Broadview
www.broadview.com.cn

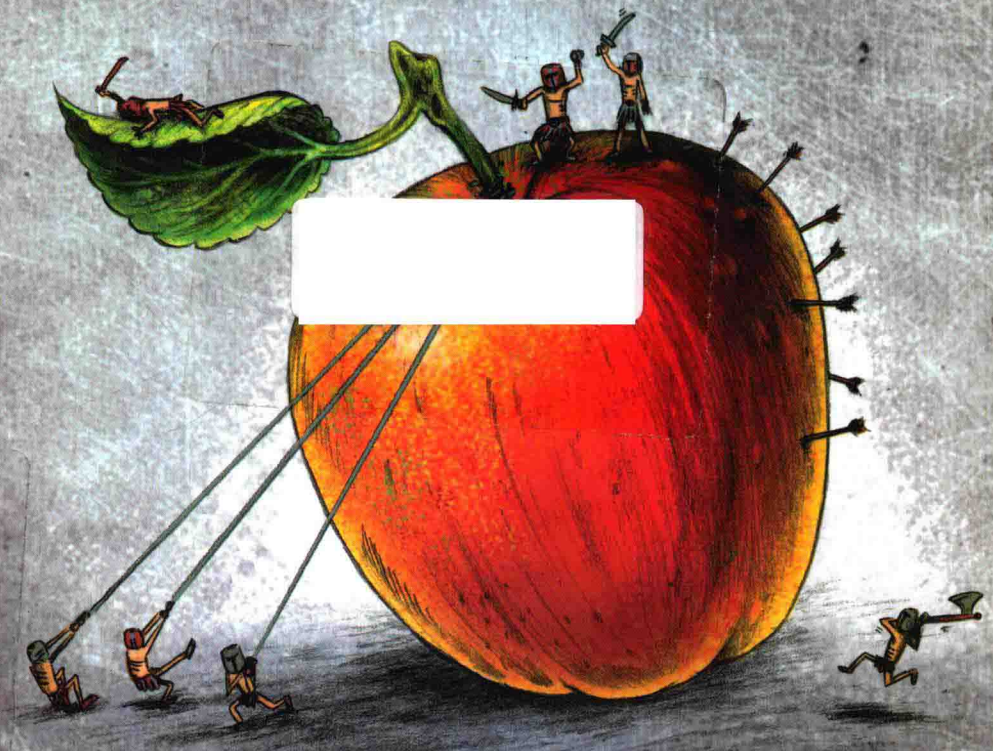
no starch press

iOS应用安全权威指南

【美】David Thiel 著
程伟 马超 李俊阳 译

iOS Application Security

The Definitive Guide for Hackers and Developers



中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

iOS应用安全权威指南

iOS Application Security

The Definitive Guide for Hackers and Developers

【美】David Thiel 著

程伟 马超 李俊阳 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

对于所有希望保护用户免受恶意攻击的开发者来说，消除 iOS 应用当中的安全漏洞至关重要。在本书中，移动端安全专家 David Thiel 向你揭示了那些会导致严重安全问题的常见 iOS 编码漏洞，并阐述了找到并修复这些漏洞的方法。

避免在应用的安全漏洞方面出现重大纰漏很重要。无论是需要加强应用的防御能力，还是要在他人的代码当中寻找安全漏洞，本书都能帮助你很好地完成工作。

本书适合有一定经验、正致力于探究 iOS 应用漏洞的开发者，也适合对渗透测试感兴趣的读者。

版权贸易合同登记号　图字：01-2016-4319

图书在版编目（CIP）数据

iOS 应用安全权威指南 /（美）戴维·希尔（David Thiel）著；程伟，马超，李俊阳译. —北京：电子工业出版社，2017.1

书名原文：iOS Application Security: The Definitive Guide for Hackers and Developers

ISBN 978-7-121-30606-8

Ⅰ. ①i… Ⅱ. ①戴… ②程… ③马… ④李… Ⅲ. ①移动终端－应用程序－程序设计－安全技术 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2016)第 303077 号

策划编辑：张春雨

责任编辑：付　睿

印　　刷：三河市鑫金马印装有限公司

装　　订：三河市鑫金马印装有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　邮编 100036

开　　本：787×980　1/16　印张：18.25　字数：299 千字

版　　次：2017 年 1 月第 1 版

印　　次：2017 年 1 月第 1 次印刷

定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

献给那个姑娘，如果你正好走入我的心房

还有小时候对我的电脑设防、操心的爹娘

当然少不了喵星首长，生命因你们而辉煌

推荐序

在数字浪潮席卷全球之前，人们出门时并不会随身携带装满私人敏感信息的电子设备。而现在，几乎每个人都会随身携带一部手机，里面装满了各种私人信息。

智能手机给我们带来的不只是便利。它包含如此多的信息，对许多美国人来说，它就是“生活隐私”的载体。技术的进步让信息无处不在，但是这并不会降低信息的重要性，我们需要为了保护信息不断奋斗，就像国父们一样。

——首席大法官 John Roberts，出自莱利诉加利福尼亚案（2014）

大多数人都承认，智能手机是 21 世纪影响力最大的发明。自 2007 年第一代 iPhone 问世以来，智能手机的用户量暴增。到 2015 年年末写这本书时，全球已有近 34 亿手机用户，大概占世界人口的一半（全世界有超过 73 亿人）。在全球范围内，智能手机已经超过了电脑，成为访问互联网的主力。智能手机的普及对人类文明的影响完全能写一本书。手机正在改变世界，无数人通过手机访问互联网上的教育和娱乐资源，到处都是金矿。在某些国家，移动互联网和社交网络甚至能让专制政权垮台，能推动社会变革。

即使是美国最高法院的七旬老人也已经意识到现代移动设备的威力，并做出了新的判例。就像上面莱利诉加利福尼亚案所提到的，智能手机不仅是一部电子设备——它是人民隐私的入口。

和所有的技术革命一样，移动技术的普及也会产生一些负面影响。我们有

能力与世界各地的人建立联系，但这并不能提升面对面的沟通能力，而且移动技术也无法消除世界上长久以来的贫富差距。与此同时，和企业云计算、个人计算机及网络革命一样，智能手机也会引入新的安全漏洞，同时需要面对现有的各种安全问题。

2007 年发布的智能手机确实有一些重要的技术创新，但是真正吸引第三方开发者的是之后发布的 SDK 和开放应用商店。由此也诞生了一批新时代的开发者，他们需要从过去的安全教训中汲取经验，适应未来全新的、不确定的环境。

我和 David Thiel 已经相识 10 年之久，他对新技术的热情给我留下了深刻的印象。一旦出现新技术，David 就会马上通过检查、反编译、破解来掌握它，并用新知识来增强其他技术的安全性。David 很早就意识到 iPhone 会出现新的安全问题，因此从 iPhone 操作系统 SDK 发布的第一天起，他就已经开始研究应用开发者可能会犯的错误，以及如何越过平台限制开发安全的应用。

本书是迄今为止对 iOS 安全介绍最为全面的书籍。每一位心系用户的开发者都应该遵循本书的指导来设计自身的产品、组织结构和技术决策。David 把多年来踩过的坑和解决方案都写了下来，希望你能认真学习。

智能手机潜力巨大，但要让它真正发挥作用，我们必须尽最大努力让设备安全可信，保护用户隐私不被泄露。

Alex Stamos
Facebook 首席安全官

译者序

随着 2007 年第一代 iPhone 的诞生，一个崭新的移动互联网时代拉开了序幕。截至 iPhone 7 上市，苹果的 Apple Store 应用累计下载 1400 亿次，巨大的商机令全球的开发者们纷至沓来。截至 2015 年，仅中国的开发者人数就突破了 100 万，稳居世界第一。

随着硬件机能的日新月异，iOS——这一具有划时代意义的操作系统也迎来了它的 10 岁生日。在每年的 WWDC 上，苹果总是不遗余力地向全世界的开发者们展示最新的操作系统，介绍新的特性、新的 API，甚至是最佳编程实践。

然而正所谓树大招风，iPhone 用户也是黑产们眼中高质量的攻击目标。虽然苹果独特的沙盒机制已经最大限度地保护了用户的隐私安全，但目前针对 iOS 的黑色产业链已涉及方方面面。2015 年 9 月中国大陆地区出现了 XcodeGhost 病毒感染开发工具 Xcode 的风波，给所有 iOS 开发的从业者敲响了警钟，就连微信这种国民级 App 都未能幸免，这进一步说明了提升国内 iOS 开发者安全意识的紧迫性。

每次 iOS 版本更新时，市面上都会出版大量与 iOS 开发相关的书籍，但绝大部分的书都是教你如何使用 iOS SDK 提供的 API 来开发一个 App。大多数情况下，能熟练运用这些 API 就足够了，但如果想立志成为一名进阶的 iOS 开发者，使编写出来的代码更加稳定和安全，就要对 iOS 系统的底层细节和安全机制有所了解。可惜的是，市面上关于 iOS 系统安全机制和底层研究的书凤毛麟角。

Swift.GG 作为国内最走心的翻译组之一，通过一次偶然的机会得知电子工业出版社正在寻找《iOS 应用安全权威指南》一书的译者。双方一拍即合，决

定为中国的开发者们带来一本关于 iOS 安全机制与底层细节的书籍，从而补上 iOS 开发“安全”这最后一块拼图。

本书的作者 David Thiel，自 2008 年初代 iPhone 诞生后的第一年起，就开始对 iOS 应用进行各种安全检测和渗透实验，时至今日，已经积攒了大量的经验。无论你是刚入门的小菜鸟，还是已经摸爬滚打多年的老司机，本书都会循序渐进地带你重温 iOS 最基础的基本结构、安全背景；再更进一步地手把手教你搭建一个安全测试环境，进行代码分析、逆向工程；之后本书将 iOS 最薄弱、易受攻击的软肋一一列举，并告诉开发者如何避开这些陷阱；最后一部分，将会为我们演示如何利用加密手段最大限度地保护用户的数据和隐私。

本书在翻译过程中，得到了阿里巴巴安全大神蒸米同学的悉心指导；在本书的校对过程中，SwiftGG 的创始人梁杰同学提出了很高的要求，并投入了极大的精力，在此对二位提出特别感谢。参与翻译的李俊阳、马超同学，以及后期参与校对的 SwiftGG 小伙伴们，虽然大家身处异地，但从翻译到校对的整个过程都配合得行云流水，体现出了整个团队的高效与专业，无愧于国内“走心翻译组”这一称号，再次感谢大家。最后感谢电子工业出版社给了我们这次机会，感谢编辑们和审稿专家的细心检查。译者水平有限，bug 在所难免，还请读者批评指正。

程伟

2016 年 12 月 5 日

作者简介

David Thiel 在计算机安全领域有近 20 年的经验，他的研究成果和著作 *Mobile Application Security*（McGraw-Hill 出版社出版）促进了 iOS 应用安全领域的发展，他还多次在安全大会（比如 Black Hat 和 DEF CON）上进行演讲。Thiel 曾在 iSEC 担任多年应用安全顾问，目前就职于 Internet.org 的 Connectivity 实验室。

技术评审者

Alban Diquet 是软件工程师和安全研究员，主要研究领域包括 iOS 安全协议、数据隐私和移动安全。Diquet 曾发布过几个开源的安全工具，包括 SSLyze、iOS SSL Kill Switch 和 TrustKit。此外，他也经常出席各种安全会议，包括 Black Hat、Hack in the Box 和 Ruxcon。

前言

目前市面上有许多与 iOS 安全有关的文章，主要涉及 iOS 的安全模型、越狱、查找代码执行漏洞以及其他一些安全相关的特性。此外，还有一些文章从取证学角度来介绍，比如在犯罪调查中如何从物理设备或备份中提取数据。这些信息非常有用，但市面上主流的 iOS 书籍都在介绍应用开发，而本书的目标是要填补一个更大的空白。

在真实世界中，人们的注意力并没有聚焦在如何开发安全的 iOS 应用或对 iOS 应用进行安全评估上。由此产生的后果，就是在 iOS 应用中存在着一些令人尴尬的安全漏洞，这些漏洞会曝光用户的敏感数据、规避认证机制，甚至滥用用户隐私（无论是有意还是无意的）。随着 iPhone 等智能设备的普及，人们越来越多地使用 iOS 来处理一些关键任务，并委托这些应用来处理与之相关的大量敏感信息，所以 iOS 应用的安全性需要被充分考虑到。

我编写本书的目的，就是尽可能地为读者介绍真实环境中是如何安全地开发 iOS 应用程序的。iOS 是一个快速更迭的系统，但是我会尽可能讲解一些不变的知识，并教你使用一些工具来剖析 iOS 系统，让你能适应未来 API 的变化（万变不离其宗）。

不同版本的 iOS 存在不同的安全漏洞。虽然苹果已经终结了某些设备的寿命（停产），但是开发者仍然希望他们的应用能够运行在这些老旧设备上（比如第一代 iPad）。本书所要展示的漏洞覆盖了（本书完稿之时）从 iOS 5.x 到 9.0 的系统，针对每一个版本，我都会讨论相应的风险与应对措施。

本书的目标读者

首先，这是一本关于安全的书。如果你是一个开发者或者安全专家，并且正在研究 iOS 应用存在的漏洞（以及对应的修复方法），那么恭喜你，看这本书就对了！

如果你有一些 iOS 开发经验或者熟悉 iOS 应用的底层工作机制，那么你将从本书中学到大量的干货。不过，即使没有这些知识，只要你是一个有经验的程序员或渗透测试人员，必要时有钻研苹果官方文档的勇气，那就完全可以畅读本书。我会在第 2 章带你快速预览一遍 Objective-C 和一些常用的 API，顺便熟悉一下 Cocoa Touch。如果需要强化一下语言方面的基础知识（或者想复习一下），可以从第 2 章开始。

本书的内容

大概从 2008 年开始，我就一直在进行各种各样的 iOS 应用安全检测和渗透实验，从中收集了大量的陷阱和错误，这些都是开发者在实际开发 iOS 应用程序时会碰到的。如果你正在寻求更安全的应用开发最佳实践，又或者是作为一个安全专家想要学习如何定位 iOS 的安全问题，那么你一定不能错过本书总结的知识点。

本书的结构

第一部分：iOS 基础，你将深入了解 iOS 的背景，熟悉它的安全历史及其应用程序的基本结构。

- **第 1 章：iOS 安全模型**，简要分析 iOS 的安全模型，介绍该平台的基础防范措施，展示它能做什么、不能做什么。
- **第 2 章：Objective-C 简明教程**，解释了 Objective-C 与其他编程语言的不同之处，简要介绍了一些专业术语和设计模式。对于经验丰富的 Objective-C 程序员来说，这可能并不算什么新的内容，但是对于初学者和初次涉猎 iOS 的开发者来说很有价值。

- **第 3 章：iOS 应用剖析**，概述了 iOS 应用程序的结构和打包方式，研究了本地的存储机制以及泄露敏感信息的方式。

第二部分：安全性测试，你将学习如何在开发或渗透测试中建立安全的测试环境，我也会分享一些配置 Xcode 的小技巧，从而最大化地利用现有的安全机制。

- **第 4 章：构建测试平台**，工欲善其事必先利其器，本章将介绍所有用到的工具软件，并学习如何配置这些软件，让它们帮助我们来检查和测试 iOS 应用。具体来说，本章将会介绍模拟器、配置代理、绕过 TLS 验证以及分析应用的特征行为。
- **第 5 章：使用 lldb 和其他工具进行调试**，你可以使用 lldb 和 Xcode 的内建工具来更加深入地监控应用程序的行为，这些工具将帮助你分析代码中的复杂问题，还能帮你实现错误注入这样的功能。
- **第 6 章：黑盒测试**，深入研究一些工具，学习在没有源代码的情况下对应用程序进行分析。这涉及基本的逆向工程、二进制修改、复制程序以及使用 lldb 的远程实例在设备上进行调试的内容。

第三部分：Cocoa API 的安全怪癖，这部分会介绍 Cocoa Touch API 中常见的安全隐患。

- **第 7 章：iOS 网络通信**，讨论了网络和安全传输协议（TLS）在 iOS 中的工作原理，包括信息认证、证书锁定和 TLS 连接中的错误处理。
- **第 8 章：进程间通信**，介绍进程间的通信机制，包括 URL scheme 和最新的通用链接机制。
- **第 9 章：适用于 iOS 的 Web 应用**，涉及 Web 应用与 iOS 原生应用程序的集成，你可以直接使用 Web 视图或 JavaScript 和 Cocoa 的桥接框架 Cordova。
- **第 10 章：数据泄漏**，讨论了敏感数据泄漏的多种渠道，也许无意中，你的重要数据就会泄漏到本地存储、其他应用或是传播到网络中。
- **第 11 章：C 语言的遗留问题**，介绍 C 语言在 iOS 应用程序中遗留的漏洞：栈和堆的内存损坏、格式化字符串漏洞、内存释放后又重新使用以及一些 Objective-C 变种的经典漏洞。
- **第 12 章：注入攻击**，涉及的攻击主要有 SQL 注入、跨站点脚本、XML

注入以及谓词注入，它们都可以用来攻击 iOS 应用程序。

最后，**第四部分：保证数据安全**，介绍了数据的隐私和加密问题。

- **第 13 章：加密与认证**，本章将介绍加密的最佳实践，包括如何正确地使用钥匙串、数据保护 API 以及 CommonCrypto 框架提供的其他密码学原语。
- **第 14 章：移动端隐私问题**，在本书的最后一章将会讨论用户隐私，以及收集应用不需要的数据对于用户和开发者来说意味着什么。

学完本书之后，你会掌握以下技能：面对一个 iOS 应用程序，无论它是否提供源代码，你都能迅速地定位并找出它的安全漏洞。在实际的 iOS 开发过程中，你也能写出足够安全可靠的代码。

本书的约定

由于 Objective-C 是一门相当啰嗦的语言，有着极端冗长的类名和方法名，所以，为了便于阅读，我在源代码中封装了一下，但是这可能并不是你熟悉的代码格式。有时候代码看起来很别扭——你可以把它们复制到 Xcode 中，让 Xcode 来格式化代码。

更多的细节请阅读第 2 章，我支持在 Objective-C 中使用传统的中缀符而不是点语法。此外，我也习惯将花括号与方法声明放置在同一行，理由很简单：我是一个“老人”了。

书中 Objective-C 的类和方法名将使用 `monospaced` 字体，C 语言的函数也将同样采用 `monospaced` 字体。为了看上去简单明了，路径*/Users/<your username>/Library/Developer/CoreSimulator/*会用*$SIMPATH* 代替。

关于 Swift

业界对于苹果新出的 Swift 语言抱有极大的兴趣，但是我并不打算在此书中涉及，主要有以下几个原因。

首先，我还没有在实际的生产环境中使用过 Swift。Objective-C 在 iOS 应用开发中仍然是最受欢迎的语言，我们还要向前兼容很多年前的旧代码。

第二，目前由 Swift 产生的安全问题还比较少，因为它不是基于 C 开发的，

所以容易写出安全的代码。据我所知，到目前为止 Swift 语言本身还没有引入新的安全漏洞。

第三，Swift 使用了和 Objective-C 相同的 API，Cocoa Touch API 的安全隐患换一种语言也同样存在。所以你在本书中学到的知识对于 Objective-C 和 Swift 都适用。

最后，在 Swift 中不存在中缀符和方括号，对此我表示遗憾。

移动安全的承诺和威胁

当我第一次开发移动应用时，曾经怀疑过是否需要把移动应用的安全独立出来。当时我认为移动应用和桌面应用没什么不同，都存在着相同的漏洞：栈和堆的溢出、格式化字符串漏洞、内存被释放后又重新使用，以及其他一些代码执行问题。这些漏洞或许也存在于 iOS 之中，但是移动设备的安全焦点主要聚焦在用户隐私、数据窃取及进程间的恶意通信等方面。

在阅读这本 iOS 安全书籍时，请时刻提醒自己，用户不希望手机上安装的应用程序将自己的安全隐私暴露在风险之中。即使一个开发者刻意避开那些显而易见的风险，他还是需要做很多加固措施。本节会讨论应用程序应该对用户做出的安全承诺，以及对应这些承诺的攻击方式。

移动应用不应该具备哪些能力

人们从早期桌面操作系统的错误设计中汲取了经验，当前主流的移动操作系统都把应用程序相互隔离，这与桌面应用程序不同。而用户所运行的任意桌面应用程序可以访问全部的用户数据，甚至控制整个机器。

由于应用间的沙盒隔离和 iOS 平台的日益完善，用户期望也随之水涨船高。通常来讲，移动应用（包括你自己开发的）不应该做以下几件危险的事情。

导致其他应用程序行为异常

应用程序不应该让其他应用程序崩溃。在过去的“黑暗”时代，你不仅可以读取其他程序的数据，还可以修改或删除数据，甚至能够把整个操作系统搞

崩溃。随着时间的推移，通过桌面进程相互独立使这一状况得到改善，但它们的主要目标是增强稳定性，不是解决安全或隐私问题。

移动操作系统解决了这些问题，但是鱼和熊掌不可兼得，既要满足用户应用程序之间必要的互操作，又要保证所有进程的完全隔离，是不现实的。应用程序之间的边界总是存在着一些“孔洞”。这需要开发者来确保他们的应用不会做坏事，并采取一切谨慎措施来保障数据安全，防止其他恶意程序的侵袭。

拒绝为用户服务

iOS 就是为手机操作系统而生的，这决定了当一个用户需要紧急呼叫时，应用程序不能阻拦。很多地方的法律都有对应的规定，这也是攻击者（和用户）不能篡改底层操作系统的原因。

盗取用户数据

一个应用程序不应该读取来自其他应用程序或本地操作系统的数据，更不能将其发送给第三方。应用程序不能在未获得用户许可的情况下访问敏感数据。操作系统可以阻止一个应用直接读取另一个应用的数据，但是开发者需要自己防范其他的数据盗取手段，仔细研究发送和接收数据所要用到的 IPC 机制。

恶意扣费

在没有得到用户批准的情况下，应用程序不应该产生费用。许多手机恶意软件普遍存在的恶行就是让用户订阅第三方服务，这些服务所产生的费用又要由机主来买单。因此应用内的购买功能需要明确地向用户说明，并且最终的购买行为需要用户完全确认。

移动安全威胁分类

为了有助于理解移动设备的安全威胁以及对应的缓解措施，提前了解一些攻击类型将对我们非常有用。这将有助于我们分析实际存在的威胁，并帮助我们分析各种攻击和防御所产生的真实影响。

取证攻击

取证攻击者一旦有权限访问设备或者它的备份，就会尝试获取设备的机密。最常见的做法就是访问设备的物理区域。相比计算机，手机或平板电脑更容易被盗，因此大部分攻击都是取证攻击。

取证攻击可以由一个抱有碰运气心态的攻击者或熟练掌握定向攻击的攻击者发起。对于碰运气的攻击者来说，窃取信息就是偷一部没有任何 PIN 保护的手机，然后他们就能从手机上窃取图片、文字以及所有可以通过正常途径访问到的数据。如果用户开启了两步验证，攻击者也能很轻易地得到这些验证码信息或短信。

一个熟练的取证攻击者可能是一个流氓雇员、公司、政府、执法官员或者专门的敲诈勒索人员。这种攻击者知道某种技术可以用来实现临时越狱，破解简单的 PIN，然后访问设备文件系统中的数据。这些数据部分来自系统层，部分来自应用层。因此攻击者不但可以获取 UI 所展示的数据，也能得到系统底层的缓存数据。这些缓存数据包括截图、按键记录、Web 请求中缓存的敏感数据等。

我将在第 10 章列举取证攻击感兴趣的数据，然后在第 13 章介绍相对应的防范措施。

代码执行攻击

远程代码执行攻击会在设备上执行一段代码来破坏整个设备和数据，这个过程并不需要实际拿到该设备。这种攻击有多种渠道：网络、二维码、NFC、恶意文件解析，以及连接感染病毒的硬件外设。一旦设备执行了远程攻击代码，就可以执行取证攻击，从而获取用户的机密信息。有几种经常出现的代码执行攻击，它们都利用了底层编程语言的漏洞，我们将在第 11 章探讨。

基于 Web 的攻击

基于 Web 的远程代码执行攻击主要使用恶意的 HTML 和 JavaScript 来误导用户或窃取数据。远程攻击者可以操控一个恶意网站或者合法网站，也可以简单地将恶意代码发布到公共论坛。

这些攻击可以从 HTML5 的数据库或本地存储中窃取数据，还能修改或窃

取存储在 SQLite 数据库中的数据、读取会话 cookie 或植入一个虚假登录表单来盗取用户的凭据。我将会在第 9 章和第 12 章探讨更多与 Web 应用相关的问题。

基于网络的攻击

基于网络执行的代码攻击，通常会通过网络注入一些可执行代码，从而控制相关的应用程序或整个系统。具体来说，你可以修改设备的网络请求内容，也可以使用漏洞利用程序来破解系统服务或者内核。如果被攻击的目标具有比较高的权限，攻击者不仅可以获得某个特定应用程序的数据，还能获得设备中存储的全部数据。他们不仅可以监控设备的运行状态，还能植入后门程序方便今后的访问。我们将在第 7 章探讨网络相关的 API。

物理攻击

通过物理手段攻击设备，往往要用到 NFC 或 USB 接口。这种类型的攻击在过去被用在设备越狱上，但也可以通过短暂的物理接触攻击设备。大部分攻击都是针对操作系统的，不过在第 14 章将会介绍物理攻击的其他作用。

iOS 安全测试人员注意事项

渗透测试应该尽可能地结合源代码来执行。这样做的目的不是为了防范外部攻击，而是在有限的时间内最大限度地找出致命错误。真实世界里的攻击者有充足的时间去分析他们感兴趣的应用程序，而且 Objective-C 也非常适合做逆向工程。只要给他们足够的时间，总能解决问题。但是，大部分渗透测试受制于时间和金钱，因此我们没必要模拟真实世界的攻击者。

在本书中同时提到了白盒测试（有源码辅助）和黑盒测试，但是重点将集中在有源码辅助的渗透测试上，因为这样能更快地找出更多的问题，还可以帮助我们更好地学习 Cocoa 的基本库。其实本书中提到的很多技巧都可以用来辅助学习。

众所周知，很多 iOS 开发者都是由其他语言背景转过来的，原有的开发经验会引入不易察觉的安全隐患。无论是测试别人的应用还是自己的应用，要注