

# 目录

前言	1.1
iOS安全概览	1.2
iOS系统安全	1.3
iOS安全防护	1.4
代码混淆	1.4.1
Obfuscator-LLVM	1.4.1.1
iOS Class Guard	1.4.1.2
防止导出头文件	1.4.2
代码动态运行	1.4.3
SSL证书绑定	1.4.4
附录	1.5
参考资料	1.5.1

# iOS安全与防护

- 最新版本: v0.8
- 更新时间: 20211021

## 简介

介绍iOS安全与防护。包括iOS安全在安全领域所属的范畴，iOS系统本身的安全设计，以及iOS的安全的具体防护手段。包括但不限于代码角度的代码混淆、防止逆行导出头文件、代码动态执行、以及抓包相关的SSL证书绑定等手段。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### HonKit源码

- [crifan/ios\\_security\\_protect: iOS安全与防护](#)

### 如何使用此HonKit源码去生成发布为电子书

详见：[crifan/honkit\\_template: demo how to use crifan honkit template and demo](#)

### 在线浏览

- [iOS安全与防护 book.crifan.org](#)
- [iOS安全与防护 crifan.github.io](#)

### 离线下载阅读

- [iOS安全与防护 PDF](#)
- [iOS安全与防护 ePUB](#)
- [iOS安全与防护 Mobi](#)

## 版权和用途说明

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如有版权，请通过邮箱联系我 [admin 艾特 crifan.com](mailto:admin@crifan.com)，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 crifan 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan\\_ebook\\_readme: Crifan的电子书的使用说明](#)

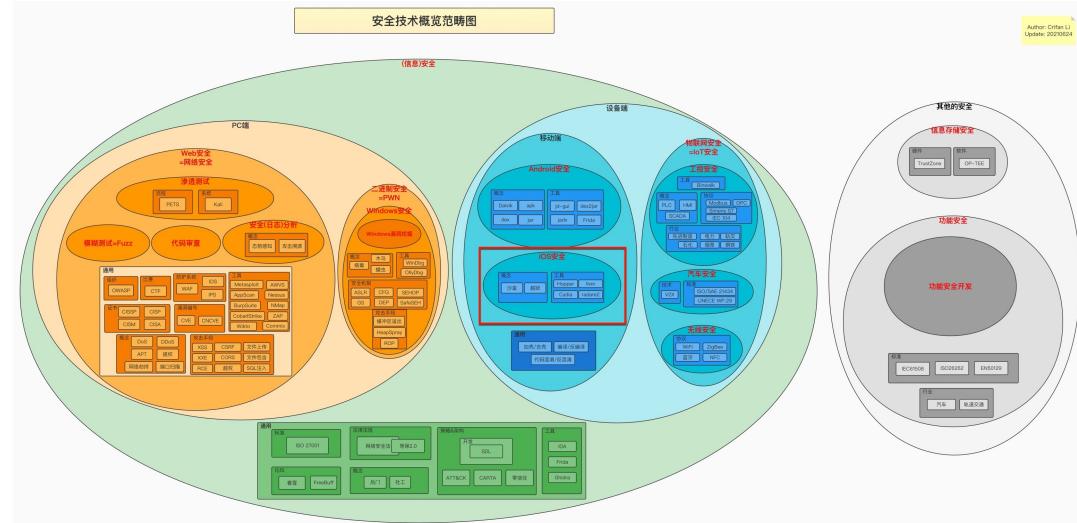
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 16:37:11

# iOS安全概览

- iOS安全

- 所属范畴

- 从大的信息安全范畴，属于： 信息安全 中 设备端 中 移动端 中 iOS安全



- 详见：[信息安全概览](#) [安全概览](#)
- iOS开发，从 安全 领域的 攻防 角度分：
  - 反方的 攻 : iOS逆向 = iOS破解 = iOS攻击
  - 正方的 防 : iOS安全 = iOS防护

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 14:41:20

# iOS系统安全

- iOS系统安全 = iOS操作系统级别的安全
  - 概述
    - 总体上说，iOS系统比Android系统更安全
      - 当然也更封闭
      - 当然，安全只是相对的，并没有绝对的安全
        - 高级黑客还是可以破解和黑你的iPhone的
  - 详解
    - 技术层面
      - iOS系统本身
        - 安全设计=安全机制
          - 可信启动链
          - 代码签名
          - 沙盒执行环境
          - 权限隔离和数据加密
          - 更严格的版本控制
            - 不能降级(安装低版本的iOS操作系统)
              - 该策略使得iOS设备一旦升级后，就只能停留在当前或者最新版本
              - 有效避免了操作系统版本碎片化问题，减少了已公开漏洞的影响范围
        - 严格掌控的应用市场
          - 杜绝向第三方应用开放高级数据访问权限，限制了iOS恶意应用的传播和能力

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 11:52:29

# iOS安全防护

iOS的逆向和破解，与之相对的正向的防护，叫：[ios安全和防护](#)

- iOS安全防护的 手段 =防护的角度=具体涉及内容
  - 代码角度： 代码混淆
    - 被反编译后，也只能看到 乱码的函数
    - 部分防止被破解，被猜测到核心逻辑
  - 被逆向导出头文件
    - 把 ObjC 换 Swift，以增加被破解难度？
  - 把核心逻辑和代码，变成动态下载再运行
  - 其他相关
    - 抓包角度：做SSL证书的 ssl pinning
    - 甚至额外做本地证书校验

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-21 16:35:51

# 代码混淆

iOS安全从代码角度，可以去做： 代码混淆

- 代码混淆 目的
  - 更好地保护代码， 增加逆向破解难度 = 不被轻松地恶意分析破解
- 代码混淆 优缺点
  - 优点
    - 无须变动项目源码
    - 功能灵活可选， 根据需要自由组合
  - 缺点
    - 导致安装包体积增大
    - 混淆后的代码有会被编译优化掉的风险
    - 提交审核（AppStore）存在被拒的风险
- 代码混淆方式=子功能模块
  - obfuscator 类
    - bcf = Bogus Control Flow =虚假块=伪控制流
    - fla = cff = Control Flow Flattening =控制流展开=控制流平坦化
    - split =基本块分割
    - sub = Instructions Substitution =指令膨胀=指令替换
    - acd =anti class-dump=反class-dump
  - indibran =indirect branch=基于寄存器的相对跳转， 配合其他加固可以彻底破坏IDA/Hopper的伪代码(俗称F5) = 故意制造堆栈不平衡， 不能F5， 函数内利用寄存器跳转BX12
  - strcry =字符串加密
  - funcwra =函数封装
  - 插入垃圾指令
- iOS代码混淆工具
  - Obfuscator-LLVM = ollvm
  - iOS Class Guard
  - Hikari
    - 基于ollvm
    - 主页
      - <https://github.com/HikariObfuscator/Hikari>
      - <https://github.com/HikariObfuscator/Hikari/wiki>
- iOS代码混淆后的效果
  - 导出头文件后， 函数名变乱码
    - 比如：
      - money 变成 xadsf32
      - showMoney 变成 AFAdsa123
  - iOS逆向后看到的代码中的函数， 都是无名的函数
    - 比如： Hopper逆向app后， 有很多函数名都是： sub\_xxx， 就表示， 该函数被混淆了

```

0000000100050134    ret
; endp

loc_1000000100050138:
0000000100050138    ldr    x9, [x0]
000000010005013c    and   x9, x9, #0xffff
0000000100050140    add   x10, x9, #0x10
0000000100050144    bic   x9, x9, x9
0000000100050148    add   x9, x9, x9
000000010005014c    ret
; endp

; ===== BEGINNING OF PROCEDURE =====

-[_TTC7wordios1HttpManager copy]; // -[wordios.HttpManager copy]
0000000100050150    b     -[_TTC7wordios1HttpManager mutableCopy]; -[wordios.HttpManager mutableCopy], Objective C Implementation define
; endp

; ===== BEGINNING OF PROCEDURE =====

sub_1000000100050154:
0000000100050154    b     sub_10004d9dc+4456
; endp

; ===== BEGINNING OF PROCEDURE =====

-[_TTC7wordiosMainCon items]; // -[wordios.MainCon items]
0000000100050158    adrp   x2, #0x1000dc000
000000010005015c    add    x2, x2, #0x100
0000000100050160    adrp   x3, #0x1000dc000
0000000100050164    add    x3, x3, #0xfc0
0000000100050168    adrp   x4, #0x1000dc000
000000010005016c    add    x4, x4, #0x100
0000000100050170    b     sub_1000581ac
; endp

; ===== BEGINNING OF PROCEDURE =====

-[_TTC7wordiosMainCon setItems]; // -[wordios.MainCon setItems]
0000000100050174    adrp   x3, #0x1000dc000
0000000100050178    add    x3, x3, #0xfc0
000000010005017c    adrp   x4, #0x1000dc000
; endp

```

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 15:37:19

# Obfuscator-LLVM

此处介绍，可以用于iOS代码混淆的工具：[Obfuscator-LLVM](#)

- `Obfuscator-LLVM = ollvm`
  - 是什么：基于LLVM的代码混淆工具
  - 谁开发的：瑞士伊夫尔东莱班的应用科学与艺术大学信息安全小组
  - 什么时候：2010年6月
  - 目的：增强软件代码安全
    - 基于LLVM的编译套件
    - 通过防篡改(tamper-proofing)和代码混淆(code obfuscation)
  - 支持语言
    - C, C++, Objective-C, Ada 和 Fortran
  - 支持架构
    - x86, x86-64, PowerPC, PowerPC-64, ARM, Thumb, SPARC, Alpha, CellSPU, MIPS, MSP430, SystemZ 和 XCore
  - 代码混淆方式
    - control flow flattening = 控制流扁平化 = 控制流平坦化
      - 语法：`-mllvm -fla`
    - instruction substitution = 指令替换
      - 语法：`-mllvm -sub`
    - bogus control flow = 控制流伪造 = 虚假控制流程
      - 语法：`-mllvm -bcf`
  - 资料
    - GitHub
      - obfuscator-llvm/obfuscator
        - <https://github.com/obfuscator-llvm/obfuscator>
      - 文档入口
        - Home · obfuscator-llvm/obfuscator Wiki
          - <https://github.com/obfuscator-llvm/obfuscator/wiki>
      - 快速上手
        - obfuscator/GettingStarted.rst at llvm-4.0 · obfuscator-llvm/obfuscator
          - <https://github.com/obfuscator-llvm/obfuscator/blob/llvm-4.0/docs/GettingStarted.rst>

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-21 15:35:09

# iOS Class Guard

- Github
  - Polidea/ios-class-guard: Simple Objective-C obfuscator for Mach-O executables.
    - <https://github.com/Polidea/ios-class-guard>

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 15:38:10

## 防止导出头文件

iOS的ObjC的特性，使得有机会，在app运行期间，导出ObjC的类的头文件，得到众多的iOS的类的函数定义和具体属性。

而如果把iOS的 `objc` 代码，改为 `Swift` 代码，就可以避免被导出类的头文件，增加了破解难度。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-21 14:51:23

## 代码动态执行

把iOS的app中，核心的代码，比如核心通信逻辑、加密算法、参数生成逻辑，放到动态代码，即从服务器端动态下载要运行的代码，然后在本地运行核心逻辑，生成要的值。

此种iOS安全防护手段，一般称为： 代码动态执行

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-21 14:53:37

# SSL证书绑定

- 背景
  - iOS的app，多数和服务器端通信，也是基于常见 HTTP / HTTPS 协议。
  - 而iOS逆向中往往用，用抓包工具，比如 Charles 去抓包分析你的网络请求。
    - 其中因为抓包可以直接看到明文数据而多数已不用 HTTP 了。
    - 而改用加密的 HTTPS ，而抓包 HTTPS ，一般来说无法直接看到明文数据，只能看到加密后的乱码。
    - 但是采用了根证书信任等手段，往往也可以抓包到 HTTPS 的明文。
    - 而最新的手段一般是：采用 证书绑定 = SSL pinning ， app内部会对于SSL的证书和本地的证书做绑定和校验，使得抓包工具比如Charles的证书，无法通过验证，从而导致无法抓包到 HTTPS 的明文。
  - 而iOS防护的话，不希望被抓包，被看到HTTPS的明文，所以往往也会去采用： 证书绑定 = SSL pinning
    - 而证书绑定中，更高级和更严格一点的手段是： 本地证书校验 ?
      - 比如 抖音 内部就做了证书校验
      - 注：只有hook了证书校验的部分，才能绕过校验，才能实现抓包 HTTPS 看到明文数据。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-21 15:02:46

## 附录

下面列出相关参考资料。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 11:52:29

## 参考资料

- [Class-dump导出头文件没有属性和方法名怎么处理 - 技能讨论 - 睿论坛](#)
- [iOS逆向攻防实战 - 掘金 \(juejin.cn\)](#)
- [\[原创\]某App去混淆-iOS安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com](#)
- [iOS代码混淆工具 — Hikari（支持Xcode14以下全部版本混淆）-Apibug](#)
- [基于llvm的iOS代码混淆工具 -- Hikari - 简书](#)
- [ios手动代码混淆函数和变量名基本原理和注意事项教程\(含demo\)\\_小手琴师的博客-CSDN博客\\_xcode代码混淆](#)
- [iOS混淆--OLLLVM在iOS中的实践（逻辑混淆） - 简书](#)
- 

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 16:27:19