

丰生强 | 著  
邢俊杰 | 编

# macOS 软件安全 与逆向分析

## macOS软件安全第一书

软件开发与安全人员案头必备的技术专著



全面分析macOS系统  
中最新的软件安全、逆  
向分析与加密解密技术

包含信息安全领域一线  
软件安全专家的多年  
实战经验，干货满满

获BAT、360等众多  
互联网公司一线软件安  
全专家一致认可和推荐



中国工信出版集团



人民邮电出版社  
POSTS & TELECOM PRESS



丰生强 邢俊杰 | 著

人民邮电出版社  
北京

## 图书在版编目 (C I P) 数据

macOS软件安全与逆向分析 / 丰生强, 邢俊杰著. --  
北京 : 人民邮电出版社, 2017.7  
(图灵原创)  
ISBN 978-7-115-46063-9

I. ①m… II. ①丰… ②邢… III. ①软件开发—安全  
技术 IV. ①TP311.52

中国版本图书馆CIP数据核字(2017)第147374号

## 内 容 提 要

本书深入介绍 macOS 系统的软件安全、逆向分析与加密解密技术，主要包括 macOS 软件的开发基础、macOS 系统工作机制、macOS 软件调试接口与机制、二进制程序的格式、反汇编技术、逆向与动态调试技术、反破解技术以及系统安全与反病毒。本书包含了信息安全领域一线软件安全专家的多年实战经验，是安全人员与开发人员案头必备的技术专著。

本书适合所有 macOS 平台软件开发工程师、信息安全专业学生、信息安全专业从业人员阅读学习。

- 
- ◆ 著 丰生强 邢俊杰  
责任编辑 张霞  
责任印制 彭志环  
◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号  
邮编 100164 电子邮件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京鑫正大印刷有限公司印刷  
◆ 开本: 800×1000 1/16  
印张: 29.5  
字数: 697千字 2017年7月第1版  
印数: 1-3 500册 2017年7月北京第1次印刷
- 

定价: 79.00元

读者服务热线: (010)51095186转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京东工商广登字 20170147 号

**丰生强**，网名非虫，独立软件安全研究员，资深安全专家，ISC2016安全训练营独立讲师，有着丰富的软件安全实战经验。自2008年起，在知名安全杂志《黑客防线》上发表多篇技术文章，从此踏上软件安全道路，常年混迹于国内各大软件安全论坛。著有畅销安全图书《Android软件安全与逆向分析》。

---

**邢俊杰**，资深程序员，软件安全爱好者，C++ Web框架Cinatra开发者，对编译器与调试器开发有着深入的研究。现就职于国内某互联网公司。业余时间喜欢研究软件与系统的底层，热爱读书与动画。



微信连接



回复“安全”查看相关图书



微博连接

关注 @图灵教育每日分享IT好书



QQ连接

图灵读者官方群I: 218139230

图灵读者官方群II: 164939616

## 图灵社区 iTuring.cn

在线出版, 电子书, 《码农》杂志, 图灵访谈

站在巨人的肩上  
**Standing on Shoulders of Giants**



iTuring.cn

# 前　　言

对于很多人来说，苹果系统的硬件设备都让人瞩目，尤其是高配置的 Macbook Pro 电脑，无论是在价格上还是在品质上，都是笔记本电脑行业的标杆。2016 年，苹果公司在硬件新品发布会上公开了全新设计带 Touch Bar 的 Macbook Pro，顶配 15 英寸的价格高达 32 888 元，这个价格足以让多数人望而却步。苹果的 macOS 只允许在自家的硬件设备上运行，不过，除了顶配的 Macbook Pro，还有一些入门级的硬件可供选择。在苹果硬件系列中，最便宜的应该是 Mac mini，不过你还得另外买台显示器。算下来，在 macOS 系统上无论是做开发、娱乐还是其他工作，都要比其他操作系统的成本高出不少。这也导致了 macOS 系统的关注度比其他操作系统低。

本书讨论的内容偏偏正是这样一个受众较少的操作系统上软件安全相关的话题，相信读者与我一样，能够感受到这些年 macOS 系统的安全形势与变化。随着苹果系统普及率的提高，安全问题将会慢慢凸显。

## 本书特点

- 循序渐进的学习路线。从本书开篇起，书中的知识与技术点无一不是由浅入深逐渐展开的。工具的使用与原理的讲解也都符合学习的思维。读者在阅读本书时，几乎不需要频繁切换章节。
- 实例分析。在讲解不同技术点时，为了让读者充分感受技术涉及的应用场景与实际的展示效果，书中会辅以大量实例。每个实例都是笔者精心编写、反复调试过的，并且都是开放源代码的，读者可以通过阅读这些实例的源代码来加深对技术的理解。
- 实用工具讲解。本书提倡读者多动手实践，其中实践的一部分内容就是掌握书中介绍的第三方工具。本书除了系统地介绍一些命令外，还使用到了八十多个第三方工具。这些工具大多是免费与开源的，掌握这些工具的使用，阅读理解它们的代码，了解背后的工作原理，才能更好地让实践与理论相结合。

## 如何学习本书

本书共 12 章，系统地讲解了软件安全相关的环境搭建、语言基础、文件格式、静态分析、动态调试、破解与防破解、游戏安全、恶意软件等多个主题。其中，环境搭建与语言基础是本书

最低的技术门槛，属于软件开发的范畴。这部分内容主要针对零基础 macOS 开发的读者；如果你从事过 iOS/macOS 软件开发，那么可以跳过。后面每一个技术点都是独立又相辅相成的，虽然可以跳过一些章节进行学习，但不鼓励这么做。

在讲解工具的使用上，对于命令行工具，本书会以终端命令展示与输出结果的形式讲解；对于有界面的 GUI 工具，本书会辅以图示来讲解操作步骤与展示效果。读者在实际动手时，可以按照书中的指引一步步地进行操作。

## 本书适用人群

本书主要讲解 macOS 平台软件安全相关的技术，在读者的定位上自然是离不开“软件”与“安全”这两个领域的。

软件方面，适合的读者有：

- 软件开发专业的高校学生；
- iOS/macOS 软件开发工程师。

安全方面，适合的读者有：

- 信息安全专业的高校学生；
- 软件汉化人员；
- 软件安全研究员；
- 系统底层开发人员；
- 逆向工程师；
- 病毒分析师。

## 阅读须知

本书的创作花费了笔者大量的时间与精力，它得以顺利出版，是无数个日日夜夜调试与写作的成果。因此，我不欢迎阅读本书盗版的读者，无论你通过什么渠道，出于什么目的，当你通过阅读盗版书看到这番话时，都是对笔者深深的伤害。

任何一种技术都有它的应用场景，任何一种知识都有更新与迭代期。本书创作时，恰遇 OS X 系统更名为 macOS，系统刚升级至 10.12，因此本书中讲解的工具与技术，是基于这个时期的系统版本，不能保证这些工具与技术在系统日后的版本上依旧能够适用。因此，本书不对日后版本系统上的可行性做任何担保，也不欢迎好事之徒对书中技术的可行性进行无端猜测。

本书是一本工具实践书，讲解了工具的使用与原理以及在实际分析过程中遇到的多数问题，但这并不代表本书能够帮你解决所有的问题。而且单纯通过阅读一本书并不足以完全了解逆向工程这一门深奥的学问。如果读者期待仅仅通过本书就完整地理解系统的安全机制与所有的软件攻

防手法，那么本书可能不适合你。

逆向工程是一门特殊的技术，它就像一把利刀，使用得当可以保护自己，使用不当就会伤害别人。本书中的技术只供用来做技术探讨，不得用于非法商业目的，任何企图通过本书技术从恶的读者，都请好自为之。

## 实例代码与勘误

本书中的实例代码全部托管到了 GitHub 上，读者可以到 <https://github.com/feicong/macbook> 页面下载。

本书在图灵社区本书主页和 GitHub 上均开设了勘误页面，如果细心的读者发现了书中的错误，可以提交 issue。

- 图灵社区：<http://www.ituring.com.cn/book/1958>
- GitHub：[https://github.com/feicong/macbook\\_issues](https://github.com/feicong/macbook_issues)

## 致谢

首先，感谢父母的养育之恩，是他们给予了我生命。他们是最可爱的人！

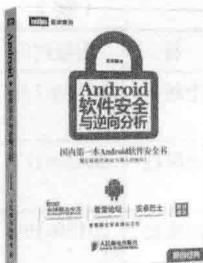
感谢老婆蓉对我工作的理解，由于工作的忙碌与截稿期的临近，我把本应该陪伴她的大量休息时间花在了伏案写作上（我不会在这里告诉你，我已经在为下一次的旅游计划做准备了）。

感谢图灵公司对本书的认可与支持，感谢策划编辑张霞对本书内容的跟进与督促，这才使得书稿能够顺利地完成。

感谢 Proteas 和熊猫正正在百忙之中抽出时间对本书进行了技术审校，他们的宝贵意见和建议使我受益匪浅。

此外，在写作过程中，我得到了很多朋友与同行的帮助。熊猫正正一起构建了本书的写作框架，提供了丰富的 macOS 系统与逆向方面的学习资料，并审阅了初稿，这些都是宝贵且难得的；仙果提供了部分章节中逆向工具方面的演示效果；黄药师给出了游戏安全章节的有效建议；Claud Xiao 对目录的制定提供了建设性的建议；还有很多朋友在写作内容与审稿上给出了宝贵的意见，他们是：Proteas、听鬼哥说故事、怒吼①聲つ喵、人生无 NG（淡然出尘）（排名不分先后）……在这里向你们表示衷心的感谢。

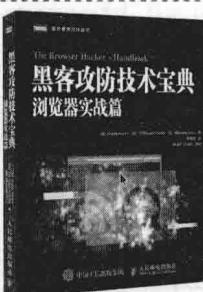
# 推荐阅读



- 国内首本Android安全图书
- 各大网店同类图书销量榜首
- 深入讲解Android攻击与防范

书号: 978-7-115-30815-3

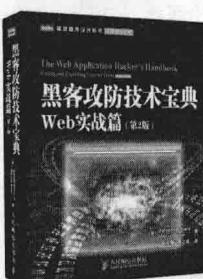
定价: 69.00 元



- 浏览器安全领域的先锋之作、浏览器攻击框架BeEF团队实战经验总结
- 涵盖所有主流浏览器以及移动浏览器
- 分三阶段、七大类讲解浏览器攻防方法

书号: 978-7-115-43394-7

定价: 109.00 元



- 安全技术宝典全新升级
- 亚马逊书店五星赞誉
- 探索和研究Web应用程序安全漏洞的实践指南

书号: 978-7-115-28392-4

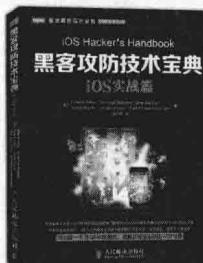
定价: 99.00 元



- Android安全第一书，专注于阐述设备root、逆向工程、漏洞研究和软件漏洞利用等技术细节
- 顶级白帽子原著 + 一线安全专家演绎
- tombkeeper、Flanker、dm557、非虫等知名白帽子鼎力推荐

书号: 978-7-115-38570-3

定价: 89.00 元



- 美国国家安全局全球网络漏洞攻击分析师、连续4年Pwn2Own黑客竞赛大奖得主Charlie Miller主笔
- 作者阵容超级豪华，6位均为信息安全领域大名鼎鼎的顶级专家，各有所长，且多有专著出版
- 国内唯一专注iOS平台漏洞、破解及安全攻防的中文专著

书号: 978-7-115-32848-9

定价: 69.00 元

# 目 录

## 第 1 章 如何分析 macOS 软件 ..... 1

1.1 分析环境搭建 ..... 1
1.1.1 安装 Clang ..... 1
1.1.2 HT Editor ..... 2
1.1.3 Homebrew ..... 6
1.2 第一个 macOS 程序 ..... 8
1.3 使用 HT Editor 进行破解 ..... 10
1.4 本章小结 ..... 14

## 第 2 章 系统安全架构 ..... 15

2.1 系统架构概述 ..... 15
2.1.1 shell 环境 ..... 16
2.1.2 目录结构 ..... 16
2.1.3 文件权限 ..... 17
2.2 系统调用 ..... 17
2.3 进程间通信 ..... 18
2.4 安全框架 ..... 19
2.4.1 CommonCrypto ..... 19
2.4.2 Keychain ..... 20
2.4.3 安全传输 ..... 25
2.5 系统安全机制 ..... 28
2.5.1 FileVault 2 ..... 29
2.5.2 代码签名 ..... 31
2.5.3 ASLR / kASLR ..... 33
2.5.4 沙盒 ..... 37
2.5.5 Rootless ..... 39
2.5.6 Gatekeeper ..... 42
2.6 软件安全开发建议 ..... 50
2.7 本章小结 ..... 50

## 第 3 章 软件开发基础 ..... 51

3.1 Objective-C 语言 ..... 51
-----------------------------

3.1.1 开发环境 ..... 51
3.1.2 Objective-C 语言特性 ..... 54
3.1.3 内存管理 ..... 60
3.2 Swift 语言 ..... 65
3.2.1 Playground ..... 65
3.2.2 Swift 语法简介 ..... 67
3.3 其他语言 ..... 88
3.4 框架 ..... 88
3.4.1 框架的开发与使用 ..... 88
3.4.2 在 Objective-C 中使用 Swift 编写的框架 ..... 93
3.4.3 常用的框架 ..... 94
3.5 第三方开发工具 ..... 94
3.5.1 Qt Creator ..... 94
3.5.2 Xamarin Studio ..... 95
3.5.3 JetBrains 系列开发工具 ..... 96
3.5.4 Visual Studio Code ..... 97
3.6 完整的 Cocoa GUI 程序 ..... 97
3.6.1 创建工程 ..... 98
3.6.2 Storyboard 和 xib ..... 98
3.6.3 Outlet 和 Action 机制 ..... 101
3.7 本章小结 ..... 103

## 第 4 章 软件内幕 ..... 104

4.1 可执行文件 ..... 105
4.2 下载与安装软件 ..... 106
4.2.1 免费与付费软件 ..... 106
4.2.2 安装软件 ..... 106
4.3 Bundle ..... 107
4.3.1 Bundle 目录结构 ..... 107
4.3.2 在代码中访问 Bundle ..... 109

4.4 通用二进制格式 .....	109	5.5.1 与 C 语言互相调用 .....	197
4.5 Mach-O 文件格式 .....	112	5.5.2 使用系统调用 .....	200
4.5.1 Mach-O 简介 .....	112	5.6 本章小结 .....	201
4.5.2 Mach-O 头部 .....	113		
4.5.3 加载命令 .....	116		
4.5.4 LC_CODE_SIGNATURE .....	117		
4.5.5 LC_SEGMENT .....	129		
4.6 动态库 .....	131		
4.6.1 构建动态库 .....	132		
4.6.2 dyld .....	135		
4.6.3 动态库的加载 .....	136		
4.7 静态库 .....	151		
4.7.1 构建静态库 .....	152		
4.7.2 静态库格式 .....	154		
4.7.3 管理静态库 .....	156		
4.8 框架 .....	156		
4.8.1 构建框架 .....	157		
4.8.2 框架的使用与安装 .....	158		
4.9 pkg .....	160		
4.9.1 构建 pkg .....	160		
4.9.2 pkg 的安装与卸载 .....	167		
4.9.3 pkg 文件格式 .....	170		
4.9.4 破解 pkg .....	173		
4.10 dmg .....	177		
4.10.1 构建 dmg .....	177		
4.10.2 管理 dmg .....	179		
4.11 本章小结 .....	181		
<b>第 5 章 汇编基础 .....</b>	<b>182</b>		
5.1 搭建汇编语言开发环境 .....	182		
5.2 Hello World 代码概览 .....	185		
5.3 伪指令 .....	186		
5.4 x86_64 汇编基础 .....	189		
5.4.1 寄存器 .....	190		
5.4.2 汇编语法 .....	192		
5.4.3 数据传送指令 .....	195		
5.4.4 控制转移指令 .....	195		
5.4.5 栈操作指令 .....	196		
5.4.6 运算指令 .....	197		
5.5 与其他模块的交互 .....	197		
<b>第 6 章 软件静态分析 .....</b>	<b>202</b>		
6.1 代码分析与二进制分析 .....	202		
6.2 分析工具 .....	203		
6.2.1 Radare2 .....	203		
6.2.2 IDA Pro .....	207		
6.2.3 Hopper .....	209		
6.3 代码分析技术 .....	211		
6.3.1 行为分析 .....	211		
6.3.2 资源分析 .....	212		
6.3.3 数据分析 .....	215		
6.3.4 流量分析 .....	216		
6.3.5 API 分析 .....	218		
6.4 反汇编工具的使用 .....	219		
6.4.1 反汇编 .....	219		
6.4.2 流程图 .....	224		
6.4.3 伪代码 .....	225		
6.5 破解 Mach-O 程序 .....	227		
6.5.1 定位修改点 .....	227		
6.5.2 修改程序 .....	228		
6.5.3 代码签名处理 .....	230		
6.5.4 重新打包 .....	234		
6.5.5 Keygen .....	234		
6.6 本章小结 .....	235		
<b>第 7 章 软件动态调试与跟踪 .....</b>	<b>236</b>		
7.1 DTrace .....	236		
7.1.1 DTrace 简介 .....	236		
7.1.2 DTrace 示例 .....	236		
7.2 D 脚本语言 .....	237		
7.2.1 脚本加载方式 .....	237		
7.2.2 D 语言与 C 语言 .....	238		
7.2.3 D 语言语法 .....	238		
7.2.4 变量 .....	241		
7.2.5 参数传递 .....	243		
7.2.6 聚合 .....	243		
7.2.7 内置函数与变量 .....	244		

7.3 调试器.....	246
7.3.1 GDB .....	246
7.3.2 LLDB .....	248
7.3.3 IDA Pro .....	258
7.3.4 Hopper.....	267
7.4 本章小结.....	269
<b>第 8 章 调试器开发 .....</b>	<b>270</b>
8.1 概述 .....	270
8.2 开发环境搭建 .....	270
8.2.1 安装所需环境.....	271
8.2.2 编译 Saber.....	280
8.3 系统调试接口 .....	285
8.3.1 ptrace 简介.....	286
8.3.2 Mach 调试接口.....	287
8.4 macOS 异常机制.....	292
8.4.1 异常与 Mach RPC/IPC.....	292
8.4.2 信号.....	300
8.5 调试器功能实现 .....	302
8.5.1 调试器架构 .....	302
8.5.2 开始调试 .....	303
8.5.3 异常处理循环.....	305
8.5.4 读写被调试进程内存 .....	308
8.5.5 获取基地址与入口点 .....	309
8.5.6 单步调试 .....	310
8.5.7 断点 .....	311
8.5.8 继续运行 .....	312
8.5.9 反汇编 .....	313
8.6 本章小结.....	316
<b>第 9 章 破解技术.....</b>	<b>317</b>
9.1 软件破解步骤 .....	317
9.2 常见的保护类型 .....	318
9.2.1 试用版&序列号.....	319
9.2.2 License 授权 .....	319
9.2.3 重启验证与暗桩.....	330
9.2.4 防拷贝技术 .....	338
9.2.5 网络验证 .....	338
9.2.6 混合验证 .....	342
9.3 App Store 内购机制 .....	342
9.4 Hook 技术 .....	351
9.4.1 DYLD_INSERT_LIBRARIES .....	352
9.4.2 SymbolTable Hook.....	355
9.4.3 Inline Hook .....	358
9.4.4 Method Swizzling .....	359
9.5 代码注入 .....	362
9.5.1 静态注入 .....	362
9.5.2 动态注入 .....	365
9.5.3 Hook 与注入框架 .....	366
9.6 补丁&注册机 .....	373
9.7 本章小结.....	375
<b>第 10 章 反破解技术 .....</b>	<b>376</b>
10.1 反破解技术类型 .....	376
10.2 校验保护 .....	377
10.2.1 完整性检查 .....	377
10.2.2 代码签名验证 .....	377
10.2.3 沙盒检测 .....	382
10.2.4 来源检测 .....	386
10.3 代码保护 .....	386
10.3.1 代码混淆 .....	386
10.3.2 SMC .....	387
10.3.3 代码校验 .....	387
10.3.4 壳保护 .....	387
10.4 数据保护 .....	391
10.4.1 数据清除 .....	391
10.4.2 数据存储 .....	395
10.4.3 数据传输 .....	400
10.5 调试器对抗 .....	408
10.5.1 调试器检测 .....	408
10.5.2 反调试 .....	410
10.6 Hook 检测 .....	411
10.6.1 Method Swizzling 检测 .....	411
10.6.2 dyld Hook 检测 .....	412
10.7 本章小结.....	413
<b>第 11 章 游戏安全 .....</b>	<b>414</b>
11.1 游戏类型 .....	414
11.2 游戏框架与引擎 .....	414
11.2.1 SpriteKit 与 SceneKit .....	415
11.2.2 GameplayKit & ReplayKit .....	417
11.2.3 Cocos2d-x.....	417

---

11.2.4	Unity3D	419
11.3	游戏分析工具	422
11.3.1	静态分析工具	423
11.3.2	动态调试工具	424
11.3.3	资源修改工具	424
11.3.4	内存修改工具	427
11.4	游戏分析方法	427
11.4.1	对比分析	427
11.4.2	动态调试	429
11.4.3	静态补丁	429
11.4.4	动态补丁	430
11.5	防破解技术	430
11.6	本章小结	431
<b>第 12 章</b>	<b>恶意软件与 Rootkit</b>	<b>432</b>
12.1	安全趋势	432
12.1.1	知名恶意软件	432
12.1.2	安全漏洞	433
12.1.3	安全软件	435
12.2	文件关联技术	435
12.3	软件自启动技术	439
12.3.1	Launch Items	439
12.3.2	Login Items	441
12.3.3	StartupItems	442
12.3.4	Login/Logout Hooks	444
12.3.5	Cron Jobs	444
12.3.6	Periodic Scripts	446
12.3.7	Authorization Plugins	446
12.3.8	Browser Extensions	447
12.3.9	Spotlight Importers	448
12.3.10	QuickLook Plugins	448
12.3.11	Kernel Extensions	448
12.4	Rootkit	449
12.4.1	文件隐藏	449
12.4.2	进程隐藏	451
12.4.3	内核模块隐藏	452
12.4.4	Root 提权	453
12.5	本章小结	454
<b>附录</b>	<b>macOS 工具一览表</b>	<b>455</b>
<b>参考资料</b>		<b>460</b>

## 第1章

# 如何分析macOS软件

# 1

万事开头难。许多希望学习逆向工程的朋友通常在网上翻看了许多相关的博客和教程之后仍会觉得无从下手，第1章将会带你从头开始搭建一个最简单的分析环境，引导你自己动手写一个简单的CrackMe并破解它。

这一章不会介绍复杂的IDE，也不会使用各种“酷炫”的逆向工具，一切删繁就简，只使用命令行工具和简单的命令，来完成我们第一次Mac平台的逆向之旅，让你对逆向的过程有一个初步的认识。虽然目前在这个分析环境中只有几个命令行工具，但在后面的章节我们会不断地扩充。另外，Mac平台上的分析工具目前还不像Windows上那样种类繁多，所以我们还会自己开发一些实用的工具，添加到我们的分析环境中，使它变得更为充实。

## 1.1 分析环境搭建

首先，我们需要一个编译器来编译代码，目前在macOS系统上最流行的编译器自然是大名鼎鼎的Clang。

### 1.1.1 安装 Clang

如果你安装过Xcode，那说明你已经安装了Clang编译器，就可以先跳过本节。

Clang隶属于苹果公司的开源项目LLVM，是LLVM的一个前端，LLVM的官网为<http://llvm.org>，如图1-1所示。

你可以直接到LLVM官网下载编译好的Clang，但是这样下载Clang缺少一些必要的工具，使用起来很不方便，这些工具大部分跟Clang一样，包含在苹果的开发工具包中，这个工具包可以直接从App Store上下载。但是，还有更简便的方法。

首先打开一个终端，点击Launchpad→其他→终端，在终端中输入clang并回车，系统会自动检测到我们有没有安装Clang编译器，然后会提示我们是否下载并安装命令行开发者工具，如图1-2所示。

The screenshot shows the LLVM website homepage. On the left, there's a sidebar with a 'Site Map' containing links like Overview, Features, Documentation, Command Guide, FAQ, Publications, LLVM Projects, Open Projects, LLVM Users, Bug Database, LLVM Logo, Blog, Meetings, and LLVM Foundation. Below that is a 'Download!' section with links for Download now, LLVM 3.9.1, All Releases, APT Packages, Win Installer, and View the open-source license. The main content area has three main sections: 'LLVM Overview' (describing the LLVM Project as a collection of modular and reusable compiler and toolchain technologies), 'Latest LLVM Release!' (mentioning LLVM 3.9.1), and 'ACM Software System Award!' (noting LLVM's award). The LLVM Overview section includes two numbered points: 1. Describing the LLVM Core libraries and their optimizer, and 2. Describing Clang as an LLVM native C/C++/Objective-C compiler.

图1-1 LLVM官网

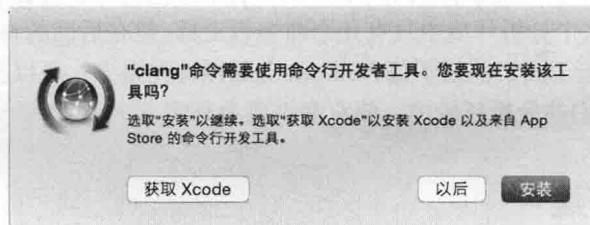


图1-2 安装命令行开发者工具提示

选择“安装”，就会出现安装协议，同意安装协议，过一会儿Clang编译器就会下载并安装到系统中了，一起安装的还有make等常用的命令行编译工具。我们可以执行`clang -v`查看Clang是否安装正确。

```
$ clang -v
Apple LLVM version 7.0.0 (clang-700.1.76)
Target: x86_64-apple-darwin14.5.0
Thread model: posix
```

如果能正确输出版本信息，则说明已经成功安装了Clang，接下来就可以使用它来编译程序了。

### 1.1.2 HT Editor

HT Editor是一个开源跨平台的十六进制编辑器，但它的功能可远远不止十六进制编辑器这么简单，它还有强大的反汇编/汇编功能，支持x86、x64、ARM、Power等多种处理器，并支持Windows平台上的PE文件格式、Linux上的ELF格式以及macOS的Mach-O文件格式。我们在这里要用它来破解CrackMe。

HT Editor的官网为`http://hte.sourceforge.net/`, 如图1-3所示( 打开该网站可能需要国外的代理)。

The screenshot shows the HT Editor website at `hte.sourceforge.net`. The top navigation bar includes links for News, Documentation, Downloads, Screenshots, Authors, and Links. The main content area features several news items:

- HT 2.1.0 released** (Submitted by **Seppel** on 11th of January, 2015): This release features more advanced display and handling of PE relocations and a PE checksum calculation (thanks Mertens Engineering). HT now also contains a disassembler for the Atmel AVR 8-bit microcontroller. Additionally we fixed a lot of crashes concerning broken ELF files. We also updated the included minilzo.
- ChangeLog | Downloads**
- Official HT source repository moved to Github** (Submitted by **Seppel** on 14th of September, 2014): The official source repository of HT moved to Github!
- HT 2.0.22 released** (Submitted by **Seppel** on 11th of January, 2015): Amongst various bugfixes, this release features the new option "editor/scroll-offset" which determines how many extra lines the cursor should be visible when scrolling (Thanks tecknicaltom).
- ChangeLog | Downloads**

图1-3 HT Editor官网

HT Editor官网只提供了Windows版本的二进制文件，在Mac上我们需要自己动手编译来生成它。不过不要担心，这并不是什么难事，在macOS上编译大多数开源项目跟在UNIX系统是一样的，基本上只需要“`./configure && make`”就可以了。

首先，点击Downloads超链接，下载最新版本的源代码，如图1-4所示。

The screenshot shows the HT Editor website's Downloads page at `hte.sourceforge.net/downloads.html`. The top navigation bar includes links for News, Documentation, Downloads, Screenshots, Authors, and Links. The main content area features two tables:

### Current releases

release date	version	what	where
11.01.2015	2.1.0	sources-bz2 binary-win32	<a href="#">ht-2.1.0.tar.bz2</a> <a href="#">ht-2.1.0-win32.exe</a>

### Get source from GitHub

```
git clone git@github.com:seppel/hteditor.git
```

### Old releases

release date	version	what	where
13.06.2013	2.0.22	sources-gzip sources-bz2 binary-win32	<a href="#">ht-2.0.22.tar.gz</a> <a href="#">ht-2.0.22.tar.bz2</a> <a href="#">ht-2.0.22-win32.exe</a>
20.11.2012	2.0.21	source-gzip source-bz2 binary-win32	<a href="#">ht-2.0.21.tar.gz</a> <a href="#">ht-2.0.21.tar.bz2</a> <a href="#">ht-2.0.21-win32.exe</a>
03.03.2012	2.0.20	sources-gzip sources-bz2	<a href="#">ht-2.0.20.tar.gz</a> <a href="#">ht-2.0.20.tar.bz2</a>

图1-4 HT Editor官网

在写作本书时，HT的最新版本是2.1.0。点击“`ht-2.1.0.tar.bz2`”超链接下载源代码并将其解压缩到一个合适的目录，我将其解压到用户目录下的Project目录中。然后打开一个终端，使用`cd`命令切换到源代码所在的目录，然后在终端中执行`./configure`并回车，会看到大量的“`checking`”：

```
$ cd Project/ht-2.1.0/
$ ./configure
checking build system type... x86_64-apple-darwin14.5.0
checking host system type... x86_64-apple-darwin14.5.0
checking target system type... x86_64-apple-darwin14.5.0
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... ./install-sh -c -d
checking for gawk... no
checking for mawk... no
checking for nawk... no
checking for awk... awk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes

.....
config.status: creating minilzo/Makefile
config.status: creating output/Makefile
config.status: creating tools/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands

./configure successful.

=====
Configuration summary
=====

X11 textmode support available: no
enable profiling: no
make a release build: yes
using included minilzo: yes
```

如果一切顺利，会在终端输出“./configure successful.”，表示已经成功生成了编译需要的makefile文件。

---

**提示** 如果读者安装了X11，需要将命令改为`./configure --disable-x11-text-mode`禁用X11 textmode的支持，否则可能会导致编译错误。

---

接下来输入`make`并回车，`make`程序会自动根据之前生成的makefile进行编译，并产生更多的编译输出。

```
$ make
/usr/bin/make all-recursive
Making all in tools
gcc -DHAVE_CONFIG_H -I. -I.. -DNOMACROS -O3 -fomit-frame-pointer -Wall -fsigned-char
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -MT bin2c.o -MD -MP -MF .deps/bin2c.Tpo -c -o bin2c.o
bin2c.c
bin2c.c:135:6: warning: variable 'inname' is used uninitialized whenever 'if' condition is false
```

Clang会以彩色输出显示编译中的警告和错误，一般警告为鲜红色，错误为暗红色。编译完成后如果没有错误，就可以在源代码根目录中看到编译好的HT。

### 1.1.3 Homebrew

很多时候，需要在系统中安装一些命令行工具与脚本，比如，安装wget或curl作为命令行下的下载工具，安装Git版本控制软件来管理工程的源代码。比较传统的安装方式是到这些软件的官网上下载编译好的程序，或者下载源代码进行编译，然后将程序放到一个目录下，将路径添加到PATH环境变量下，以后在终端中直接输入命令就可以使用它们。如果所有这些工具都使用这种方式安装，不但不方便进行管理，而且需要花费太多时间去搜索与安装它们。在主流的UNIX系统上，一般都有对这类软件进行统一管理的工具，如Ubuntu系统的apt-get，安装wget只需要在Ubuntu的终端中执行sudo apt-get install wget，就会在系统中自动安装wget。

苹果系统并没有提供这样的管理工具，但幸运的是，已经有第三方开发人员开发了这样的工具，并免费供用户使用。主流的有Homebrew与Macports，这里以Homebrew的使用为例。安装Homebrew只需要在终端中执行下面的代码即可：

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

执行后会产生如下输出：

```
=> This script will install:  
/usr/local/bin/brew  
/usr/local/share/doc/homebrew  
/usr/local/share/man/man1/brew.1  
/usr/local/share/zsh/site-functions/_brew  
/usr/local/etc/bash_completion.d/brew  
/usr/local/Homebrew  
=> The following new directories will be created:  
/usr/local/Cellar  
/usr/local/Homebrew  
/usr/local/Frameworks  
/usr/local/bin  
/usr/local/etc  
/usr/local/include  
/usr/local/lib  
/usr/local/opt  
/usr/local/sbin  
/usr/local/share  
/usr/local/share/zsh  
/usr/local/share/zsh/site-functions  
/usr/local/var
```

Press RETURN to continue or any other key to abort

```
=> /usr/bin/sudo /bin/mkdir -p /usr/local/Cellar /usr/local/Homebrew /usr/local/Frameworks  
/usr/local/bin /usr/local/etc /usr/local/include /usr/local/lib /usr/local/opt /usr/local/sbin  
/usr/local/share /usr/local/share/zsh /usr/local/share/zsh/site-functions /usr/local/var
```

WARNING: Improper use of the sudo command could lead to data loss  
or the deletion of important system files. Please double-check your  
typing when using sudo. Type "man sudo" for more information.

To proceed, enter your password, or type Ctrl-C to abort.

```

Password:
=> /usr/bin/sudo /bin/chmod g+rwx /usr/local/Cellar /usr/local/Homebrew /usr/local/Frameworks
/usr/local/bin /usr/local/etc /usr/local/include /usr/local/lib /usr/local/opt /usr/local/sbin
/usr/local/share /usr/local/share/zsh /usr/local/share/zsh/site-functions /usr/local/var
=> /usr/bin/sudo /bin/chmod 755 /usr/local/share/zsh /usr/local/share/zsh/site-functions
=> /usr/bin/sudo /usr/sbin/chown mbp /usr/local/Cellar /usr/local/Homebrew /usr/local/Frameworks
/usr/local/bin /usr/local/etc /usr/local/include /usr/local/lib /usr/local/opt /usr/local/sbin
/usr/local/share /usr/local/share/zsh /usr/local/share/zsh/site-functions /usr/local/var
=> /usr/bin/sudo /usr/bin/chgrp admin /usr/local/Cellar /usr/local/Homebrew /usr/local/Frameworks
/usr/local/bin /usr/local/etc /usr/local/include /usr/local/lib /usr/local/opt /usr/local/sbin
/usr/local/share /usr/local/share/zsh /usr/local/share/zsh/site-functions /usr/local/var
=> /usr/bin/sudo /bin/mkdir -p /Users/mbp/Library/Caches/Homebrew
=> /usr/bin/sudo /bin/chmod g+rwx /Users/mbp/Library/Caches/Homebrew
=> /usr/bin/sudo /usr/sbin/chown mbp /Users/mbp/Library/Caches/Homebrew
=> Searching online for the Command Line Tools
=> /usr/bin/sudo /usr/bin/touch /tmp/.com.apple.dt.CommandLineTools.installondemand.in-progress
.....

```

在安装过程中会创建系统的目录与软链接，如果系统没有安装Command Line Tools，则会自动下载安装。安装完成后，执行brew doctor命令可以查看Homebrew的环境是否正常。通常在第一次安装brew之后，还需要安装苹果的Command Line Tools。如果事先安装过Xcode，Command Line Tools会在Xcode第一次启动时提示安装。

安装好Homebrew后，执行以下命令就可以安装指定的软件包。

```
$ brew install 软件包名
```

更新Homebrew所有的软件可以执行以下命令。

```
$ brew update
$ brew upgrade
```

卸载指定的软件包可以执行：

```
$ brew remove 软件包名
```

上一节介绍的HT Editor在Homebrew中也有移植版本，只需要执行以下命令即可自动安装。

```
$ brew install ht
=> Installing dependencies for ht: lzo
=> Installing ht dependency: lzo
=> Downloading https://homebrew.bintray.com/bottles/lzo-2.09.el_capitan.bottle.tar.gz
#####
100.0%
=> Pouring lzo-2.09.el_capitan.bottle.tar.gz
/usr/local/Cellar/lzo/2.09: 29 files, 565.0K
=> Installing ht
=> Downloading https://homebrew.bintray.com/bottles/ht-2.1.0.el_capitan.bottle.tar.gz
#####
100.0%
=> Pouring ht-2.1.0.el_capitan.bottle.tar.gz
/usr/local/Cellar/ht/2.1.0: 8 files, 2.0M
```

Homebrew会依次下载安装HT Editor的依赖库，然后安装HT Editor并配置好它的路径，整个安装过程不需要用户手动干预，安装完成后，在终端下执行ht就可以打开HT Editor了。

此外，还有一个基于Homebrew的扩展工具Homebrew-Cask，使用它可以直接下载App Store中的GUI程序，安装Homebrew-Cask需要执行如下命令：

```
$ brew tap phinze/cask
$ brew install brew-cask
```

完成后就可以安装其他软件了，如安装腾讯的QQ聊天工具就可以执行如下命令：

```
$ brew cask install qq
```

如果你觉得在命令行下管理软件不够直观，可以尝试基于Homebrew移植的GUI版本CakeBrew，只需要到它的官网<https://www.cakebrew.com>下载安装即可。CakeBrew运行效果如图1-5所示。

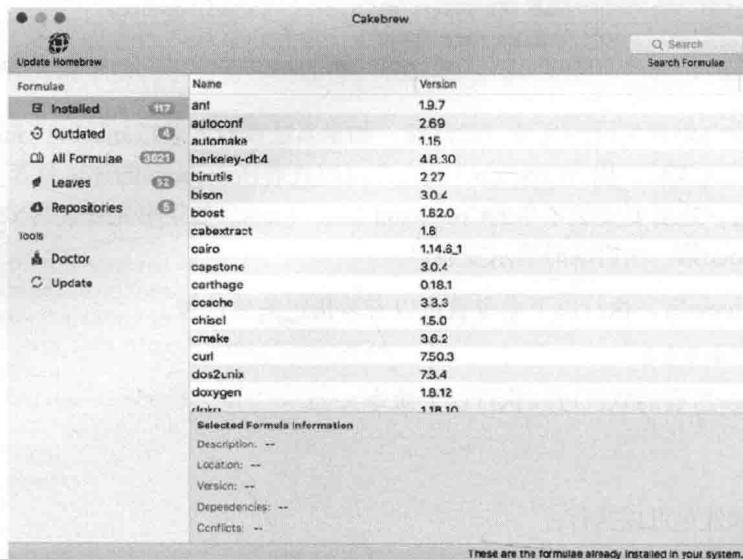


图1-5 CakeBrew运行效果

## 1.2 第一个macOS程序

现在我们已经装好了编译环境，并使用这个编译环境编译了一个开源项目，接下来就要用这个环境编写CrackMe了。

首先，我们需要一个编辑器编写代码。这里为了演示方便，选择的是系统自带的Vim编辑器，读者也可以自行选择其他方便的编辑器（后面会介绍更多实用的编辑器，本书附录的工具一览表中也会列出它们）。

我并不希望读者从一开始就陷入复杂的IDE的细节中，而是希望大家将重点放到代码的编写和逆向分析过程上。关于Xcode等IDE的使用，会在第3章讲解软件开发时进行详细的介绍。

首先，在Project文件夹下创建一个CrackMe01文件夹，里面存放编写的源代码以及编译结果：

```
$ cd Project
$ mkdir CrackMe01 && cd CrackMe01/
```

创建CrackMe的源代码文件cm01.c：

```
$ vim cm01.c
```

这时可以看到出现了Vim的命令行界面，并且左下角有“"cm01.c" [New File]”的提示，此时在键盘上按“a”键进入编辑模式。“a”键表示在当前光标位置之后插入字符，之后就可以像正常编辑器一样输入代码了，但是需要注意的是，数字不能用小键盘输入。

```
#include <stdio.h>

int main()
{
    int secret = 0;
    printf("Please enter the secret num:");
    scanf("%d",&secret);
    if (secret != 123)
    {
        printf("Incorrect secret num.\n");
        return 0;
    }
    printf("Hello world!\n");
    return 0;
}
```

代码很简单，先要求用户输入一个数字，然后判断这个数字是否是“123”，如果不是则输出“Incorrect secret num.”，如果是则输出“Hello world!”。输入完成后按“esc”键退出编辑模式，输入:wq命令保存并退出Vim，然后使用Clang编译。

```
$ clang cm01.c -o cm01
```

如果没有编译错误，就会生成cm01可执行文件；如果有错误，可以重新执行vim cm01.c命令，回到Vim中编辑代码的错误之处。然后测试一下CrackMe运行是否正常。

```
$ ./cm01
Please enter the secret num:456
Incorrect secret num.
$ ./cm01
Please enter the secret num:123
Hello world!
$
```

当输入的数字不是“123”的时候，程序输出“Incorrect secret num.”，如果是“123”则输出“Hello world！”，说明程序运行正常。

接下来就要破解这个简单的CrackMe了，让它在我们输入任意值的时候都会输出“Hello world！”。

## 1.3 使用 HT Editor 进行破解

终于到破解环节了！如果这是你初次接触Mac上的破解，那么下面就将是第一次亲手破解的程序。

这一节会用到少量的汇编指令和Mach-O文件格式的相关知识，后面的章节会详细讨论Mach-O文件格式和x86\_64汇编指令，这里我们只需要知道几条关键汇编指令的含义即可，其余的都让HT Editor帮我们进行处理。

- **jz**指令：跳转指令，可以理解成如果前面比较指令的比较结果相同则跳转到指定的地址，如果不相等就不跳转，继续执行它下面的指令；
- **jnz**指令：与jz指令正好相反，不相等则跳转；
- **jmp**指令：不管任何情况都会进行跳转；
- **call**指令：调用过程指令，一般对应高级语言中的函数调用。

这些跳转指令的作用是什么呢？我们用高级语言所写的代码，最终都会被编译器翻译成机器码，其中的判断语句一般翻译成跳转指令，所以我们可以修改跳转指令达到修改软件的执行流程的目的，最终让目标程序跳过检查，执行我们所期望的代码。

有了理论基础，下面就该开始动手实践了。首先用HT Editor打开要破解的程序，也就是我们自己写的cm01。首先打开一个终端，并通过cd命令切换到cm01所在的文件夹，然后在终端中运行ht命令，会启动HT Editor的命令行界面，如图1-6所示。



图1-6 HT Editor主界面

HT Editor在POSIX兼容系统上有些按键会有问题，需要用esc代替ctrl键，所以主菜单就变成了esc+红色字母，而下面的1到0快捷键分别对应的是fn+F1到fn+F10。首先按fn+F3（open），会出现选择文件的界面，如图1-7所示。

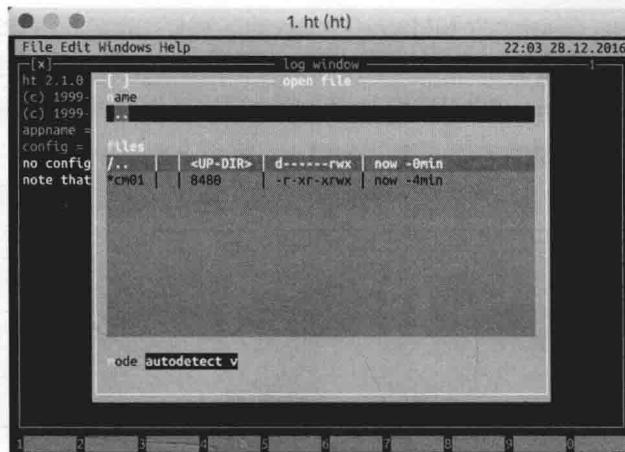


图1-7 选择文件

然后按tab键将光标移到下面的列表框，找到cm01并回车，就会出现十六进制编辑界面，如图1-8所示。

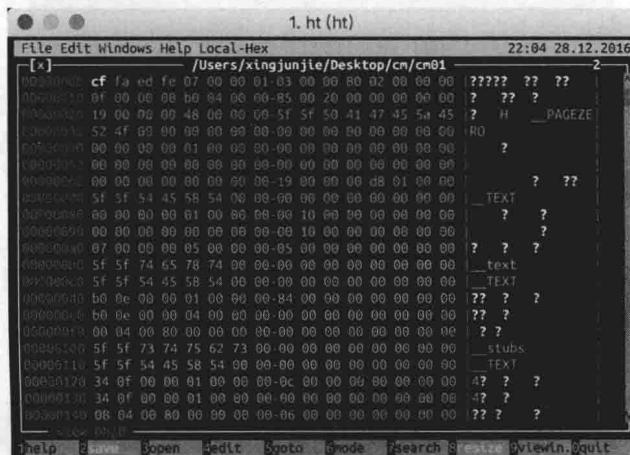


图1-8 十六进制编辑

现在我们要把它切换到反汇编界面，按fn+F6 ( mode ) 出现select mode对话框，如图1-9所示。

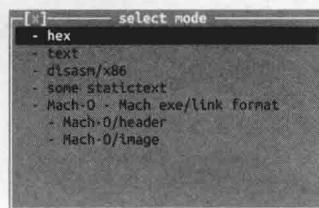


图1-9 select mode

选择Mach-O/image。disasm/x86和Mach-O/image的区别是，前者把整个文件当成二进制文件从头开始反汇编，而后者则会根据Mach-O文件格式对代码段进行反汇编。关于Mach-O文件格式将会在第5章进行详细的讨论，这里只需要知道为什么要这样做就可以了。

选择之后，HT会显示代码段的反汇编，如图1-10所示。

```

1. ht (ht)
File Edit Windows Help Analyser /Users/xingjunlei/Desktop/cm/cm01 22:07 28.12.2016
[x] 0x100000eb0 push ebp
main:
    push    rbp
    mov     rbp, rsp
    sub    rsp, 28h
    lea     rdi, [strz_Please_enter_the_secret_num:_1000
    mov     dword ptr [rbp-], 0
    mov     dword ptr [rbp-], 0
    mov     al, 0
    call    wrapper_100001010_100000f34
    lea     rdi, [data_100000f81]
    lea     rsi, [rbp-8]
    mov     [rbp-1ch], eax
    al, 0
    call    wrapper_100001018_100000f3a
    cmp    dword ptr [rbp-8], 7bh
100000ee9

```

图1-10 HT反汇编界面

有了反汇编代码，我们就需要阅读反汇编并找到判断数字是否正确的关键跳，并对其进行修改。HT为我们识别出了main函数，我们可以顺着main函数理清逻辑，这个CrackMe很简单，我们这么做也不是特别费力，但是如果代码比较复杂，这种方法就不适用了。

还有更简单的方法，当我们输入一个错误的数字的时候，程序会提示一句“Incorrect secret num”，我们可以用这句提示作为突破口。HT提供了一个搜索（search）功能，我们可以使用这个功能来查找这句错误提示。

首先，按快捷键fn+F7，出现搜索对话框，mode选择“display: regex”，表示使用正则表达式搜索HT界面中显示的字符，e为搜索的正则表达式，不过我们目前还用不到复杂的正则表达式，只需要输入“incorrect”就可以了。选中“case insensitive（不区分大小写）”，如图1-11所示。

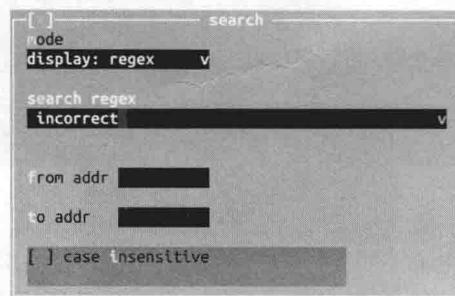


图1-11 HT搜索对话框

然后按回车，光标会定位到下面这行代码：

```
100000ee9 ! lea rdi, [strz_Incorrect_secret_num._100000f78]
```

这句汇编代码的意思是将100000f78这个地址放入rdi寄存器，100000f78中的内容是一个以0结尾的字符串(strz)，内容为“Incorrect secret num.”，HT自动将内容中的空格换成了下划线。按shift+fn+F7可以继续搜索，但是在Mac上这个快捷键似乎无效。

我们到100000f78这个地址这里就可以看到“Incorrect secret num.”这个字符串。找到了这个字符串，那它上面的代码就应该是判断的跳转，往上一看，它上面就是如下代码：

```
100000ee3 ! jz loc_100000f06
```

到100000f06处进行查看，会发现如下代码：

```
100000f06 !
..... ! loc_100000f06: ;xref j100000ee3
..... ! lea rdi, [strz_Hello_world__100000f8f]
100000f0d ! mov al, 0
100000f0f ! call wrapper_100001010_100000f28
```

这里将“Hello world”字符串的地址传给了一个函数，不难猜测，这个函数应该就是printf，到这里就应该是成功了。地址100000ee3处的跳转就是跳向成功的关键跳，也就是我们要修改的地方。

回到100000ee3处，按快捷键ctrl+a修改这里的汇编代码，修改成jnz loc\_100000f06，将原来的相等则跳转改成不相等则跳转，如图1-12所示。

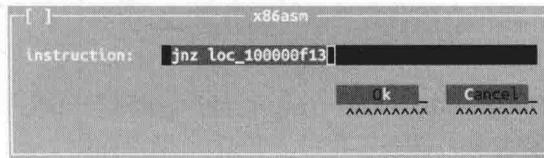


图1-12 HT修改汇编代码对话框

回车后，HT会列出所写的汇编代码可以翻译成的机器码，因为x86和x64一条汇编语句可能对应多种机器码，这里选择和原来机器码长度相同的机器码最合适，既不需要填充nop指令，也不会覆盖后面的机器码。HT默认为我们选中了最合适的机器码，直接回车即可，如图1-13所示。

可以看到，实际上只修改了一个字节的机器码（变红的85），但此时的修改还没有保存到磁盘文件，按fn+F2进行保存。保存完后，红字就消失了，我们在来试试下面这个CrackMe程序。

```
$ ./cm01
Please enter the secret num:456
Hello world!
$ ./cm01
Please enter the secret num:123
Incorrect secret num.
```

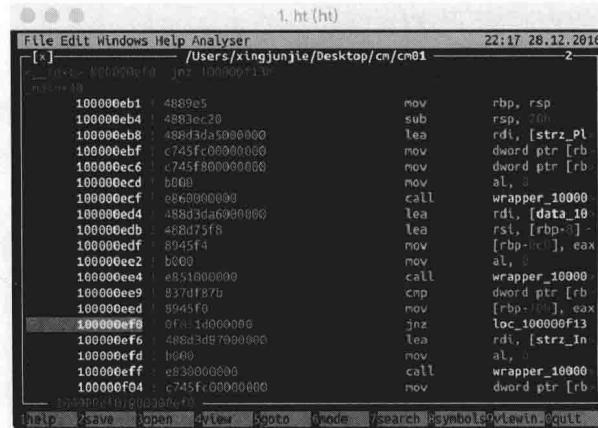


图1-13 HT修改结果

如果我们输入一个错误的数字，会输出“Hello world!”，输入正确的反倒提示失败了，这说明我们的修改成功了！

不过还是有点不完美，我们希望输入正确的数字也要提示成功。虽然在实际使用中乱猜正好输入了正确的数字可能性太低了，但是我们要在技术上追求完美。回到HT，找到跳转指令，按ctrl+a快捷键，这次我们把jnz改成jmp，让它无论数字正确与否都直接跳到打印“Hello world!”处，回车保存，再次测试如下。

```
$ ./cm01
Please enter the secret num:74551122
Hello world!
$ ./cm01
Please enter the secret num:123
Hello world!
$ ./cm01
Please enter the secret num:6697
Hello world!
```

哈！不管输入什么都会提示成功了！第一次的破解之旅到此完美结束！

## 1.4 本章小结

本章介绍了最基本的分析环境搭建和一个简单程序的破解。通过本章的学习，相信你已经对macOS系统上的软件逆向分析技术有基本的认识了。

# 系统安全架构

macOS虽然属于UNIX系的操作系统，但在安全性方面，有着自己独特的系统架构与安全保护措施。苹果系统真正的变革是在迈入版本10之后，从10.0到10.6，系统都以Mac OS X命名，并且以大型猫科动物的名字作为系统代号。从版本10.7到10.11，系统又改名为OS X。到了版本10.12时，更名为macOS。每一次系统的重命名，都代表着系统全新的变化，每一次系统的跨版本更新，都会带来许多新的安全特性。从版本10.6 Snow Leopard雪豹以后，OS X正式放弃了PowerPC架构，进入了x86-64时代。本章主要探讨苹果系统在该版本之后引入的安全特性。

## 2.1 系统架构概述

在一些公开的演讲与技术报告中，苹果公司向公众展示了一些系统架构方面的信息。整个操作系统按照分层的方式进行组织，一种典型的针对非开发人员的分层方式将系统分成了以下4层。

- User Experience：用户体验层。包括Aqua、Dashboard、Spotlight、Dock、输入法、屏幕保护、Accessibility（辅助功能）、语音、位置与地图、搜索等内容。
- Application Framework：应用框架层。包括Cocoa，用于OS X系统上应用程序的开发。
- Graphics and Media：图形和媒体层。包括核心框架、OpenAL、OpenGL、Quartz、SceneKit、SpriteKit等。
- Darwin：系统核心层。包括系统内核及shell环境等。

Darwin（达尔文）是苹果公司开发的一套UNIX实现，它继承了UNIX上一些传统的特性，如shell环境、目录结构、文件权限等。

在macOS开发文档中，展示了另一种针对开发人员的分层方式，如下所示。

- Cocoa Layer：Cocoa框架层。包括了用于开发界面程序的框架集合。
- Foundation Layer：基础框架。提供了程序开发时使用到的基础数据类型、数值处理、网络/文件IO、日期等方方面面的接口。可以说，开发可视的Mac App基本离不开它。
- Media Layer：媒体层。提供了图像、声音、视频、动画及游戏开发需要的接口。

- Core Service Layer：核心服务层。提供了系统安全、底层、内部数据访问及存储的接口。如AddressBook用于访问地址簿、CoreData用于数据存储、QuickLook用于快速浏览插件开发。另外，CoreFoundation框架属于这一层。
- Core OS Layer：核心系统层。包括加速器、蓝牙、异常处理、网络扩展、系统配置等框架。
- Kernel & Driver Layer：内核与驱动层。包括开发设备驱动程序与内核扩展所需的一些框架。

对于一般的开发人员而言，内核与驱动层基本不会用到，而前面5层框架在开发不同场景的应用时，或多或少会有所涉及。本书将在第3章中着重探讨前3层框架。

### 2.1.1 shell环境

Linux用户对于shell环境并不陌生，大多数Linux上的命令行工具在macOS中也可以使用。如ls命令列目录。打开系统自带的终端程序，执行命令ls -l /bin/\*sh会输出系统支持的shell如下：

```
$ ls -l /bin/*sh
-r-xr-xr-x 1 root wheel 628496 Dec 3 14:36 /bin/bash
-rwxr-xr-x 1 root wheel 378624 Dec 3 14:36 /bin/csh
-r-xr-xr-x 1 root wheel 1394432 Dec 3 14:36 /bin/ksh
-r-xr-xr-x 1 root wheel 632672 Dec 3 14:36 /bin/sh
-rwxr-xr-x 1 root wheel 378624 Dec 3 14:36 /bin/tcsh
-rwxr-xr-x 1 root wheel 573600 Dec 3 14:36 /bin/zsh
```

### 2.1.2 目录结构

在macOS系统中保留着很多与UNIX相同的目录。如/usr/bin目录中存放着用户安装的命令行工具，/etc目录存放着系统的配置信息。典型的macOS系统目录如下：

```
$ ls -l /
drwxrwxr-x+ 157 root admin 5338 Mar 2 11:45 Applications
drwxr-xr-x+ 63 root wheel 2142 Oct 22 15:19 Library
drwxr-xr-x@ 2 root wheel 68 Oct 9 11:41 Network
drwxr-xr-x@ 4 root wheel 136 Jan 23 11:35 System
drwxr-xr-x 7 root admin 238 Jan 5 11:10 Users
drwxrwxrwt@ 6 root admin 204 Mar 1 21:10 Volumes
drwxr-xr-x@ 39 root wheel 1326 Jan 23 11:34 bin
drwxrwxr-t@ 2 root admin 68 Oct 9 11:41 cores
dr-xr-xr-x 3 root wheel 7856 Feb 28 19:30 dev
lrwxr-xr-x@ 1 root wheel 11 Oct 9 11:40 etc -> private/etc
dr-xr-xr-x 2 root wheel 1 Feb 28 19:30 home
dr-xr-xr-x 2 root wheel 1 Feb 28 19:30 net
drwxrwxr-x@ 5 root wheel 170 May 31 2015 opt
drwxr-xr-x@ 6 root wheel 204 Oct 9 11:41 private
drwxr-xr-x@ 59 root wheel 2006 Jan 23 11:34 sbin
lrwxr-xr-x@ 1 root wheel 11 Oct 9 11:40 tmp -> private/tmp
drwxr-xr-x@ 12 root wheel 408 Feb 24 13:04 usr
lrwxr-xr-x@ 1 root wheel 11 Oct 9 11:40 var -> private/var
```

macOS在UNIX的基础上，在根目录下为自己添加了部分特有的目录。