目录

防止iPhone被黑：iOS安全

- 最新版本：`v0.6`
- 更新时间：`20210525`

简介

介绍iOS安全，防止你的iPhone被黑。先介绍iOS安全的概览。再介绍iOS操作系统层面的安全机制。再整理iOS的安全和破解相关的工具和技术，包括Mac的逆向工具Hopper Disassembler；Linux的编译套件LLVM及相关的LLDB和Obfuscator-LLVM；Linux的调试器lldb-server、逆向工具Miasm、radare2及相关的Cutter；以及iOS的各种工具，包括正向的混淆工具ios-class-guard；破解方面，调试器debugserver，插件开发的MonkeyDev，苹果二进制格式Mach-O的处理工具class-dump、MachOView、jtool、otool等；砸壳工具bfinject、Clutch、Dumpdecrypted、frida-ios-dump；越狱工具Cydia Substrate、frida、Electra、unc0ver。最后给出参考资料。

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：

Gitbook源码

- crifan/prevent_iphone_hacked_ios_security: 防止iPhone被黑：iOS安全

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 防止iPhone被黑：iOS安全 book.crifan.com
- 防止iPhone被黑：iOS安全 crifan.github.io

离线下载阅读

- 防止iPhone被黑：iOS安全 PDF
- 防止iPhone被黑：iOS安全 ePub
- 防止iPhone被黑：iOS安全 Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

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

iOS安全概览

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

iOS安全与破解工具

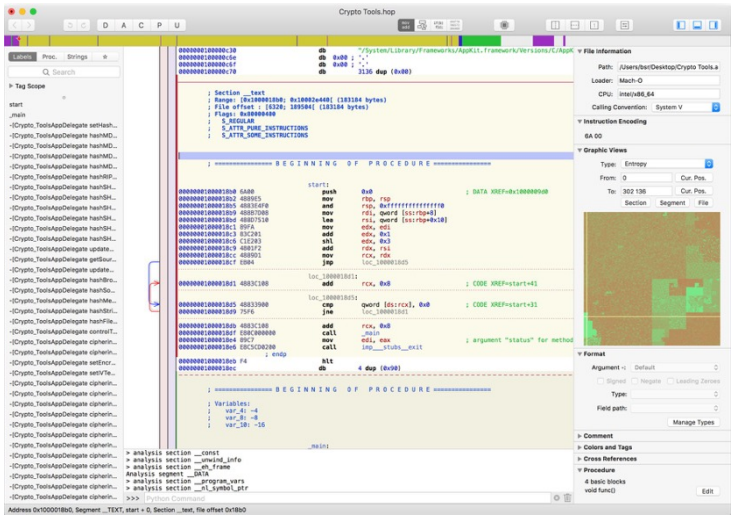
- iOS安全
- 相关
  - 说明：由于iOS安全和破解往往涉及到，在Mac和Linux中使用相关工具，所以iOS安全，往往也涉及到Mac和Linux的安全
  - 常用工具
    - Mac
      - `Hopper Disassemble`
    - Linux
      - `LLDB`
      - `GDB`

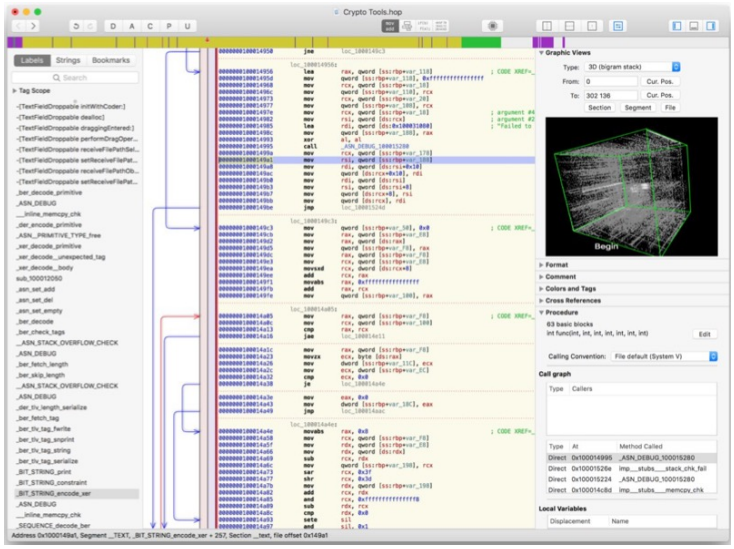
下面详细介绍具体包含哪些常用工具。

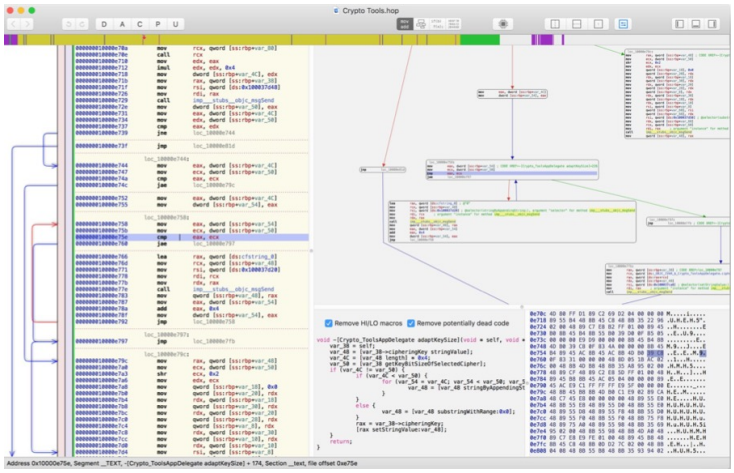
Mac安全

逆向工具

Hopper Disassembler

- `Hopper Disassembler`
  - = `Hopper` = `hd`
  - 是什么：Hopper is a reverse engineering tool for OS X and Linux
  - 一句话描述：
    - the reverse engineering tool that lets you disassemble, decompile and debug your applications
  - 功能：disassemble, and decompile
  - 支持平台、架构：32/64bits Intel Mac, Linux, Windows and iOS executables
  - 详解
    - This tool will let you disassemble any binary you want, and provide you all the information about its content, like imported symbols, or the control flow graph! Hopper can retrieve procedural information about the disassembled code like the stack variables, and lets you name all the objects you want.
  - 主要用于：static binary analyses
  - 官网
    - https://www.hopperapp.com
  - 截图
    - 





- 对比：
  - Hopper vs IDA
    - 平台支持
      - Hopper：更倾向于 `Mac`
      - IDA：支持多平台： `Windows` 、 `Linux` 、 `Mac`

Linux

编译套件

LLVM

- `LLVM`
  - = `Low Level Virtual Machine`
  - 是什么=一句话描述
    - 一套用于构建出高度优化的编译器、优化器、运行环境的工具集合
      - a toolkit for the construction of highly optimized compilers, optimizers, and runtime environments.
  - 主要包含3个部分
    - `LLVM套件` = `LLVM Suite`
      - 包含各种
        - 工具
          - `汇编器` = `assembler`
          - `反汇编器` = `disassembler`
          - `位码分析器` = `bitcode analyzer`
          - `位码优化器` = `bitcode optimizer`
          - 简单的回归测试
            - 用于测试LLVM工具和Clang前端
        - 库
        - 头文件
    - `Clang` = `Clang前端` = `Clang front end`
      - 是什么：LLVM的内置的原生的 `C` / `C++` / `Objective-C` 编译器
      - 可以把 `C`，`C++`，`Objective-C` 和 `Objective-C++` 的代码，编译成 `LLVM bitcode`
        - 然后就可以用LLVM套件去操作此（编译后的）程序了
    - `测试套件` = `Test Suite`
      - 一堆工具的集合
        - 测试LLVM的功能和性能
  - 子项目
    - `LLVM Core` libraries
      - a modern source- and target-independent optimizer, along with code generation support for many popular CPUs
    - `Clang`
      - an `LLVM native` C/C++/Objective-C compiler
    - `LLDB`
      - a great native debugger
        - 基于 `LLVM` 和 `Clang`
    - `libc++` 和 `libc++ ABI`
      - a standard conformant and high-performance implementation of the C++ Standard Library
        - including full support for C++11 and C++14
    - `compiler-rt`
      - provides highly tuned implementations of the low-level code generator
    - `MLIR`

            - a novel approach to building reusable and extensible compiler infrastructure
        - `OpenMP`
            - an OpenMP runtime for use with the OpenMP implementation in Clang
        - `polly`
            - a suite of cache-locality optimizations as well as auto-parallelism and vectorization using a polyhedral model
        - `libclc`
            - implement the OpenCL standard library
        - `klee`
            - implements a "symbolic virtual machine" which uses a theorem prover to try to evaluate all dynamic paths through a program in an effort to find bugs and to prove properties of functions
        - `LLD`
            - a new linker
                - a drop-in replacement for system linkers and runs much faster
    - 资料
        - 官网
            - The LLVM Compiler Infrastructure Project
                - https://llvm.org
        - 快速上手
            - Getting Started with the LLVM System — LLVM 12 documentation
                - https://llvm.org/docs/GettingStarted.html
    - 相关
        - 概念
            - `IR` = `Intermediate Representation` = `中间表示层`

LLDB

- `LLDB`
  - = `LLDB Debugger`
  - 是什么：调试器
  - 背景
    - LLVM项目
      - 旗下有多个组件=功能模块
        - 其中有个 `调试器` = LLDB Debugger
  - 一句话描述：一个新一代的高性能的调试器
    - a next generation, high-performance debugger
  - 与之相关
    - 编译器：`Clang`
    - 反汇编器：`llvm disassembler`
  - 应用
    - Mac中XCode的默认的调试器
      - 支持语言：`C`，`Objective-C` 和 `C++`
      - 支持平台：`Mac` 和 `iOS`
  - 支持众多平台和系统

Features matrix [4]

Feature	FreeBSD	Linux	macOS	Windows
Backtracing	✓	✓	✓	✓
Breakpoints	✓	✓	✓	✓
C++11:	✓	✓	✓	?
Command-line lldb tool	✓	✓	✓	✓
Core file debugging	✓	✓	✓	✓
Debugserver (remote debugging)	Not ported	Not ported	✓	Not ported
Disassembly	✓	✓	✓	✓
Expression evaluation	?	Works with some bugs	✓	Works with some bugs
JIT debugging	?	Symbolic debugging only	Untested	✘
Objective-C 2.0:	?	N/A	✓	N/A

  - 常见命令举例
    - `run`
    - `break set -n main`
    - `bt`
    - `register read`
    - `di -n main`
  - 资料
    - 官网
      - LLDB Homepage — The LLDB Debugger
        - http://lldb.llvm.org
    - 教程
      - Tutorial — The LLDB Debugger
        - https://lldb.llvm.org/use/tutorial.html
    - LLDB和GDB命令对比
      - GDB to LLDB command map — The LLDB Debugger
        - https://lldb.llvm.org/use/map.html

Obfuscator-LLVM

- `Obfuscator-LLVM`
  - 别称：`OLLVM`
  - 是什么：针对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

调试器

lldb-server

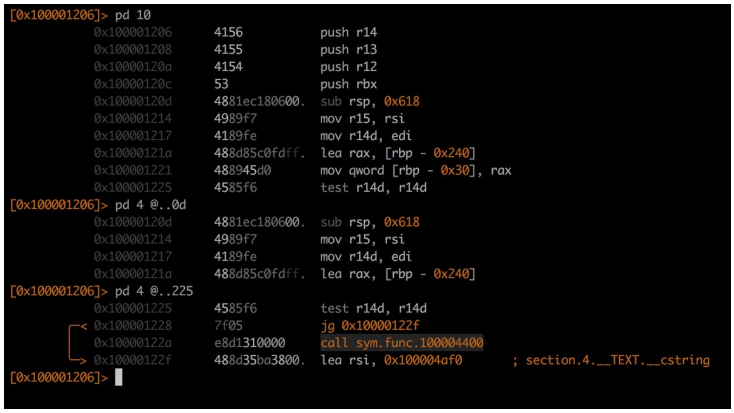
- 远程调试
  - 分2个端
    - `lldb client`
      - 运行在 local system
        - 比如 Linux，Mac
    - `lldb server`
      - 不同平台
        - `Linux` 和 `Android`： `lldb-server`
          - 不依赖于 `lldb`
            - 因为：已静态链接包含了 `LLDB` 的核心功能
            - 对比： `lldb` 是默认是动态链接 `liblldb.so`
        - `Mac` 和 `iOS`： `debugserver`
      - 运行在 remote system
      - 实现了remote-gdb的功能
    - 两者通讯
      - 用的是： `gdb-remote` 协议
        - 一般是在TCP/IP之上运行
  - 细节详见：
    - `docs/lldb-gdb-remote.txt`
  - 资料
    - 主页
      - Remote Debugging — The LLDB Debugger
        - http://lldb.llvm.org/use/remote.html

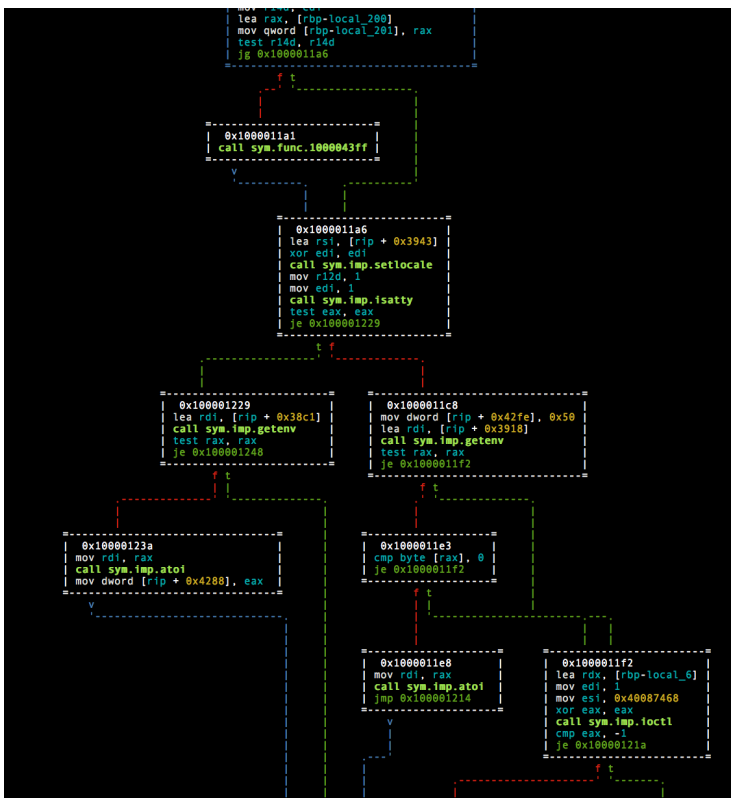
逆向工具

Miasm

- 一句话介绍：用Python实现的逆向工程框架
  - Reverse engineering framework in Python
- 用途：分析、修改、生成 二进制程序
  - analyze / modify / generate binary programs
- 特性
  - Opening / modifying / generating PE / ELF 32 / 64 LE / BE
  - Assembling / Disassembling X86 / ARM / MIPS / SH4 / MSP430
  - Representing assembly semantic using intermediate language
  - Emulating using JIT (dynamic code analysis, unpacking, ...)
  - Expression simplification for automatic de-obfuscation
- 资料
  - GitHub
    - cea-sec/miasm: Reverse engineering framework in Python
      - https://github.com/cea-sec/miasm
  - 官网
    - Home — Miasm's blog
      - https://miasm.re/blog/

radare2

- `radare2`
  - 是什么：一个著名的开源逆向工程平台
    - Unix-like reverse engineering framework and commandline tools
  - 评价
    - 可谓是一大神器
      - 支持包括反汇编、分析数据、打补丁、比较数据、搜索、替换、虚拟化等等，同时具备超强的脚本加载能力，并且可以运行在几乎所有主流的平台
  - 竞品
    - `IDA`
  - 截图
    - 

    - 

  - 支持平台

        - Mac
        - Windows
        - Linux
        - Solaris
        - Android
        - iOS
        - Haiku
    - 历史
        - Radare project started as a forensics tool, a scriptable commandline hexadecimal editor able to open disk files
        - but later support for analyzing binaries, disassembling code, debugging programs, attaching to remote gdb servers
    - 功能：Radare is a portable reversing framework that can
        - Disassemble (and assemble for) many different architectures
        - Debug with local native and remote debuggers (gdb, rap, webui, r2pipe, winedbg, windbg)
        - Run on Linux, *BSD, Windows, OSX, Android, iOS, Solaris and Haiku
        - Perform forensics on filesystems and data carving
        - Be scripted in Python, Javascript, Go and more
        - Support collaborative analysis using the embedded webserver
        - Visualize data structures of several file types
        - Patch programs to uncover new features or fix vulnerabilities
        - Use powerful analysis capabilities to speed up reversing
        - Aid in software exploitation
    - 特性
        - Batch, commandline, visual and panels interactive modes
        - Embedded webserver with js scripting and webui
        - Assemble and disassemble a large list of CPUs
        - Runs on Windows and any other UNIX flavour out there
        - Analyze and emulate code with ESIL
        - Native debugger and GDB, WINDBG, QNX and FRIDA
        - Navigate ascii-art control flow graphs
        - Ability to patch binaries, modify code or data
        - Search for patterns, magic headers, function signatures
        - Easy to extend and modify
        - Commandline, C API, script with r2pipe in any language
    - 包含工具
        - `rabin2`：
            - 获取 `ELF`，`PE`，`Mach-O`，`Java CLASS` 文件的区段、头信息、导入导出表、字符串相关、入口点等等
                - 包括打印出二进制文件的系统属性、语言、字节序、框架、以及使用了哪些加固技术
            - 支持多种格式的输出文件
            - 截图

```
root@kali:~/study# rabin2 -I megabeets_0x1
arch     x86
binsz    6220
bintype  elf
bits     32
canary   false
class    ELF32
crypto   false
endian   little
havecode true
intrp    /lib/ld-linux.so.2
lang     c
linenum  true
lsyms    true
machine  Intel 80386
maxopsz  16
minopsz  1
nx       false
os       linux
pcalign  0
pic      false
relocs   true
relro    partial
```

    - `radiff2`：比较文件不同的
    - `rahash2`：各种密码算法，hash算法集成
    - `rasm2`：汇编和反汇编
    - `ragg2`：开发shellcode工具(radare2自己编写的编译器)
    - `radare2`：整合了所有工具
- 资料
    - 官网
        - radare
            - https://rada.re/n/radare2.html
    - GitHub
        - radareorg/radare2: UNIX-like reverse engineering framework and command-line toolset
            - https://github.com/radareorg/radare2
    - 教程
        - The Official Radare2 Book
            - https://book.rada.re/index.html

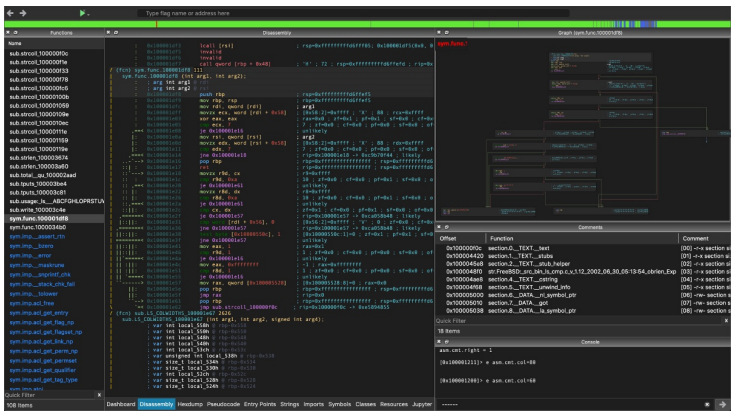
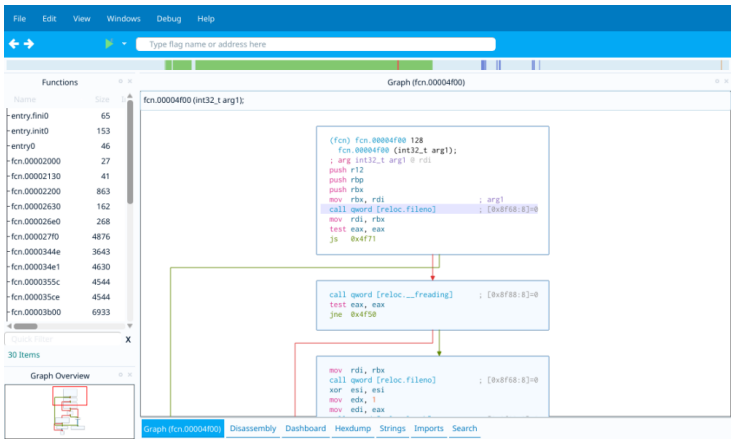
help帮助语法

```
$ radare2 -h
Usage: r2 [-ACdfLMnNqStuvwzX] [-P patch] [-p prj] [-a arch] [-b bits] [-i file
          [-s addr] [-B baddr] [-m maddr] [-c cmd] [-e k=v] file|pid|-|--|=
 --           run radare2 without opening any file
 -            same as 'r2 malloc://512'
 =            read file from stdin (use -i and -c to run cmds)
 -=           perform !=! command to run all commands remotely
 -0           print \x00 after init and every command
 -2           close stderr file descriptor (silent warning messages)
 -a [arch]    set asm.arch
 -A           run 'aaa' command to analyze all referenced code
 -b [bits]    set asm.bits
 -B [baddr]   set base address for PIE binaries
 -c 'cmd..'   execute radare command
 -C           file is host:port (alias for -c+=http://%s/cmd/)
 -d           debug the executable 'file' or running process 'pid'
 -D [backend] enable debug mode (e cfg.debug=true)
 -e k=v       evaluate config var
 -f           block size = file size
 -F [binplug] force to use that rbin plugin
 -h, -hh      show help message, -hh for long
 -H ([var])   display variable
 -i [file]    run script file
 -I [file]    run script file before the file is opened
 -k [OS/kern] set asm.os (linux, macos, w32, netbsd, ...)
 -l [lib]     load plugin file
 -L           list supported IO plugins
 -m [addr]    map file at given address (loadaddr)
 -M           do not demangle symbol names
 -n, -nn      do not load RBin info (-nn only load bin structures)
 -N           do not load user settings and scripts
 -q           quiet mode (no prompt) and quit after -i
 -Q           quiet mode (no prompt) and quit faster (quickLeak=true)
 -p [prj]     use project, list if no arg, load if no file
 -P [file]    apply rapatch file and quit
 -r [rarun2]  specify rarun2 profile to load (same as -e dbg.profile=X)
 -R [rr2rule] specify custom rarun2 directive
 -s [addr]    initial seek
 -S           start r2 in sandbox mode
 -t           load rabin2 info in thread
 -u           set bin.filter=false to get raw sym/sec/cls names
 -v, -V       show radare2 version (-V show lib versions)
 -w           open file in write mode
 -x           open without exec-flag (asm.emu will not work), See io.exec
 -X           same as -e bin.usextr=false (useful for dyldcache)
 -z, -zz      do not load strings or load them even in raw
```

R2Pipe

- `R2Pipe`
  - 是什么：`R2Pipe`是一个可以调用`radare2`的`Python`脚本库
  - 示例代码
    - https://github.com/radareorg/radare2-r2pipe/tree/master/python/examples

Cutter

- `Cutter`
  - 一句话描述：radare2的GUI版本
    - Free and Open Source RE Platform powered by radare2
    - Cutter is the official UI for radare2 for Linux, macOS and Windows, it's written in C++ and uses the Qt
  - 支持多平台
    - `Linux`
    - `Mac`
    - `Windows`
  - 实现细节
    - `C++` 语言写的
    - 前端：`QT`
  - 截图
    - 
    - 
  - 特点
    - 内置Ghidra decompiler
      - 无需额外安装Java
  - 核心功能和特点
    - 开源 Open Source
      - Completely FREE and licensed under GPLv3
    - Decompiler
      - Native integration of Ghidra's decompiler in Cutter releases
    - Graph View

        - Fully featured graph view as well as mini-graph for fast navigation
    - Debugger
        - Multiplatform native and remote debugger for dynamic analysis
    - Disassembly
        - Linear disassembly view
    - Hex Editor
        - View and modify any file with a rich and powerful Hex View
    - Python Scripting Engine
        - Quickly write python scripts to automate tasks
    - Plugins
        - Use Native or Python plugins to extend Cutter's core functionality
    - Binary Patching
        - Add, remove and modify bytes and instructions
    - Emulation
        - Great for automation, crypto algorithms and malware analysis
    - Theme Editor
        - Fully featured theme editor for easy and user-friendly customization of Cutter
    - Modern & Customizable UI
        - Built using Qt C++ and design best practices
    - Integrated Radare2 Console
    - Multi Language
    - Binary Searching
    - Types & Structs
    - Syntax Highlighting
    - STDIO Redirection
    - Remote Debugging
    - Kernel Debug
    - Graph Overview
- 资料
    - 官网
        - Cutter
            - https://rada.re/n/cutter.html
        - Cutter
            - https://cutter.re
    - GitHub
        - radareorg/cutter: Free and Open Source Reverse Engineering Platform powered by radare2
            - https://github.com/radareorg/cutter

iOS安全

iOS安全包含很多方面：

- 数据保护
  - 网络数据
  - IPA 资源数据
  - 沙盒数据
    - sqlite
      - sqlite加密
        - SQLCipher
          - https://github.com/sqlcipher/sqlcipher
  - Keychain 数据
- 混淆保护
  - 符号混淆
    - 工具
      - `ios-class-guard`
    - 破解
      - `class-dump`
  - 字符串混淆
  - 指令混淆
    - =? 代码逻辑混淆
- 反调试保护
  - 常见方式
    - `ptrace`
    - `sysctl`
- 异常检测
  - 越狱设备检测
  - 重签名检测
  - 动态库注入检测
  - 调试检测
  - 钩子检测
- 扫描工具
  - `fortify`
    - 侧重于代码的安全漏洞
  - `coverity`
    - 侧重于代码质量
- 逆向
  - 工具
    - `dumpdecrypted`
    - `Reveal`
    - `Cycript`
  - 高级内容
    - 程序加载
    - `Mach-O` 文件格式
    - ARM 汇编
    - hook=钩子

        - 常见方式
            - MethodSwizzle
                - 通过 runtime 交换方法的实现
            - fishhook
                - facebook 开源的一个库
                    - facebook/fishhook: A library that enables dynamically rebinding symbols in Mach-O binaries running on iOS.
                        - https://github.com/facebook/fishhook

- 破解工具
    - `MonkeyDev`
- 越狱工具
    - `adv-cmds`
        - 执行 ps 命令报错，需要安装这个工具
    - `appsync`
        - 让系统不再验证签名，以免安装应用失败
    - `iFile`
        - 在手机上查看文件目录
    - `Cydia Substrate`
        - 允许第三方开发者在越狱系统的方法中打一些补丁或扩展方法

正向工具

混淆

ios-class-guard

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

常用工具

调试器

- 苹果设备调试
  - 之前用：`gdb`
  - 后来改用：`lldb`

debugserver

- `debugserver`
  - 是什么：一个终端的应用
    - 也是：`XCode` 去调试iOS设备中程序时候的进程名
  - 在哪里：iOS设备中
    - 位置：`/Developer/usr/bin/debugserver`
    - 注：iOS中默认没安装
      - iOS中安装 `debugserver`
        - 在设备连接过一次 `Xcode` ，并在 `Window` -> `Devices` 中添加此设备后
          - `debugserver` 才会被 `Xcode` 安装到 `iOS` 的 `/Developer/usr/bin/` 下
  - 作用：作为服务端，接受来自远端的 `gdb` 或 `lldb` 的调试
    - 可以理解为：`lldb` 的 `server`
  - 为何需要
    - iOS中，由于App运行检测到越狱后会直接退出，所以需要通过 `debugserver` 来启动程序
  - 通过 `debugserver` 来启动程序
    - 举例
      - `debugserver -x backboard 0.0.0.0:1234 ./*`
      - `debugserver *:1234 -a "MoneyPlatListedVersion"`

插件开发

MonkeyDev

- MonkeyDev
  - 一句话描述：一个基于Xcode模块技术快速开发越狱和非越狱插件的工具，可以自动完成逆向中的固定步骤，一键集成非越狱插件，大大提升逆向分析和开发效率
  - 主要包含模块
    - Logos Tweak
      - 使用theos提供的logify.pl工具将*.xm文件转成*.mm文件进行编译，集成了CydiaSubstrate，可以使用MSHookMessageEx和MSHookFunction来Hook OC函数、C/C++函数或指定地址
    - CaptainHook Tweak
      - 使用CaptainHook提供的头文件进行OC函数的Hook，以及属性的获取
    - Command-line Tool
      - 可以直接创建运行于越狱设备的命令行工具
    - MonkeyApp
      - 自动给第三方应用集成Reveal、Cycript和注入dylib的模块，支持调试dylib和第三方应用，支持Pod给第三方应用集成SDK，只需要准备一个砸壳后的ipa或者app文件即可
    - MonkeyPod
      - 将自动开发的非越狱插件制造成Pod以供其它人通过pod的方法来使用
    - MonkeyAppMac
      - 针对Mac逆向开发的模块，可以自动集成substitute，注入以及符号还原工作
  - 资料
    - Github
      - AloneMonkey/MonkeyDev: CaptainHook Tweak、Logos Tweak and Command-line Tool、Patch iOS Apps, Without Jailbreak.
        - https://github.com/AloneMonkey/MonkeyDev
    - 官网
      - 文档 | MonkeyDev
        - https://monkeydev.org/docs/index.html

Mach-O处理

`Apple` 的 `Mac` 、 `iOS` 等平台的可执行文件，都是 `Mach-O` 格式的。

即苹果的可执行文件主要都是 `Mach-O` 格式的。

现有很多可以处理 `Mach-O` 格式的工具。

- `Mach-O`
  - = `Mach Object`
  - 文件类型
    - `Executable` =应用=可执行文件
    - `Dylib Library` =动态链接库= `DSO` 或 `DLL`
    - `Static Library` =静态链接库
    - `Bundle` ： 不能被链接的Dylib，只能在运行时使用dlopen( )加载，可当做macOS的插件
    - `Relocatable Object File` =可重定向文件
- 相关概念
  - `FatFile` / `FatBinary`
    - 一个由**不同的编译架构**后的 `Mach-O` 产物所合成的集合体
      - 一个架构的 `Mach-O` 只能在相同架构的机器或者模拟器上用
        - 为了支持不同架构需要一个集合体
- 常见工具
  - `class-dump`
  - `MachOView`
  - `jtool`
  - `otool`

class-dump

- class-dump
  - 一句话描述：用于处理 `Objective-C` 的 `Mach-O` 文件信息的命令行工具，可以导出类的定义、分组和协议。
    - command-line utility for examining the Objective-C segment of Mach-O files
  - 说明
    - 和 `otool -ov` 导出的信息是一样的
      - 但是显示为 `Objective-C` 定义，更易读
  - 原理
    - 利用了 `Objective-C` 语言的运行时的特性
      - 将存储在 `Mach-O` 文件中的头文件信息提取出来，并生成对应的 `.h` 文件
  - 用途
    - 查看闭源的 `应用` 、 `frameworks` 、 `bundles`
      - 查看其中的头文件信息
    - 对比一个 APP 不同版本之间的接口变化
      - 通过导出不同版本的库的头文件的对比看出来
    - 对一些私有 `frameworks` 做些有趣的试验
  - 资料
    - GitHub
      - nygard/class-dump: Generate Objective-C headers from Mach-O files.
        - https://github.com/nygard/class-dump
    - 官网
      - class-dump - Steve Nygard
        - http://stevenygard.com/projects/class-dump/

下载

- class-dump-3.5.dmg
- class-dump-3.5.tar.gz
- class-dump-3.5.tar.bz2

用法举例

- class-dump AppKit
  - `class-dump /System/Library/Frameworks/AppKit.framework`
- class-dump UIKit
  - `class-dump /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk/System/Library/Frameworks/UIKit.framework`
- class-dump UIKit and all the frameworks it uses

    - `class-dump /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk/System/Library/Frameworks/UIKit.framework -r --sdk-ios 4.3`
- class-dump UIKit (and all the frameworks it uses) from developer tools that have been installed in /Dev42 instead of /Developer
    - `class-dump /Dev42/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.0.sdk/System/Library/Frameworks/UIKit.framework -r --sdk-root /Dev42/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.0.sdk`

实际使用举例

之前从WebDriverAgent的源码中看到很多头文件的头部都有：`Generated by class-dump`

举例：

`refer/WebDriverAgent/PrivateHeaders/XCTest/XCTestDriver.h`

```
//
//     Generated by class-dump 3.5 (64 bit).
//
//     class-dump is Copyright (C) 1997-1998, 2000-2001, 2004-2013 by Steve Ny
//
```

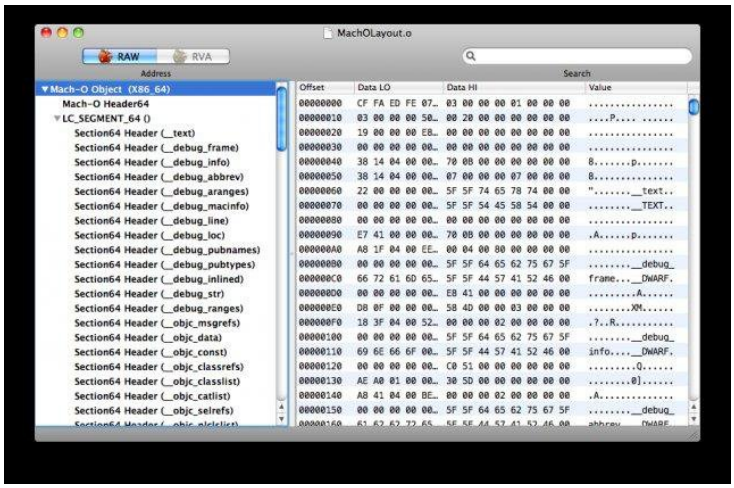
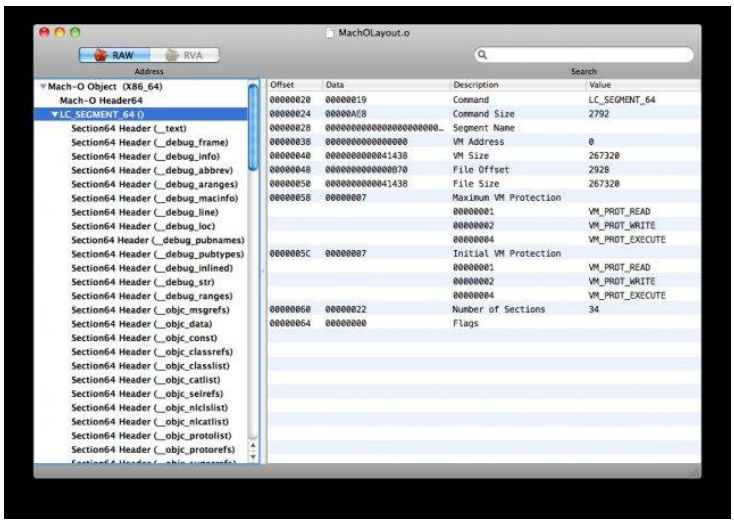
-》说明这些文件都是通过 `class-dump` 从库文件中导出生成的。

help帮助语法

```
class-dump 3.5 (64 bit)
Usage: class-dump [options] <mach-o-file>

  where options are:
    -a             show instance variable offsets
    -A             show implementation addresses
    --arch <arch>  choose a specific architecture from a universal binary (ppc
    -C <regex>     only display classes matching regular expression
    -f <str>       find string in method name
    -H             generate header files in current directory, or directory sp
    -I             sort classes, categories, and protocols by inheritance (ove
    -o <dir>       output directory used for -H
    -r             recursively expand frameworks and fixed VM shared libraries
    -s             sort classes and categories by name
    -S             sort methods by name
    -t             suppress header in output, for testing
    --list-arches  list the arches in the file, then exit
    --sdk-ios      specify iOS SDK version (will look in /Developer/Platforms/
    --sdk-mac      specify Mac OS X version (will look in /Developer/SDKs/MacO
    --sdk-root     specify the full SDK root path (or use --sdk-ios/--sdk-mac
```

MachOView

- `MachOView`
  - 是什么：查看和编辑Intel的 `x86` 和 `ARM` 的 `Mach-O` 二进制文件的工具
  - 截图
    - 
    - 
  - 资料
    - 最早好像是在sourceforge
      - MachOView download | SourceForge.net
        - https://sourceforge.net/projects/machoview/
    - 后来有人fork到GitHub
      - gdbinit/MachOView: MachOView fork
        - https://github.com/gdbinit/MachOView
    - 现在有国人fork后继续维护
      - fangshufeng/MachOView: 分析Macho必备工具
        - https://github.com/fangshufeng/MachOView

jtool

- `jtool`
  - 新版叫：`jtool2`
  - 类似于 `otool` 的，解析查看 `Mach-O` 文件格式信息
    - 区别：添加了许多Mach-O相关的命令
      - jtool比otool功能更完善
  - 支持多种运行平台
    - `OS X` = `Mac`
    - `iOS`
    - `Linux`
  - 功能
    - in-binary search functionality
    - symbol injection
    - built-in disassembler functionality with (limited but constantly improving) emulation capabilities, which already outdo fancy commercial GUI disassemblers.
    - Color terminal output, enabled by JCOLOR=1
  - 资料
    - 官网
      - JTool2 - Taking the O out of otool - squared
        - http://www.newosxbook.com/tools/jtool.html

otool

- `otool`
  - = `object file displaying tool`
  - 是什么：针对目标文件的展示工具
  - 做什么：用来发现应用中使用到了哪些系统库，调用了其中哪些方法，使用了库中哪些对象及属性
  - 来源：Xcode自带的常用工具
- 相关
  - 比otool更好的：`jtool`
  - `otool` 的 `GUI` 版：`otx`
    - x43x61x69/otx: The Mach-O disassembler. Now 64bit and Xcode 6 compatible.
      - https://github.com/x43x61x69/otx

查看当前otool位置：

```
✘ crifan@licrifandeMacBook-Pro □ ~ □ which otool
/usr/bin/otool
```

当前版本：

```
✘ crifan@licrifandeMacBook-Pro □ ~ □ otool --version
llvm-otool(1): Apple Inc. version cctools-927.0.2
Apple LLVM version 10.0.1 (clang-1001.0.46.4)
  Optimized build.
  Default target: x86_64-apple-darwin19.2.0
  Host CPU: broadwell

  Registered Targets:
    aarch64    - AArch64 (little endian)
    aarch64_be - AArch64 (big endian)
    arm        - ARM
    arm64      - ARM64 (little endian)
    armeb      - ARM (big endian)
    thumb      - Thumb
    thumbeb    - Thumb (big endian)
    x86        - 32-bit X86: Pentium-Pro and above
    x86-64     - 64-bit X86: EM64T and AMD64
```

help帮助语法

```
✘ crifan@licrifandeMacBook-Pro □ ~ □ otool -help
error: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xcto

Usage: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xcto
    -f print the fat headers
    -a print the archive header
    -h print the mach header
    -l print the load commands
    -L print shared libraries used
    -D print shared library id name
    -t print the text section (disassemble with -v)
    -p <routine name>  start dissassemble from routine name
    -s <segname> <sectname> print contents of section
    -d print the data section
    -o print the Objective-C segment
    -r print the relocation entries
    -S print the table of contents of a library (obsolete)
    -T print the table of contents of a dynamic shared library (obsolete)
    -M print the module table of a dynamic shared library (obsolete)
    -R print the reference table of a dynamic shared library (obsolete)
    -I print the indirect symbol table
    -H print the two-level hints table (obsolete)
    -G print the data in code table
    -v print verbosely (symbolically) when possible
    -V print disassembled operands symbolically
    -c print argument strings of a core file
    -X print no leading addresses or headers
    -m don't use archive(member) syntax
    -B force Thumb disassembly (ARM objects only)
    -q use llvm's disassembler (the default)
    -Q use otool(1)'s disassembler
    -mcpu=arg use `arg' as the cpu for disassembly
    -j print opcode bytes
    -P print the info plist section as strings
    -C print linker optimization hints
    --version print the version of /Applications/Xcode.app/Contents/Developer/
```

砸壳工具

iOS中的app，发布渠道一般都是 `App Store` 。

从 `App Store` 下载的APP全都是经过苹果加密过的 `ipa` 包。

而Apple会为了安全，给app加密(使用Apple ID相关的对称加密算法)，这个过程俗称为： `加壳` ，就像给app外部上加了一层壳。

而加密后的 `ipa` 包，是无法对其进行反编译的(需要对其进行解密才能反编译)，也无法 `class-dump`

- 相关说明
  - 自己编译的项目没有加密，能够解析出来
  - 但是 `class-dump` 不能直接将 `App Store` 上的app的头文件导出来
    - 你只会导出 `CDStructures.h` 这个头文件
      - 而这里边基本是没有信息的
        - 所以需要用Dumpdecrypted去破壳后，才可以

想要破解分析之前，需要把这层壳砸破。

如何砸壳呢？就要先了解app运行机制：app程序运行起来都会直接在内存解密出原始代码

可以在越狱的设备里面通过内存 `dump` 方式提取解密后的程序，这种解密过程，也就是给app去壳的过程，又称为 `砸壳` = `破壳`

- 额外说明
  - 解密之后还需要手动恢复 `Mach-O` 头信息才能运行
  - 由于高版本非完美越狱里面，都没有删掉签名验证
    - 所以直接运行都会出现 `killed 9`
      - 需要手动签名之后才能使用

有很多用于砸壳的工具，整理如下。

常见砸壳工具

- `dumpdecrypted`
  - 一般配合 `Cycript` 使用?
- `clutch`
- `frida-ios-dump`
- `bfinject`

bfinject

Clutch

- `Clutch`
  - 是什么
    - Fast iOS executable dumper
    - a high-speed iOS decryption tool
  - 功能：脱壳=砸壳
    - 针对（越狱的）iOS设备，（解密）导出头文件
  - 支持平台
    - 所有iOS设备：iPhone/iPod Touch/iPad
  - 资料
    - GitHub
      - KJCracks/Clutch: Fast iOS executable dumper
        - https://github.com/KJCracks/Clutch
      - Wiki
        - Home · KJCracks/Clutch Wiki
          - https://github.com/KJCracks/Clutch/wiki
        - Tutorial · KJCracks/Clutch Wiki
          - https://github.com/KJCracks/Clutch/wiki/Tutorial
        - FAQ · KJCracks/Clutch Wiki
          - https://github.com/KJCracks/Clutch/wiki/FAQ

help语法

```
Clutch [OPTIONS]
-b --binary-dump     Only dump binary files from specified bundleID
-d --dump            Dump specified bundleID into .ipa file
-i --print-installed Print installed application
--clean              Clean /var/tmp/clutch directory
--version            Display version and exit
-? --help            Display this help and exit
```

Dumpdecrypted

- Dumpdecrypted
  - 一句话描述：iOS的砸壳工具
    - Dumps decrypted iPhone Applications to a file
  - 资料
    - GitHub
      - stefanesser/dumpdecrypted: Dumps decrypted mach-o files from encrypted iPhone applications from memory to disk. This tool is necessary for security researchers to be able to look under the hood of encryption.
        - https://github.com/stefanesser/dumpdecrypted

frida-ios-dump

- 一句话描述：Pull a decrypted IPA from a jailbroken device
- Github
  - AloneMonkey/frida-ios-dump: pull decrypted ipa from jailbreak device
    - https://github.com/AloneMonkey/frida-ios-dump

越狱工具

- `iOS 11`
  - 主要越狱工具
    - `Electra`
    - `unc0ver`

Cydia Substrate

frida

Electra

unc0ver

附录

下面列出相关参考资料。

参考资料

- iOS安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com
- iOS 的系统安全性比Android 系统要高！// 解读iOS安全机制 - 知乎 (zhihu.com)
- iOS安全杂谈 (inforsec.org)
- iOS应用安全开发概述 · 唐巧的博客 (devtang.com)
- 安卓和iOS谁更安全 2021年比较报告 (ganbey.com)
- iOS开发安全-InfoQ
- 不想iPhone被黑？赶紧试试这个-iPhone,被黑,Apple ID,验证, ——快科技(驱动之家旗下媒体)--科技改变未来 (mydrivers.com)
- iPhone可以被黑客入侵吗？ 如果是，您必须采取什么行动？ — iStarTips