is stolen; this is no uncommon occurrence. The electronic information age has made the theft of data a very lucrative occupation. Whether it's phishing scams or large-scale data breaches, criminals stand to greatly benefit from electronic crimes, making their investment well worth the risk. When I say that this occurrence is not uncommon, my goal isn't to be dismissive, but rather to alarm you. The chances that your company's applications will be vulnerable to attack are very high. Hackers of the criminal variety have an arsenal of tools at their disposal to reverse engineer, trace, and even manipulate applications in ways that most programmers aren't aware. Even many encryption implementations are weak, and a good hacker can penetrate these and other layers that, so many times, present only a false sense of security to the application's developers.

Take everything hackers collectively know about security vulnerability and apply it to a device that is constantly connected to a public network, wrapped up in a form factor that can fit in your pocket and is frequently left at bars. Your company's applications, and the data they protect, are now subject to simpler forms of theft such as pickpocketing, file copies that can take as little as a few minutes alone with a device, or malicious injection of spyware and root kits—all of which can be performed as the device's owner reaches for another drink. One way or another, software on a mobile platform can be easily stolen and later attacked at the criminal's leisure, sometimes without the device's owner even knowing, and sometimes without physical access to the device.

This book is designed to demonstrate many of the techniques black hats use to steal data and manipulate software in an attempt to show you, the developer, how to avoid many all too common mistakes that leave your applications exposed to easy attacks. These attacks are not necessarily limited to just the theft of data from the device, but can sometimes even lead to much more nefarious attacks. In this book, you'll see an example of how some credit card payment processing applications can be breached, allowing a criminal to not only expose the credit card data stored on the device, but also to manipulate the application to grant him huge credit card refunds for purchases that he didn't make, paid straight from the merchant's stolen account. You'll see many more examples, too, of exploits that have made mobile applications not just a data risk, but downright dangerous to those using them. The reader will also gain an understanding of how these attacks are executed, and many examples and demonstrations

of how to code more securely in ways that won't leave applications exposed to such attacks.

Audience of This Book

This book is geared toward iOS developers looking to design secure applications. This is not necessarily limited to government or financial applications, but may also pertain to applications with assets or other features that the developer is looking to protect. You'll need a solid foundation of Objective-C coding on iOS to understand a majority of this book. A further understanding of C or assembly language will also help, but is not required.

While this book primarily focuses on iOS, much of the material can also be applied directly to the Mac OS X desktop. Given that both environments run an Objective-C environment and share many of the same tools, you'll find much of this book can be used to expose vulnerabilities in your company's desktop applications as well.

Organization of the Material

This book is split into two halves. The first half discusses hacking and exposes the many vulnerabilities in iOS and iOS applications, while the second half covers techniques to better secure applications.

Chapter 1 explains the core problem with mobile security, and outlines common myths, misconceptions, and overall flaws in many developers' ways of thinking about security.

Chapter 2 introduces the reader to many techniques of compromising an iOS device, including jailbreaking. The reader will learn how to build and inject custom code into an iOS device using popular jailbreaking techniques and custom RAM disks.

Chapter 3 demonstrates how the filesystem of an iOS device can be stolen in minutes, and how developers can't rely solely on a manufacturer's disk encryption. You'll also learn about some common social engineering practices that secure access to a device without the owner's knowledge.

Chapter 4 covers the forensic data left by the operating system, and what kind of information one can steal from a device.

Chapter 5 explains how iOS's keychain encryption and data protection encryption can be defeated, and the inherent problems of each.

Chapter 6 demonstrates how the HFS journal can be scraped for deleted files, and provides examples of how to securely delete files so they cannot be recovered.

Chapter 7 introduces you to tools for spying on and manipulating the runtime environment, and demonstrates how black hat hackers can manipulate your application's objects, variables, and methods to bypass many layers of security.

Chapter 8 introduces you to tools and approaches for disassembling and debugging your application, injecting malicious code, and performing low-level attacks using a number of techniques.

Chapter 9 illustrates some of the tools used to hijack SSL sessions, and how to protect your application from falling victim to these attacks.

Chapter 10 elaborates on security and describes additional methods to protect your data with proper encryption techniques.

Chapter 11 explains how to help prevent forensic data leakage by designing your application to leave fewer traces of information.

Chapter 12 explains many best practices to increase the complexity needed for an attack on your applications.

Chapter 13 explains techniques used to detect when an application is running on a device jailbroken with some of the popular jailbreaking tools available.

Chapter 14 wraps up the book and explains how important it is to understand and strategize like your adversary.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.


We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Hacking and Securing iOS Applications* by Jonathan Zdziarski. Copyright 2012 Jonathan Zdziarski, (ISBN 9781449318741)."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Legal Disclaimer

The technologies discussed in this publication, the limitations on these technologies that the technology and content owners seek to impose, and the laws actually limiting the use of these technologies are constantly changing. Thus, some of the hacks described in this publication may not work, may cause unintended harm to equipment or systems on which they are used, or may be inconsistent with applicable law or user agreements. Your use of these projects is at your own risk, and O'Reilly Media, Inc. disclaims responsibility for any damage or expense resulting from their use. In any event, you should take care that your use of these projects does not violate any applicable laws, including copyright laws.

Safari® Books Online

 Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at <http://my.safaribooksonline.com>.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

<http://www.oreilly.com/catalog/9781449318741>

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

Table of Contents

Preface	xi
1. Everything You Know Is Wrong	1
The Myth of a Monoculture	2
The iOS Security Model	5
Components of the iOS Security Model	5
Storing the Key with the Lock	7
Passcodes Equate to Weak Security	9
Forensic Data Trumps Encryption	10
External Data Is at Risk, Too	11
Hijacking Traffic	11
Data Can Be Stolen...Quickly	12
Trust No One, Not Even Your Application	13
Physical Access Is Optional	14
Summary	15

Part I. Hacking

2. The Basics of Compromising iOS	19
Why It's Important to Learn How to Break Into a Device	19
Jailbreaking Explained	20
Developer Tools	20
End User Jailbreaks	23
Jailbreaking an iPhone	23
DFU Mode	25
Tethered Versus Untethered	26
Compromising Devices and Injecting Code	26
Building Custom Code	28
Analyzing Your Binary	29
Testing Your Binary	32
Daemonizing Code	34

Deploying Malicious Code with a Tar Archive	37
Deploying Malicious Code with a RAM Disk	38
Exercises	50
Summary	51
3. Stealing the Filesystem	53
Full Disk Encryption	53
Solid State NAND	54
Disk Encryption	54
Where iOS Disk Encryption Has Failed You	56
Copying the Live Filesystem	56
The DataTheft Payload	57
Customizing launchd	65
Preparing the RAM disk	70
Imaging the Filesystem	71
Copying the Raw Filesystem	73
The RawTheft Payload	73
Customizing launchd	78
Preparing the RAM disk	79
Imaging the Filesystem	79
Exercises	80
The Role of Social Engineering	81
Disabled Device Decoy	81
Deactivated Device Decoy	82
Malware Enabled Decoy	83
Password Engineering Application	84
Summary	84
4. Forensic Trace and Data Leakage	87
Extracting Image Geotags	88
Consolidated GPS Cache	89
SQLite Databases	91
Connecting to a Database	91
SQLite Built-in Commands	92
Issuing SQL Queries	93
Important Database Files	93
Address Book Contacts	93
Address Book Images	95
Google Maps Data	97
Calendar Events	101
Call History	103
Email Database	103
Notes	105

Photo Metadata	105
SMS Messages	105
Safari Bookmarks	106
SMS Spotlight Cache	106
Safari Web Caches	107
Web Application Cache	107
WebKit Storage	107
Voicemail	107
Reverse Engineering Remnant Database Fields	108
SMS Drafts	110
Property Lists	110
Important Property List Files	111
Other Important Files	115
Summary	117
5. Defeating Encryption	119
Sogeti's Data Protection Tools	119
Installing Data Protection Tools	120
Building the Brute Forcer	120
Building Needed Python Libraries	121
Extracting Encryption Keys	122
The KeyTheft Payload	122
Customizing Launchd	123
Preparing the RAM disk	124
Preparing the Kernel	125
Executing the Brute Force	125
Decrypting the Keychain	128
Decrypting Raw Disk	130
Decrypting iTunes Backups	131
Defeating Encryption Through Spyware	132
The SpyTheft Payload	133
Daemonizing spyd	137
Customizing Launchd	137
Preparing the RAM disk	138
Executing the Payload	139
Exercises	139
Summary	140
6. Unobliterating Files	141
Scraping the HFS Journal	142
Carving Empty Space	144
Commonly Recovered Data	144
Application Screenshots	144

Deleted Property Lists	146
Deleted Voicemail and Voice Recordings	146
Deleted Keyboard Cache	146
Photos and Other Personal Information	146
Summary	147
7. Manipulating the Runtime	149
Analyzing Binaries	150
The Mach-O Format	150
Introduction to class-dump-z	154
Symbol Tables	155
Encrypted Binaries	156
Calculating Offsets	158
Dumping Memory	159
Copy Decrypted Code Back to the File	161
Resetting the cryptid	161
Abusing the Runtime with Cycrypt	163
Installing Cycrypt	164
Using Cycrypt	164
Breaking Simple Locks	166
Replacing Methods	172
Trawling for Data	174
Logging Data	177
More Serious Implications	177
Exercises	185
SpringBoard Animations	185
Call Tapping...Kind Of	186
Making Screen Shots	187
Summary	187
8. Abusing the Runtime Library	189
Breaking Objective-C Down	189
Instance Variables	191
Methods	191
Method Cache	192
Disassembling and Debugging	193
Eavesdropping	197
The Underlying Objective-C Framework	199
Interfacing with Objective-C	201
Malicious Code Injection	203
The CodeTheft Payload	203
Injection Using a Debugger	204
Injection Using Dynamic Linker Attack	206

Full Device Infection	207
Summary	208
9. Hijacking Traffic	209
APN Hijacking	209
Payload Delivery	212
Removal	214
Simple Proxy Setup	214
Attacking SSL	215
SSLStrip	216
Paros Proxy	217
Browser Warnings	219
Attacking Application-Level SSL Validation	222
The SSLTheft Payload	222
Hijacking Foundation HTTP Classes	228
The POSTTheft Payload	228
Analyzing Data	231
Driftnet	232
Building	233
Running	234
Exercises	234
Summary	236

Part II. Securing

10. Implementing Encryption	241
Password Strength	241
Beware Random Password Generators	244
Introduction to Common Crypto	244
Stateless Operations	245
Stateful Encryption	249
Master Key Encryption	252
Geo-Encryption	257
Geo-Encryption with Passphrase	260
Split Server-Side Keys	262
Securing Memory	264
Wiping Memory	265
Public Key Cryptography	266
Exercises	270
11. Counter Forensics	273
Secure File Wiping	273

DOD 5220.22-M Wiping	274
Objective-C	275
Wiping SQLite Records	277
Keyboard Cache	282
Randomizing PIN Digits	283
Application Screenshots	285
12. Securing the Runtime	287
Tamper Response	287
Wipe User Data	288
Disable Network Access	289
Report Home	289
Enable Logging	289
False Contacts and Kill Switches	290
Process Trace Checking	291
Blocking Debuggers	293
Runtime Class Integrity Checks	295
Validating Address Space	295
Inline Functions	306
Complicating Disassembly	312
Optimization Flags	313
Stripping	317
They're Fun! They Roll! -funroll-loops	323
Exercises	326
13. Jailbreak Detection	327
Sandbox Integrity Check	328
Filesystem Tests	329
Existence of Jailbreak Files	329
Size of /etc/fstab	331
Evidence of Symbolic Linking	331
Page Execution Check	332
14. Next Steps	333
Thinking Like an Attacker	333
Other Reverse Engineering Tools	333
Security Versus Code Management	334
A Flexible Approach to Security	335
Other Great Books	336

Everything You Know Is Wrong

Secure coding is about increasing the complexity demanded for an attack against the application to succeed. No application can ever be truly secure. With the right resources and time, any application, including those utilizing strong encryption, can be broken. The determination of how secure an application is depends on the trade-off between the time and complexity of an attack versus the value of the resource when it is breached. For example, a list of stolen credit card numbers is very useful to an attacker—if that list is only 10 minutes old. After 24 hours, the value of this data becomes increasingly diminished, and after a week it is virtually worthless. Securing an application is about increasing the complexity needed to attack it, so that the resource—when breached—will have a significantly diminished value to the attacker. Increasing the complexity needed for an attack also reduces the pool size of potential attackers. That is, attacks requiring higher skillsets reduce the number of people capable of attacking your application.

The term *mobile security*, as used in the marketplace today, has fallen out of sync with this premise. For many, security has become less about attack complexity and more about reducing overhead by depending on a monoculture to provide secure interfaces. As it pertains to iOS, this monoculture consists of a common set of code classes from the manufacturer to provide password encryption routines, user interface security, file system encryption, and so on. In spite of the many great advancements in security that Apple has made, the overall dependence on the operating system has unfortunately had the opposite effect on the security of applications: it has made them less complex, and given the keys out for every single application when the monoculture is breached.

We use words like “encryption” as if they are inherently secure solutions to the decades-old problem of data theft, yet countless millions of seemingly encrypted credit card numbers, social security numbers, and other personal records have been stolen over the years. Application developers are taught to write secure applications, but never told that they can’t even trust their own runtime. Bolting on SSL has become the norm, even though a number of attacks against SSL have been successfully used to rip off credentials and later to empty bank accounts. Everything we are taught about security is wrong, because the implementation is usually wrong. Even well thought out implementations,