

目录

前言	1.1
安全概览	1.2
安全背景知识	1.3
编程语言	1.3.1
汇编语言	1.3.1.1
X86	1.3.1.1.1
ARM	1.3.1.1.2
体系架构	1.3.2
寄存器	1.3.2.1
X86	1.3.2.1.1
ARM	1.3.2.1.2
堆栈	1.3.2.2
可执行文件	1.3.3
安全通用知识	1.4
安全机制	1.4.1
Windows	1.4.1.1
ASLR	1.4.1.1.1
CFG	1.4.1.1.2
DEP	1.4.1.1.3
GS	1.4.1.1.4
缓冲区溢出	1.4.1.1.4.1
SafeSEH	1.4.1.1.5
安全学习方法	1.4.2
CTF比赛	1.4.3
正向安全	1.4.4
漏洞编号	1.4.5
CVE	1.4.5.1
CNVD	1.4.5.2
Web端安全	1.5
渗透测试	1.5.1
常用工具	1.5.2
安全操作系统	1.5.2.1
Kali Linux	1.5.2.1.1
组织和标准	1.5.3
OWASP	1.5.3.1

设备端安全	1.6
计算机安全	1.6.1
Windows	1.6.1.1
反汇编器	1.6.1.1.1
Capstone	1.6.1.1.1.1
反编译器	1.6.1.1.2
dnSpy	1.6.1.1.2.1
ILSpy	1.6.1.1.2.2
静态安全检工具	1.6.1.1.3
winchecksec	1.6.1.1.3.1
可执行文件工具	1.6.1.1.4
ExeInfo PE	1.6.1.1.4.1
查壳工具	1.6.1.1.5
Detect It Easy	1.6.1.1.5.1
PEiD	1.6.1.1.5.2
逆向工具	1.6.1.1.6
IDA	1.6.1.1.6.1
Ollydbg	1.6.1.1.6.2
WinDbg	1.6.1.1.6.3
内存修改工具	1.6.1.1.7
CE	1.6.1.1.7.1
MHS	1.6.1.1.7.2
ModifyMemory	1.6.1.1.7.3
Mac	1.6.1.2
逆向工具	1.6.1.2.1
Hopper Disassembler	1.6.1.2.1.1
Linux	1.6.1.3
编译套件	1.6.1.3.1
LLVM	1.6.1.3.1.1
LLDB	1.6.1.3.1.1.1
Obfuscator-LLVM	1.6.1.3.1.1.2
调试器	1.6.1.3.2
lldb-server	1.6.1.3.2.1
逆向工具	1.6.1.3.3
Miasm	1.6.1.3.3.1
radare2	1.6.1.3.3.2
Cutter	1.6.1.3.3.2.1
移动安全	1.6.2

Android安全	1.6.2.1
iOS安全	1.6.2.2
正向工具	1.6.2.2.1
混淆	1.6.2.2.1.1
ios-class-guard	1.6.2.2.1.1.1
破解工具	1.6.2.2.2
调试器	1.6.2.2.2.1
debugserver	1.6.2.2.2.1.1
插件开发	1.6.2.2.2.2
MonkeyDev	1.6.2.2.2.2.1
Mach-O处理	1.6.2.2.2.3
class-dump	1.6.2.2.2.3.1
MachOView	1.6.2.2.2.3.2
jtool	1.6.2.2.2.3.3
otool	1.6.2.2.2.3.4
砸壳	1.6.2.2.2.4
bfinject	1.6.2.2.2.4.1
Clutch	1.6.2.2.2.4.2
Dumpdecrypted	1.6.2.2.2.4.3
frida-ios-dump	1.6.2.2.2.4.4
越狱工具	1.6.2.2.3
Cydia Substrate	1.6.2.2.3.1
frida	1.6.2.2.3.2
Electra	1.6.2.2.3.3
unc0ver	1.6.2.2.3.4
工控物联网安全	1.6.3
特定设备安全	1.6.4
WiFi安全	1.6.4.1
Aircrack-ng	1.6.4.1.1
Wifiphisher	1.6.4.1.2
信息存储安全	1.7
硬件	1.7.1
TrustZone	1.7.1.1
软件	1.7.2
TEE	1.7.2.1
其他安全相关	1.8
代码审计工具	1.8.1
Checkmarx CxEnterprise	1.8.1.1

Armorize	CodeSecure	1.8.1.2
Fortify		1.8.1.3
RIPS		1.8.1.4
相关工具		1.8.2
抓包工具		1.8.2.1
Wireshark		1.8.2.1.1
破解密码		1.8.2.2
John the Ripper		1.8.2.2.1
Hashcat		1.8.2.2.2
Hydra		1.8.2.2.3
代理工具		1.8.2.3
远程操作工具		1.8.2.4
附录		1.9
资料和文档		1.9.1
参考资料		1.9.2

信息安全概览

- 最新版本: v1.7
- 更新时间: 20210524

简介

边学习信息安全技术，边总结技术教程。已整理出宏观的各个方面的安全的分类和概念。以及基本的计算机安全、移动端安全、物联网安全等细节内容。

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/information_security_overview: 信息安全概览](#)

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

- [信息安全概览 book.crifan.com](#)
- [信息安全概览 crifan.github.io](#)

离线下载阅读

- [信息安全概览 PDF](#)
- [信息安全概览 ePUB](#)
- [信息安全概览 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](#)

crifan.com, 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： 2021-05-24 21:58:32

安全概览

背景

先说说写这个教程的背景：

- 之前已写过 安卓安全和破解 的教程
 - https://github.com/crifan/android_app_security_crack
 - 目前点赞不少 600+ 个star
 - 看来大家比较关注这个领域
- 自己计划从事 计算机安全领域
 - 之前是小白，没这方面的经验
 - 打算边自学，边总结，总结到这个教程中
 - 供自己和他人参考

信息安全技术概览

信息安全技术概念包含内容较多，且涉及维度较广，下面以不同维度来阐述，常见分类和对应内容。

- 信息安全
 - 概述

Web应用程序	桌面应用
应用技术	
Apache AngularJS JavaScript ASP.NET Django OAuth HTML WebSockets WordPress Python Java Ruby Spring Framework PHP Nginx JSON Laravel RESTful APIs XML React WebRTC MVVM MVC HTTP Ruby on Rails Session Cookies Local Storage Cookie Flags	System Services PE Anatomy Protocol Analysis Privilege Rings Windows Internals COM WMI Impersonation Levels .NET Kernel Space C UAC MOF Calling Modes WCF User Space C# Device Drivers x86/x64 Assembly Process Tokens C++ Named Pipes Windows Registry
进攻性和防守性概念	
Certificate Pinning Code Execution Principle of Least Privilege Clickjacking XSS XXE LFI Open Redirect Authentication Bypass Authorization Issues SQLi RFI CSRF Password Policy Account Lockout Cookie Flags Fail Secure SSL/TLS Issues Defense in Depth Separation of Duties	Defense in Depth Code Execution Write Primitive Stack Pivot JIT Spray Mitigation Circumvention Stack Overflow PatchGuard Principle of Least Privilege Read Primitive SafeSEH Memory Corruption Stack Cookies Use-after-Free SMEP DEP ASLR Bootkits SMAP Write-What-Where Primitive Rootkits Heap Spray Information Leak CFG DSE ROP Gadgets Separation of Duties Fail Secure Evasion and Stealth Heap Overflow Exploit Mitigations

- 根据不同 端 = 目标 = 设备 分
 - Web端 : 网络安全 = Web安全 = 互联网安全
 - 设备端
 - PC端 : 计算机安全
 - 包含
 - Windows
 - Mac
 - Linux
 - 移动端 : 移动安全
 - 包含
 - Android
 - 详见: [安卓安全和破解](#)
 - iOS
 - IoT端 : 物联网安全 ~= 工控安全
 - 详见: [工控安全概览](#)
 - 其他特定设备
 - WiFi安全
 - 根据不同领域和侧重点
 - Web网络
 - 渗透测试

- 详见：潜入你的网络：渗透测试
- 安全分析
 - 安全日志分析
- 本机
 - 二进制安全 ~= PWN
 - Window
 - Window漏洞挖掘 ~= Windows漏洞分析
 - 移动端
 - Anroid安全
 - iOS安全
- 广义的信息安全
 - 子领域=特殊领域
 - 信息存储安全
 - 典型应用场景：指纹、虹膜、信用卡PIN码等
 - 包含
 - 硬件
 - TrustZone
 - 软件
 - OP-TEE

crifan.com, 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：2021-05-24 21:57:06

安全背景知识

crifan.com, 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: 2020-08-02 20:34:06

编程语言

- 编程语言种类

- 机器语言：用二进制编码表示每条指令，是计算机能直接识别和执行的语言。

- 原理：底层都是0和1的二进制数据
 - 注：理论上来说，程序员也可以写机器语言
 - 只不过太难写，以及没必要直接写，所以实际上没人直接写机器语言

- 汇编语言

- 实质和机器语言是相同的，都是直接对硬件操作，只不过指令采用英文缩写的标识符，更容易识别和记忆

- 组成

- 指令，伪指令，宏指令

- 逻辑

- 你（程序员）写汇编的字母（和要操作的数字等），汇编程序帮你把汇编的字母（和数字）翻译成0和1的二进制

- 这样机器才能看懂，才能运行对应程序

- 高级语言

- 编写的程序不能直接被计算机识别，必须经过转换（目标代码即机器码）才能被执行

- 按照转换方式分类

- 解释类：边翻译边执行

- 编译类：先翻译再执行

- 常见语言

- C

- C语言的特点

- C语言允许直接访问物理地址，可以直接对硬件进行操作

- C语言程序代码质量高，程序执行效率高

- C语言使用范围大，可移植性好

- C++

- 由于：C++编译器优化程度太高

- 结果：世界上没有C++反编译器

- 也没有完美的C语言反编译器

- 很难做到完美

汇编语言

- 汇编
 - 汇编 = 汇编语言 = 汇编指令
 - 是什么
 - 汇编大多是指汇编语言，汇编程序
 - 把汇编语言翻译成机器语言的过程称为汇编
 - 汇编语言
 - 在汇编语言中，用符号代替机器语言的二进制码，就把机器语言变成了汇编语言
 - 是用助记符，符号和数字等来表示指令的程序设计语言，它与机器语言指令是一一对应的
 - 汇编程序
 - 用汇编语言编写的程序，机器不能直接识别。要由一种程序将汇编语言翻译成机器语言，这种起翻译作用的程序叫汇编程序。
 - 汇编程序是系统软件中语言处理的系统软件
 - 特点
 - 由于汇编更接近机器语言，能够直接对硬件进行操作，生成的程序与其他的语言相比具有更高的运行速度，占用更小的内存
 - 应用
 - 因此在一些对于时效性要求很高的程序，许多大型程序的核心模块以及工业控制方面大量应用
 - 种类
 - 有多少种不同内核的CPU，就有多少种汇编语言
 - 总结
 - 不同内核的CPU，必须有对应的汇编语言编译器将汇编语言编写的程序编译成对应CPU的机器语言代码，CPU才能正确识别和执行这些代码
 - 不同架构的CPU的汇编指令集并不相同
 - 不同的汇编程序有不同的汇编语言规定

通用知识

和逆向和破解相关的汇编语言的通用知识：

- 逆向中关键的指令：
 - `ldr` , `mov` , 读取指令，从地址读取数据到寄存器。
 - `str` , 保存指令，保存数据到寄存器。
 - `b` , 跳转指令，跳转到某个地址。
 - `cmp` , 比较指令，说明这里有分支。

X86汇编语言

- 常见X86汇编语言类型
 - ASM
 - MASM
 - TASM
 - OPTASM
 - 等

Intel 8086的汇编语言指令

- Intel 8086的指令
 - 概述
 - 117条基本指令
 - 6个功能组
 - 数据传送类指令
 - MOV/XCHG, PUSH/POP, LEA
 - 算数运算类指令
 - ADD/ADC/INC, SUB/SBB/DEC/CMP/NEG, MUL/IMUL, DIV/IDIV
 - 位操作类指令
 - AND/OR/XOR/NOT/TEST
 - 串操作类指令
 - 控制转移类指令
 - JMP/JCC/LOOP, CALL/RET, INT n
 - 处理机控制类指令
 - NOP
 - 其他
 - 伪指令

数据传送类指令

- 数据传送类指令
 - 概述：计算机中最基本，最重要的一种操作，传送指令也是最常用的一类指令
 - 作用：传送指令把数据从一个位置传送到另一个位置。
 - 特点：除标志寄存器传送指令外，均不影响标志位
 - 包含：`MOV XCHG PUSH POP LEA`
 - 传送指令`MOV`
 - 把一个字节或操作数从源地址传送至目的地址
 - 交换指令`XCHG`
 - 把两个地方的数据进行互换
 - 寄存器与寄存器之间对换数据
 - 寄存器与存储器之间对换数据
 - 不能在存储器与存储器之间对换数据
 - 进栈指令`PUSH`

- PUSH
 - Push r16/m16/seg 操作过程: 1. SP<-SP-2 2.SS:[SP]<-r16/m16
- POP
 - 出栈指令, 操作与PUSH相反
- 算数运算类指令
 - 概述: 四则运算计算机经常进行的一种操作
 - 作用: 实现二进制(十进制)数据的四则运算
 - 特点: 算术运算类指令往往对标志有影响
 - 建议
 - 掌握 ADD/ADC/INC , SUB/SBB/DEC/NEG/CMP
 - 熟悉 MUL/IMUL , DIV/IDIV
 - 理解 CBW/CWD , DAA/DAS , AAA/AAS/AAM/AAD
- 包含
 - 加法指令ADD
 - 功能: ADD指令将源与目的操作数相加, 结果送到目的操作数
 - ADD指令按状态标志的定义相应设置状态标志
 - 语法:
 - ADD reg, imm/reg/mem reg<-reg+imm/reg/mem
 - 带进位加法指令ADC
 - 功能: ADC指令将源与目的的操作数相加, 在加上进位CF标志, 结果送到目的操作数
 - ADC指令按状态标志的定义相应设置状态标志
 - 用途: ADC指令主要与ADD配合, 实现多精度加法运算
 - 语法:
 - ADD reg , imm/reg/mem
 - ADC mem , imm/reg
 - 增量指令INC
 - 功能: 对操作数加1(增量)
 - 不影响进位CF标志, 按定义设置其他状态标志
 - 语法:
 - INC reg/mem reg/mem<-reg/mem+1
 - 减法指令SUB
 - 功能: 将目的操作数减去源操作数, 结果送到目的操作数
 - 按照定义相应设置状态标志
 - 语法:
 - SUB reg , imm/reg/mem reg<-reg-imm/reg/mem
 - 带借位减法指令SBB
 - 功能: SBB指令将目的操作数减去源操作数, 在减去借位CF(进位), 结果送到目的操作数
 - 用途: SBB指令主要与SUB配合, 实现多精度减法运算

- 语法: SBB reg , imm/reg/mem reg<-reg-imm/reg/mem-CF
- 减量指令DEC
 - 功能: DEC指令对操作数减1 (减量)
 - 语法: DEC reg/mem reg/mem<-reg/mem-1
 - 说明: INC指令和DEC指令都是单操作数指令, 主要用于对计数器和地址指针的调整
- 求补指令NEG
 - 功能: NEG指令对操作数执行求补运算: 用0减去操作数, 然后结果返回操作数, 求补运算也可以表达成, 将操作数按位取反后加1
 - NEG指令对标志的影响与用0作减法的SUB指令一样
 - 语法: NEG reg/mem reg/mem<-0-reg/mem
- 比较指令CMP
 - 功能: 将目的操作数减去源操作数
 - 按照定义相应设置状态标志
 - 说明: 执行的功能与SUB指令类似, 但结果不回送目的操作数
 - 语法: CMP reg , imm/reg/mem reg—imm/reg/mem

位操作类指令

- 位操作类指令
 - 概述: 以二进制为基本单位进行数据的操作。一类常用的指令
 - 包含
 - 逻辑运算指令
 - ADD(与)
 - 功能: 对两个操作数执行逻辑与运算, 结果送到目的操作数。
 - 语法: ADD des , src des<-des^src
 - OR(或)
 - 功能: 对两个操作数执行逻辑或运算, 结果送到目的操作数
 - 语法: OR dest , src dest<-destv src
 - XOR(异或)
 - 功能: 对两个操作数执行逻辑异或运算, 结果送到目的操作数
 - 语法: XOR dest , src
 - NOT(非)
 - 功能: 对一个操作数执行逻辑非运算
 - 按位取反, 原来的0的位变1, 原来的1的位变0
 - 语法: NOT reg/mem
 - TEST(测试)
 - 移位指令
 - SHL(逻辑左移)
 - SHR(逻辑右移)
 - SAL(算术左移)
 - SAR(算术右移)

- 循环移位指令
 - ROL(左循环移位)
 - ROR(右循环移位)
 - RCL(带进位左循环移位)
 - RCR(带进位右循环移位)

串操作类指令

- 串操作类指令

控制转移类指令

- 控制转移类指令
 - 概述：用于实现分支，循环，过程等程序结构，是仅次于传送指令的常用指令
 - 建议：
 - 重点掌握：JMP/JCC/LOOP、CALL/RET、INT n/IRET 常用系统功能调用
 - 一般了解：LOOPZ/LOOPNZ INTO
 - 包含
 - 无条件转移指令JMP
 - 语法：JMP label
 - 作用：
 - 程序转向label标号指定的地址（标号要在程序其他位置标出）
 - 说明：
 - 只要执行无条件转移指令JMP，不需要任何条件，就使程序转到指定的目的地址处，从目标地址
 - 操作数是要转移到的目标地址开始执行指令
 - 原理
 - 程序的执行地址，是由段寄存器CS和指令指针IP共同确定的，即当前指令的地址为CS：IP
 - 程序的跳转是通过修改CS和IP的值来实现的
 - 条件转移指令JCC
 - 语法：Jcc label
 - 条件满足，发生转移：IP<-IP+8位位移量
 - 条件不满足，顺序执行
 - 说明：
 - 指定的条件cc如果成立，程序转移到由标号label指定的目标地址去执行指令，条件不成立，则程序将顺序执行下一条指令
 - 操作数label是短转移指令，要跳转的地址必须距当前IP地址-128~+127个单元的范围之内
 - Jcc指令不影响标志
 - 但要利用标志
 - 根据利用的标志位不同，16条指令分为3种情况
 - 判断单个标志位状态
 - 比较无符号数高低
 - 比较有符号数大小

- 循环指令LOOP
 - 语法: LOOP label
 - 功能: 循环指令是一种特殊的转移指令, 当满足某条件时, 反复执行一系列操作, 知道不满足为止
 - 说明: 循环指令利用CX寄存器作为计数器
- 子程序指令
 - 语法:
 - 4种类型
 - CALL label; 段内调用, 相对寻址。
 - CALL r16/m16; 段内调用, 间接寻址
 - CALL far ptr label; 段间调用, 直接寻址
 - CALL far ptr mem; 段间调用, 间接寻址
 - 原理:
 - 子程序是完成特定功能的一段程序
 - 当主程序(调用程序)需要执行这个功能时, 采用CALL调用指令转移到子程序的起始处执行
 - 当运行完子程序功能后, 采用RET返回指令回到主程序继续执行
 - 说明:
 - 子程序通常是与主程序分开完成特定功能的一段程序, 程序中有时要反复的实现相同的功能只不过参数不同而已, 把仅参数不同功能重复的程序编写成为子程序, 执行这个功能时, 就可以调用该子程序, 执行完成后在返回主程序。
- 中断指令
 - 语法=中断指令: INT
 - 举例:
 - INT i8
 - 特殊:
 - IRET: 中断返回指令, 实现中断返回
 - INTO: 溢出中断指令
 - 作用: 中断是一种改变程序执行顺序的方法, 在程序运行时, 遇到某些需要紧急处理的情况, 如停电, 数据的实时接收, 溢出等, 处理器暂停主程序的执行, 转去执行中断处理程序。
 - 分类:
 - 内部中断
 - 外部中断

处理器控制类指令

- 处理器控制类指令
 - 概述: 对CPU状态进行控制的指令
 - 包含
 - 空操作指令NOP
 - 语法:
 - NOP CS: SS: DS: ES
 - 作用: 不执行任何操作, 但占用一个字节存储单元, 空耗一个指令执行周期。
 - 用途:
 - NOP常用于程序调试

- 在需要预留指令空间时用NOP填充
- 代码空间多余时也可以用NOP填充
- 还可以用NOP实现软件延时
- 其他
 - LOCK HLT ESC WAIT
- 说明
 - 事实上, NOP和XCHG, AX, 的指令代码一样都是90H
 - 段超越前缀指令: 在允许段超越的存储器操作数之前, 使用段超越前缀指令, 将采用指定的段寄存器寻址操作数
 - CS: 使用代码段的数据
 - SS: 使用堆栈段的数据
 - DS: 使用数据段的数据
 - ES: 使用附加段的数据

伪指令

- 伪指令
 - 概述:
 - 没有对应的机器码的指令, 最终不被CPU所执行
 - 伪指令是由编译器来执行的指令
 - 编译器根据伪指令来进行相关的编译工作
 - 语法:
 - segment和ends是一对成对使用的伪指令
 - 这是在写可被编译器编译的汇编程序时, 必须要用到的一对伪指令
 - segment和ends的功能是定义一个段
 - segment说明一个段开始
 - 语法: 段名 segment
 - ends说明一个段结束
 - 语法: 段名 ends
 - 说明:
 - 一个汇编程序是由多个段组成的, 这些段被用来存放代码, 数据或当作栈空间来使用
 - 一个有意义的汇编程序中至少要有一个段, 这个段用来存放代码
 - 注意:
 - 不要搞混end和ends
 - end是汇编语言的结束
 - 一个汇编程序的结束标记, 编译器在编译汇编程序的过程中, 如果碰到了伪指令end, 就结束对源程序的编译
 - ends是伪指令的结束

ARM汇编语言

crifan.com, 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: 2020-08-03 23:06:19

体系架构

- PC端
 - X86
- 移动端
 - ARM

crifan.com, 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: 2020-08-03 22:52:19

寄存器

- 寄存器
 - 是什么：存储信息的单元或者说是器件
 - 注：这里讨论的寄存器都是CPU中的寄存器，位于CPU内部，而内存位于CPU外部
 - 使用
 - 对于一个汇编程序员来说，CPU中主要可以使用的也就是寄存器而已
 - 对比
 - 电脑 的内存
 - CPU的寄存器
 - 编程序员可以使用指令来读写CPU中的寄存器，从而实现对于CPU的控制
 - 特点
 - 不同的CPU，寄存器的个数和结构都是不一样的

crifan.com, 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： 2020-08-03 22:51:39

X86寄存器

- 8086 CPU中寄存器总共为14个：且均为16位（32位和64位均以16位为基础）
 - 14个寄存器：AX BX CX DX SP BP SI DI IP FLAG CS DS SS ES
 - 根据类型分
 - 通用寄存器
 - 数据寄存器：AX, BX, CX, DX
 - AX：累加寄存器，也称为累加器
 - BX：基地址寄存器
 - CX：计数器寄存器
 - DX：数据寄存器
 - 指针寄存器：SP和BP
 - SP：堆栈指针寄存器
 - BP：基指针寄存器
 - 变址寄存器：SI和DI
 - SI：源变址寄存器
 - DI：目的变址寄存器
 - 控制寄存器
 - IP：指令指针寄存器
 - FLAG：标志寄存器
 - 段寄存器
 - CS：代码段寄存器
 - DS：数据段寄存器
 - SS：堆栈段寄存器
 - ES：附加段寄存器

x86-64 的调用约定

x86-64 有16个64位寄存器，分别是：

rax, rbx, rcx, rdx, esi, edi, rbp, rsp, r8, r9, r10, r11, r12, r13, r14, r15

寄存器	描述
rax	作为函数返回值使用
rsp	栈指针寄存器，指向栈顶
rdi, rsi, rdx, rcx, r8, r9	依次用作函数参数；如果断点在 OC 方法的第一行，那 rdi 就是 self, rsi 就是 cmd
rbx, rbp, r10, r11, r12, r13, r14, r15	通用寄存器

ARM 寄存器

- ARM寄存器和架构
 - 32位 arm
 - 64位 arm

32位arm的调用约定

寄存器	描述
r0-r3	传递参数与返回值。如果断点在 OC 方法的第一行，那 r0 就是 self, r1 就是 cmd。如果超过四个参数，或者一些例如结构体的参数超过了32位 bit，那么参数将会通过栈来传递；返回值一般都在 r0 上
r4-r6, r8, r10- r11	没有特殊规定，通用寄存器
r7	栈帧指针寄存器(Frame Pointer)，指向下一个保存的栈帧(stack frame)和链接寄存器(link register, lr)在栈上的地址
r9	操作系统保留
r12	IP 寄存器(intra-procedure scratch)
r13	SP 寄存器(stack pointer)，是栈顶指针
r14	LR 寄存器(link register)，存放函数返回后需要继续执行的指令地址
r15	PC 寄存器(program counter)，指向当前指令地址
CPSR	当前程序状态寄存器(Current Program State Register)，在用户状态下存放像 condition 标志中断禁用等标志

arm64的调用约定

arm64有 r0 - r30 是31个通用整形寄存器，PC 不能再作为寄存器直接访问。每个寄存器可以存取一个64位大小的数。当使用 x0 - x30 访问时，它就是一个64位的数。当使用 w0 - w30 访问时，访问的是这些寄存器的低32位。

寄存器	描述
x0–x7	传递参数与返回值。如果参数个数超过了8个，多余的参数会存在栈上；返回值一般都在 x0 上
x29	栈帧指针寄存器(Frame Pointer)，指向一个保存的栈帧(stack frame)和链接寄存器(link register, lr)在栈上的地址
x31	SP 寄存器(stack pointer)，是栈顶指针；根据不同指令，也有可能是 zero register
x30	LR 寄存器(link register)，存放函数的返回地址
CPSR	当前程序状态寄存器(Current Program State Register)，在用户状态下存放像 condition 标志中断禁用等标志

crifan.com, 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：2020-08-03 23:06:25

堆栈

- 堆栈
 - 是什么：堆栈都是一种数据项按序排序的数据结构
 - 特点：只能在一端（称为栈顶）对数据项进行插入和删除
 - 堆：队列优先，先进先出
 - 栈：先进后出
 - 功能：暂时存放数据和地址
 - 用途：通常用来保护断电和现场
 - 操作：堆栈中定义了一些操作
 - 两个最重要的是PUSH和POP
 - PUSH操作：在堆栈的顶部加入一个元素
 - POP操作：相反，在堆栈顶部移去一个元素，并将堆栈的大小减一

crifan.com, 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：2020-08-03 22:52:40

可执行文件

常见可执行文件格式

- 不同系统
 - Windows
 - PE
 - = Portable Executable
 - COFF
 - = Common Object File Format
 - Linux 类系统: Linux 、 Unix 、 Mac 等
 - ELF
 - = Executable and Linkable Format
 - Mac
 - Mach-O
- 其他相关
 - 二进制往往还根据系统不同而略有不同
 - 举例
 - Windows系统
 - 32位 = x86
 - 64位 = x64

crifan.com, 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: 2020-08-06 22:31:47

安全通用知识

此处整理各方面的安全的基础和通用的知识。

破解 vs 开发

- 破解：属于 逆向
- 开发：属于 正向

常见术语和名词

- 常见术语和名词
 - 后渗透
 - 红方 蓝方
 - 威胁建模分析
 - PIA分析
 - malware
 - 恶意程序=恶意软件
 - 逆向工程师

常见问题

问：做安全的破解的，是否一定要会开发？

- 答：不一定。但最好会。
 - 做安全破解的，会开发，属于加分项。
 - 原因也很简单
 - 就像：做逆向破解的就像小偷去你别家偷东西
 - 肯定没有，作为正向开发，作为开发商建造房子的你，对房子内部构造更熟悉，更容易找到突破口，找到可能的漏洞，并充分利用漏洞去实现自己的攻击。

crifan.com, 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：2020-08-02 20:43:11

安全机制

一些关于安全机制的心得：

- 在用 vs 较高版本编译程序测试溢出等的时候，注意要关闭保护功能(如常见的 DEP 以及 ASLR)
 - 在项目菜单的属性里面设置

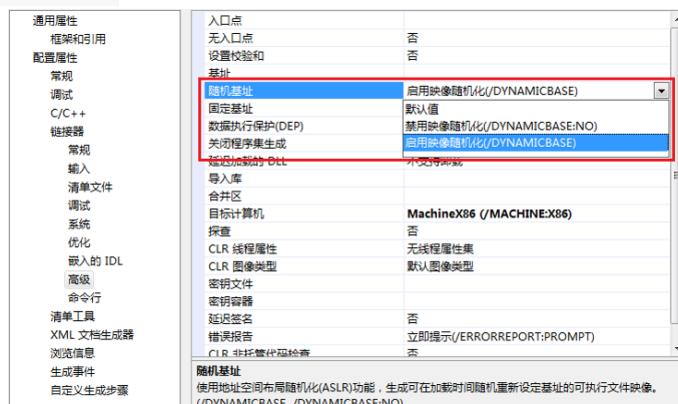
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2020-08-08 12:19:46

Windows

crifan.com, 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: 2020-08-08 12:44:55

ASLR

- ASLR
 - = Address Space Layout Randomization = 地址空间布局随机化
 - 是什么：一个编译器参数
 - 是一种针对缓冲区溢出的安全保护技术
 - 背景
 - 没有ASLR时
 - 每次进程执行，加载到内存中，代码所处堆栈stack的位置都是相同的
 - 容易被识别出所在位置
 - 容易被破解
 - 如果开启了 ASLR =机制：
 - 操作系统加载器会针对基址再去加上一个随机生成的偏移地址，然后再去加载程序模块
 - 通过对堆、栈、共享库映射等线性区布局的随机化
 - 增加攻击者预测目的地址的难度
 - 防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的
 - 提示
 - (虚拟地址) 此技术需要操作系统和软件相配合
 - 目的
 - 对付破解的一种有效的手段
 - 让程序被破解更加难
 - 破解程序一般指的是，运行 shellcode
 - 系统支持ASLR的情况
 - Linux
 - FreeBSD
 - Windows
 - PE 头文件中会设置 IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE 标示来说明其支持 ASLR
 - 如何开启
 - 语法：
 - 开启： /DYNAMICBASE
 - 关闭： /DYNAMICBASE:NO
 - Visual Studio 项目属性的配置



- 使用此技术后，杀死某程序后重新开启
 - Linux：地址会改变
 - Windows：地址不会改变，重启系统才会改变
- ASLR主要影响几种部分
 - 模块随机化
 - 堆栈随机化
 - PEB/TEB随机化
- 相关
 - PIE VS ASLR
 - Linux
 - PIE = Position-Independent Execute = 地址无关可执行文件
 - 编译时
 - 将程序编译为位置无关
 - 地址随机化针对：代码段和数据段(.data 段 .bss 段)
 - ASLR：
 - 地址随机化针对：其他内存地址
 - Linux 的 ASLR + PIE 作用 == Window 下 ASLR 的作用

如何绕过ASLR

- 攻击未启用ASLR的模块
 - 虽然有映像随机化，但有可能进程中存在未启用ASLR的模块。前面提到的 ROP 技术要求从一个固定的地址获取 Gadget，如果进程中存在未启用 ASLR 的模块，那么就可以从那个模块获取 Gadget 了。使用 OD 的 OllyFindAddr 插件可以快速找到进程空间中未启用ASLR的模块
- 堆喷射（HeapSpray）技术
 - 虽然有堆栈随机化，不过 HeapSpray 技术将 ShellCode 布局到 0x0C0C0C0C（或者其他指定的地址上，通常这个地址要比较大），并不会受堆栈随机化的影响。其实，HeapSpray 中使用 ROP 绕过 DEP 的时候，就使用了前面提到的 攻击未启用ASLR的模块 。只是，HeapSpray 把 ShellCode 布局在堆上
- 覆盖部分返回地址
 - 映像随机化中，虽然模块的加载基址发生变化，但是各模块的入口点地址的低位字不变，只有高位字进行了随机化处理。
 - 对于地址 0x12345678，其中 5678 部分是固定的，如果存在缓冲区溢出，可以通过 memcpy 对后两个字节进行覆盖，可以将其设置为 0x12340000 ~ 0x1234FFFF 中的任意一个值。
 - 如果通过 strcpy 进行覆盖，因为 strcpy 会复制末尾的结束符 0x00，那么可以将 0x12345678 覆盖为 0x12345600，或者 0x12340001 ~ 0x123400FF
 - 部分返回地址覆盖，可以使得覆盖后的地址相对于基地址的距离是固定的，可以从基地址附近找可以利用的跳转指令
 - 这种方法的通用性不是很强，因为覆盖返回地址时栈上的 cookie 会被破坏。不过具体问题具体分析，为了绕过操作系统的安全保护机制需要考虑各种各样的情况
- Java Applet Spray
 - Java Applet 中动态申请的内存空间具有可执行属性（PAGE_EXECUTE_READWRITE），类似HeapSpray技术，可以在固定的地址上分配滑板指令(如NOP)和ShellCode，然后跳转到那个地址上

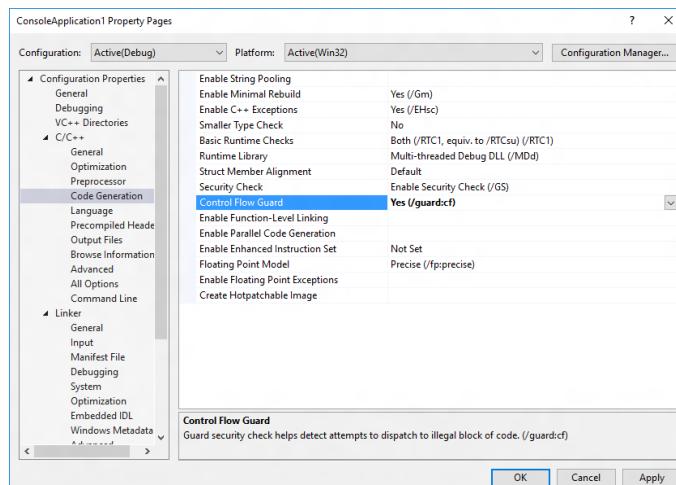
面去执行。和常规的HeapSpray不同，Applet申请空间的上限为100MB，而常规的HeapSpray可以达到1GB

- JIT Spray
 - JIT (Just In Time Compilation) 即时编译，也就是解释器（比如Python解释器）。主要思想是将 ActionScript代码中进行大量的XOR操作。然后编译成字节码，并且多次更新到Flash VM中，这样它会建立很多带有恶意Xor操作的内存块
- Tombkeeper在CanSecWest 2013上提出的基于SharedUserData的方法

crifan.com，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：2020-08-08 21:52:38

CFG

- CFG
 - = Control Flow Guard = 控制流保护
 - 是什么：一个高度优化的平台级的安全特性
 - 启用了CFG
 - 会告诉编译器，在编译期间，去分析代码的控制流，找到有哪些间接调用，并记录下来，用于分析编译出的二进制文件
 - Windows对于程序的每个间接调用都会做个检查
 - 如果检测发现不合法的，会抛出异常 `RaiseFastFailException`
 - 细节
 - 通过对一个程序可以从哪里开始运行加上严格的限制
 - 使得想要通过缓冲区溢出等漏洞去破解以执行任意代码的难度的大大增加
 - 支持情况
 - Microsoft Visual Studio 2015 之后才支持
 - 如何开启
 - 编译器参数
 - 语法： `/guard:cf`
 - 界面设置



- 如何确认一个程序支持了CFG？
 - 用 dumpbin 查看导出信息
 - `dumpbin /headers /loadconfig test.exe`
 - 输出中包含 Guard

```
OPTIONAL HEADER VALUES
    20B magic # (PE32+)
    12.10 linker version
    1BFEO0 size of code
    282600 size of initialized data
        200 size of uninitialized data
    9E090 entry point (000000014009E090)
    1000 base of code
    1400000000 image base (0000000140000000 to 0000000140447FFF)
    1000 section alignment
        200 file alignment
    10.00 operating system version
    10.00 image version
    10.00 subsystem version
        0 Win32 version
    448000 size of image
        400 size of headers
    4589A6 checksum
        2 subsystem (Windows GUI)
    C1C0 DLL characteristics
        Dynamic base
        Check integrity
        NX compatible
        Guard
    Terminal Server Aware
```

■ 加载配置中包含 CF Instrumented 和 FID table present

```
Section contains the following load config:
    000000A0 size
        0 time date stamp
    0.00 Version
        0 GlobalFlags Clear
        0 GlobalFlags Set
        0 Critical Section Default Timeout
        0 DeCommit Free Block Threshold
        0 DeCommit Total Free Threshold
    0000000000000000 Lock Prefix Table
        0 Maximum Allocation Size
        0 Virtual Memory Threshold
        0 Process Heap Flags
        0 Process Affinity Mask
        0 CSD Version
        0000 Reserved
    0000000000000000 Edit list
    000000014023C008 Security Cookie
    00000001401C41A0 Guard CF address of check-function pointer
    00000001401C41A8 Guard CF address of dispatch-function pointer
    00000001401C42A8 Guard CF function table
        E95 Guard CF function count
    00003500 Guard Flags
        CF Instrumented
        FID table present
        Protect delayload IAT
        Delayload IAT in its own section
```

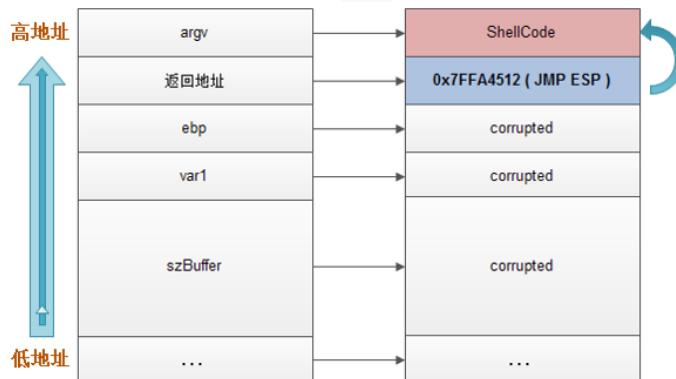
DEP

- DEP

- = Data Execution Prevention = 数据执行保护

- 背景

- 早期操作系统没有区分数据和代码， EIP 指向哪里就去哪里执行



- 含义：堆，栈上的内存页属性为不可执行，执行会出错

- 目的：防止某些内存区块，比如栈，被执行

- 如何开启

- 语法：

- 开启： /NXCOMPAT

- 关闭： /NXCOMPAT:NO

- 效果

- 开启了DEP后，能帮助检测到异常情况：



- 详解

- DEP 能够在内存上执行额外检查以帮助防止在系统上运行恶意代码
- DEP 是一套软硬件技术，能够在内存上执行额外检查以帮助防止在系统上运行恶意代码。在 Microsoft Windows XP Service Pack 2 及以上版本的 Windows 中，由硬件和软件一起强制实施 DEP
- 支持 DEP 的 CPU 利用一种叫做 No execute = 不执行 的技术识别标记出来的区域。如果发现当前执行的代码没有明确标记为可执行（例如程序执行后由病毒溢出到代码执行区的那部分代码），则禁止其执行，那么利用溢出攻击的病毒或网络攻击就无法利用溢出进行破坏了。如果 CPU 不支持 DEP，Windows 会以软件方式模拟出 DEP 的部分功能

- 相关

- 如果同时开启了 DEP 和 ASLR
 - 会让破解非常困难
 - 背景：一般用 shellcode 和 ROP 技术去破解的话
- ROP
 - = Return Oriented Programming
 - 早期叫： Ret2Libc
 - 实现原理
 - ROP 由一系列的 Gadget 组成
 - 所谓 ROP Gadget ，就是一系列以retn结尾的指令，所有的这些 Gadget 组合起来就能完成特定的任务
 - 比如调用 VirtualProtect 给指定的内存块添加可执行属性
- 寄存器
 - eip
 - esp + offset

crifan.com, 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：2020-08-09 09:34:20

GS

- GS

- = Buffer Security Check = 缓冲区安全检查

- 别称:

- Stack cookie
 - Security Cookie
 - GS cookie protection
 - GS security protection

- 是什么: 一个编译器参数

- 功能和作用: 用于缓存安全检查, 检查缓存是否溢出

- 决定编译器是否生成用于检测是否发生了缓冲区溢出 buffer overruns 的代码

- 如何设置

- Windows

- IDE: Visual Studio

- 编译器: MSVC Compiler

- 编译器参数: GS

- 如何配置:

- 开启: /GS

- 关闭: /GS-

- 代码中配置

- 文件级别

- strict_gs_check pragma

- 语法

- #pragma strict_gs_check([push,] { on | off })

- #pragma strict_gs_check(pop)

- 举例:

```
// pragma_strict_gs_check.cpp
// compile with: /c

#pragma strict_gs_check(on)

void ** ReverseArray(void **pData,
                     size_t cData)
{
    // *** This buffer is subject to being
    void *pReversed[20];

    // Reverse the array into a temporary
    for (size_t j = 0, i = cData; i ; --i
         // *** Possible buffer overrun!!
         pReversed[j] = pData[i];

    // Copy temporary buffer back into input
    for (size_t i = 0; i < cData ; ++i)
        pData[i] = pReversed[i];

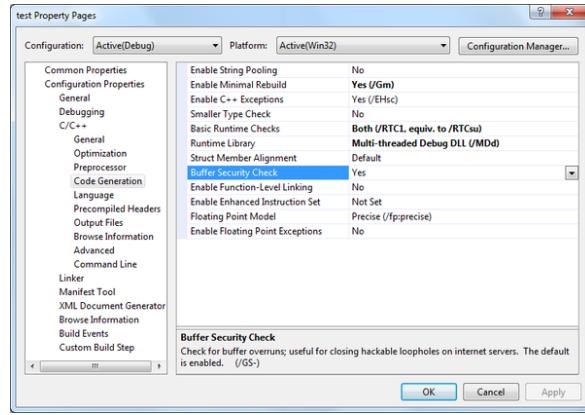
    return pData;
}
```

■ 函数级别

- 作用：指定某个函数不需要安全缓存检查
- 使用场景：你自己是个专家，会用手动做代码检查或者用其他手段确保代码很安全，无需检查
- 语法：`__declspec(safebuffers)`
- 举例：

```
// compile with: /c /GS
typedef struct {
    int x[20];
} BUFFER;
static int checkBuffers() {
    BUFFER cb;
    // Use the buffer...
    return 0;
};
static __declspec(safebuffers)
int noCheckBuffers() {
    BUFFER ncb;
    // Use the buffer...
    return 0;
}
int wmain() {
    checkBuffers();
    noCheckBuffers();
    return 0;
}
```

■ UI界面中配置



- Intel

- 编译器: Intel C++ Compiler

- 参数

- Windows :

- 语法: /GS[:keyword]

- keyword : GS的 level

- off -> /GS[:off] : 忽略, 关闭GS

- = /GS

- partial -> /GS[:partial] : 用 Microsoft

Visual Studio 2008 的标准=level

- strong -> /GS[:strong] : 提供完整的安全检查, 兼容最新版 Microsoft Visual Studio 的标准

- = /GS

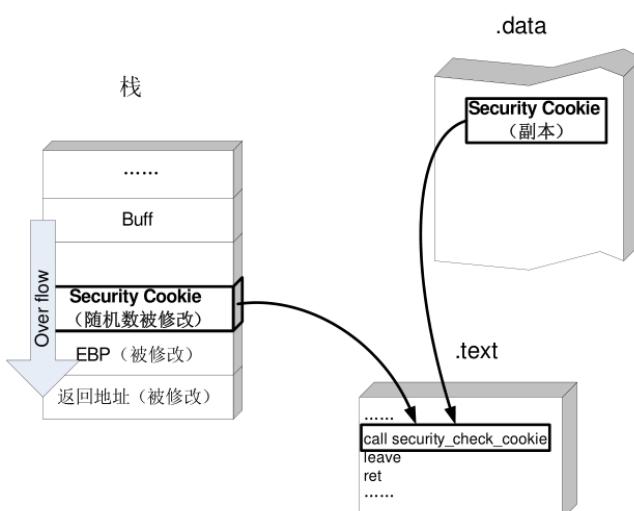
- Linnux / macos

- 开启: -fstack-security-check

- 关闭: -fno-stack-security-check

- 实现原理

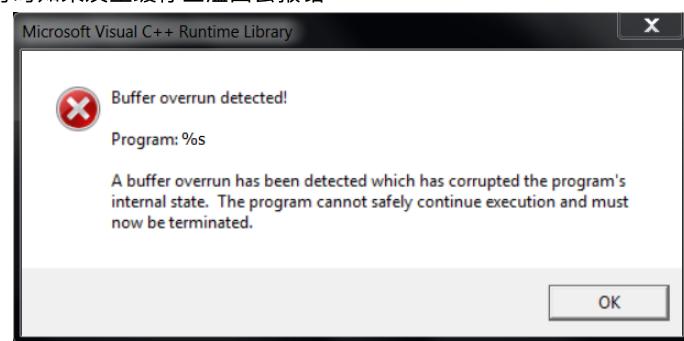
- 图解



- 文字解释

- 内部是利用 GS缓存区 = GS Buffers 实现缓冲区溢出的检测
- 其在编译器检测到的易受缓冲区溢出攻击的函数中创建 security cookie

- 如果攻击者写的代码，超过缓冲区长度，覆盖了 返回地址、异常处理程序的地址、易受攻击的函数参数，则运行时(runtime)会覆盖 security cookie。
 - 运行时会在允许执行代码跳转到此地址或返回这些函数参数之前，检测 cookie 的完整性，以避免攻击
- 使用效果
 - 运行时如果发生缓存区溢出会报错



如何绕开GS

- 利用未被保护的内存
 - 系统为了将GS对性能的影响降到最小，并不是所有的函数都会保护。例如，一个函数中不包含4字节以上的缓冲区时，即使GS处于开启状态，这个函数也是不受保护的。因此，可以针对这类函数，构造巧妙的shellcode进行溢出
- 通过猜测cookies值
 - GS保护机制采用了几个较弱的熵源，攻击者可以对其进行计算并使用计算结果来预测cookie值，但是这种办法只适用于针对本地系统的攻击（攻击者拥有该机器的访问权限）
 - 论文链接
 - <http://uninformed.org/?v=7&a=2&t=pdf>
- 覆盖虚函数(指针)
 - 程序只有在函数返回时，才会去检查 Security Cookie，而我们在程序检查 Security Cookie 之前劫持程序流程的话，就可以实现对程序的溢出。例如使用 C++ 的虚函数溢出即可实现上述功能
- 攻击异常处理
 - GS机制没有对 SEH 提供保护，因此可以通过攻击程序的异常处理机制达到绕过GS的目的。通过构造超长的字符串覆盖掉异常处理指针，然后触发一个异常，程序就会转入异常处理；由于异常处理指针已被覆盖，因此可以通过劫持SEH来控制程序的后续流程。
- 同时替换栈和 .data 中的 Cookie
 - 若要在 Security Cookie 正常工作的情况下实现绕过，由于 Cookie 具有很强的随机性，难以猜测，所以只能同时替换栈和 .data 中的 Cookie 实现绕过。构造特殊的 shellcode 使用相同的值覆盖栈和 .data 中的 Cookie，即可实现GS绕过。

相关知识

gcc编译器参数： **fstack-protector**

- gcc通用编译参数
 - `fstack-protector`
 - 功能: 开启或关闭某些或所有函数的栈溢出的安全检查
 - 语法
 - `Linux / macOS`
 - 语法
 - 开启: `-fstack-protector [-keyword]`
 - 关闭: `-fno-stack-protector [-keyword]`
 - 解释
 - `keyword`
 - `strong` -> `-fstack-protector-strong` : 任何类型的缓存(any type of buffer)都进行栈的溢出的安全检查
 - `all` -> `-fstack-protector-all` : 每个函数(every routine)都进行安全检查
 - 无参数-> `-fstack-protector` : 对于每个字符串缓存(string buffer)的栈溢出进行安全检查
 - 内部实现
 - 优先用 `gcc/glibc` 的实现
 - 如果没有, 其次用 `Intel` 的实现
 - 等价于 `-fstack-security-check`

crifan.com, 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: 2020-08-09 09:32:13

缓冲区溢出

- 缓冲区溢出
 - 会导致
 - 覆盖了
 - 函数返回值
 - overwrite a function's return address
 - 异常处理地址
 - exception handler address
 - 某些类型的参数
 - certain types of parameters
 - 黑客
 - 会利用 缓冲区溢出 去对于
 - 没有强制实现缓存大小的限制的那些代码
 - 实现漏洞检测和攻击

缓冲区溢出代码举例：

```
// compile with: /c /W1
#include <cstring>
#include <stdlib.h>
#pragma warning(disable : 4996) // for strcpy use

// Vulnerable function
void vulnerable(const char *str) {
    char buffer[10];
    strcpy(buffer, str); // overrun buffer !!

    // use a secure CRT function to help prevent buffer overruns
    // truncate string to fit a 10 byte buffer
    // strncpy_s(buffer, _countof(buffer), str, _TRUNCATE);
}

int main() {
    // declare buffer that is bigger than expected
    char large_buffer[] = "This string is longer than 10 characters!!";
    vulnerable(large_buffer);
}
```

crifan.com, 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：2020-08-08 12:16:58

SafeSEH

- SafeSEH
 - = Safe Exception Handlers
 - 作用：如果开启了 SafeSEH，那么编译器只会生成一个镜像，其 安全异常处理程序 的 静态表
 - 用告诉操作系统，此镜像文件的异常处理程序在哪
 - 目的：避免了攻击者重写异常处理程序的控制流程
 - 语法
 - 开启： /SAFESEH
 - 关闭： /SAFESEH:NO
 - 注：
 - 只支持： x86 平台
 - 不支持：那些已经标注了异常处理程序的平台
 - 举例
 - x64
 - 相关工具
 - ml64.exe = x64 的 Microsoft Assembler
 - 支持给程序生成 SEH 信息(XDATA 和 PDATA)信息
 - ARM
 - 这些平台：异常处理程序都已被放到了固定的 PDATA

crifan.com, 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：2020-08-08 14:26:39

安全学习方法

- 看漏洞公告
 - 在看的过程中理解漏洞
 - 在看的过程中学习调试和汇编指令
- 下载和研究POC
 - 来源
 - 看雪论坛
 - 等

crifan.com, 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: 2020-08-05 22:38:03

CTF比赛

TODO:

继续看完：

Getting Started - CTF Wiki

<https://ctf-wiki.github.io/ctf-wiki/>

并整理过来。

- CTF比赛
 - =夺旗赛
 - =CTF= Capture The Flag
 - 是什么：一个比赛
 - 在网络安全领域中指的是网络安全技术人员之间进行技术竞技的一种比赛形式
 - 起源：
 - 1996年DEFCON全球黑客大会，以代替之前黑客们通过互相发起真实攻击进行技术比拼的方式。发展至今，已经成为全球范围网络安全圈流行的竞赛形式
 - 2013年全球举办了超过五十场国际性CTF赛事。而DEFCON作为CTF赛制的发源地，DEFCON CTF也成为了目前全球最高技术水平和影响力的CTF竞赛，类似于CTF赛场中的"世界杯"
 - 竞赛模式
 - 解题模式
 - 在解题模式CTF赛制中，参赛队伍可以通过互联网或者现场网络参与，这种模式的CTF竞赛与ACM编程竞赛、信息学奥赛比较类似，以解决网络安全技术挑战题目的分值和时间来排名，通常用于在线选拔赛。题目主要包含逆向、漏洞挖掘与利用、Web渗透、密码、取证、隐写、安全编程等类别
 - 大多数为线上比赛，选手自由组队（人数不受限制），出题者把一些信息安全实战中可能遇到的问题抽象成一个题目，比如一个存在漏洞的网站让选手入侵，一个有漏洞的程序让选手分析来写出漏洞利用程序，一段密文让选手解密，一个图片选手从里面找出隐藏的线索等等。在完成这些出题的题目后，可以获得一串奇怪的字符串，也就是所谓的flag，提交它，就能获得这道题目的分数
 - 攻防模式=Attack-Defense
 - 在攻防模式CTF赛制中，参赛队伍在网络空间互相进行攻击和防守，挖掘网络服务漏洞并攻击对手服务来得分，修补自身服务漏洞进行防御来避免丢分。攻防模式CTF赛制可以实时通过得分反映比赛情况，最终也以得分直接分出胜负，是一种竞争激烈，具有很强观赏性和高度透明性的网络安全赛制。在这种赛制中，不仅仅是比参赛队员的智力和技术，也比体力（因为比赛一般都会持续48小时及以上），同时也比团队之间的分工配合与合作。
 - 大多数为线下比赛，参赛队伍人数有限制（通常为3到5人不等），参赛队伍保护自己的服务器，攻击其他队伍的服务器，每

个队伍的服务器开始拥有相同的配置和缺陷，比如几个有漏洞的二进制程序、有漏洞的Web应用、某些权限账户弱口令等等，然后队员需要找出这些漏洞并进行加固，同时利用这些漏洞来攻击别人的服务器，拿到其他队伍的权限后，会获取到相应flag后提交，从对方身上赚取相应的分数，每隔一段时间后，可以再次攻击并利用未加固的漏洞获取flag并赚取分数

- 混合模式=mix

- 结合了解题模式与攻防模式的CTF赛制，比如参赛队伍通过解题可以获取一些初始分数，然后通过攻防对抗进行得分增减的零和游戏，最终以得分高低分出胜负。采用混合模式CTF赛制的典型代表如iCTF国际CTF竞赛。
- 解题模式和攻防模式同时进行，解题模式可能会根据比赛的时间、进度等因素来释放需解答的题目，题目的难度越大，解答完成后获取的分数越高；攻防模式会贯穿整个CTF比赛的始终，参赛队伍需不断积累分数，最终参赛队伍的名次由两种模式累积的分数总和决定。有些有趣的CTF比赛，还会引入一些情景剧情和现场观众的互动，来增加比赛的趣味性

- 题目类别

- WEB=网络安全

- WEB是CTF竞赛的主要题型，题目涉及到许多常见的WEB漏洞，诸如XSS、文件包含、代码执行、上传漏洞、SQL注入。也有一些简单的关于网络基础知识的考察，例如返回包、TCP-IP、数据包内容和构造。可以说题目环境比较接近真实环境。
- 所需知识：PHP、python、TCP-IP、SQL

- MISC=安全杂项

- MISC即安全杂项，题目涉及隐写术、流量分析、电子取证、人肉搜索、数据分析、大数据统计等等，覆盖面比较广，主要考查参赛选手的各种基础综合知识。
- 所需知识：常见隐写术工具、wireshark等流量审查工具、编码知识。

- Crypto=密码学

- 题目考察各种加解密技术，包括古典加密技术、现代加密技术甚至出题者自创加密技术，以及一些常见编码解码，主要考查参赛选手密码学相关知识点。通常也会和其他题目相结合。
- 所需知识：矩阵、数论、古典密码学

- Reverse=逆向工程

- 题目涉及到软件逆向、破解技术等，要求有较强的反汇编、反编译扎实功底。主要考查参赛选手的逆向分析能力。
- 所需知识：汇编语言、加密与解密、常见反编译工具

- PPC=编程类题目

- 题目涉及到程序编写、编程算法实现，当然PPC相比ACM来说，还是较为容易的。至于编程语言嘛，推荐使用Python来尝试。题目较少，一般与其他类型相结合。
- 所需知识：基本编程思路、C,C++,Python,php皆可

- PWN=二进制安全

- PWN在黑客俚语中代表着攻破，取得权限，在CTF比赛中它代表着溢出类的题目，其中常见类型溢出漏洞有栈溢出、堆溢出。主要考察参赛选手对漏洞的利用能力。

- 所需知识: C, OD+IDA, 数据结构, 操作系统

- 基础知识

- 语言运用

- 计算机语言可以大致分为机器语言, 汇编语言, 高级语言, 计算机每进行的一次动作, 一个步骤, 都是按照计算机语言编好的程序来执行。而在CTF比赛中, 计算机语言的了解与掌握会有事半功倍的效果, 进程的动态调试、防护脚本的编写、源代码审计等工作都是建立在对计算机语言有所掌握的基础上进行的。

- Web安全

- 目前国内大多数CTF比赛都以Web安全为主, 但是Web安全涉及的内容非常广泛, 就典型的Web服务来说, 其安全问题可能来自于Web服务器、数据库、Web程序本身与开发语言等。了解一个Web应用的组成架构、装载与配置、指令操作及组件缺陷, 是参赛者知识储备环节中不可或缺的部分。

- 安全加固

- 安全领域的精髓在于攻防, 在CTF比赛也是同样的道理, 比赛成绩不仅取决于在有效的时间内拿下多少flag, 还取决于能抵御多少次外来攻击。有一些比赛队伍不注重或者不善于漏洞加固, 即使得到很多分数, 但是优势还是会慢慢的蚕食掉。所以, 了解漏洞的产生原因、减小漏洞的影响范围以及行之有效的安全加固也是一个成功队伍的重要能力。

- 密码算法

- 参赛者需要了解主流的密码算法, 如对称密码、公钥密码、流密码、哈希密码算法等。在不断的攻防对抗中, 一些关键信息或者突破口, 往往会通过算法的加解密将它们“隐藏”起来增加解题难度。此外还会伴随着弱口令尝试, 密码字典的暴力猜解等。

- 网络取证

- 对于网络攻击行为的溯源分析、漏洞挖掘过程中的抓包分析往往是很参赛队伍在攻防对抗中忽略的问题, 能够在最短的时间内抓到线索并做出行之有效的响应, 这方面的能力也就成为了高手和顶尖高手之间的分水岭, 古人常说: “天下大事, 必作于细; 天下难事, 必成于易”, 我想应该就是这个道理

CTF赛事

- 国外

- DEFCON CTF: CTF赛事中的“世界杯”
- UCSB iCTF: 来自UCSB的面向世界高校的CTF
- Plaid CTF: 包揽多项赛事冠军的CMU的PPP团队举办的在线解题赛
- Boston Key Party: 近年来崛起的在线解题赛
- Codegate CTF: 韩国首尔“大奖赛”, 冠军奖金3000万韩元
- Secuinside CTF: 韩国首尔“大奖赛”, 冠军奖金3000万韩元
- XXC3 CTF: 欧洲历史最悠久CCC黑客大会举办的CTF
- SIGINT CTF: 德国CCCAC协会另一场解题模式竞赛
- Hack.lu CTF: 卢森堡黑客会议同期举办的CTF
- EBCTF: 荷兰老牌强队Eindbazen组织的在线解题赛

- Ghost in the Shellcode：由Marauders和Men in Black Hats共同组织的在线解题赛
- RwthCTF：由德国OldEur0pe组织的在线攻防赛
- RuCTF：由俄罗斯Hackerdom组织，解题模式资格赛面向全球参赛，解题攻防混合模式的决赛面向俄罗斯队伍的国家级竞赛
- RuCTF：由俄罗斯Hackerdom组织面向全球参赛队伍的在线攻防赛
- PHD CTF：俄罗斯Positive Hacking Day会议同期举办的CTF
- 国内
 - XCTF全国联赛：中国网络空间安全协会竞评演练工作组主办、南京赛宁承办的全国性网络安全赛事平台，2014-2015赛季五站选拔赛分别由清华、上交、浙大、杭电和成信技术团队组织（包括杭电HCTF、成信SCTF、清华BCTF、上交OCTF和浙大ACTF），XCTF联赛总决赛由蓝莲花战队组织。XCTF联赛是国内最权威、最高技术水平与最大影响力的网络安全CTF赛事平台
 - AliCTF：由阿里巴巴公司组织，面向在校学生的CTF竞赛，冠军奖金10万元加BlackHat全程费用
 - XDCTF：由西安电子科技大学信息安全协会组织的CTF竞赛，其特点是偏向于渗透实战经验
 - HCTF：由杭州电子科技大学信息安全协会承办组织的CTF
 - 杭州电子科技大学信息安全协会由杭州电子科技大学通信工程学院组织建立，协会已有七年历史，曾经出征DEFCON,BCTF等大型比赛并取得优异成绩，同时协会还有大量有影响力的作品。协会内部成员由热爱黑客技术和计算机技术的一些在校大学生组成，有多个研究方向，主要有渗透，逆向，内核，web等多个研究方向。至今已经成功举办6次CTF比赛
 - ISCC：由北理工组织的传统网络安全竞赛，最近两年逐渐转向CTF赛制
 - TCTF：TCTF由中国网络空间安全协会竞评演练工作委员会指导、腾讯安全发起、腾讯安全联合实验室主办，Oops战队和北京邮电大学协办的CTF竞赛

CTF平台

FBCTF

- FBCTF
 - GitHub
 - <https://github.com/facebook/fbctf/>
 - 简介
 - 一套开源比赛平台，包括游戏地图、团队登记和评分系统。还可以按需提供逆向工程逆向工程、Web应用安全、取证、二进制开发和加密等挑战。用户还可以使用Facebook CTF平台定制或自定义挑战项目

绿盟科技CTF平台

- 绿盟科技CTF平台
 - 架构：

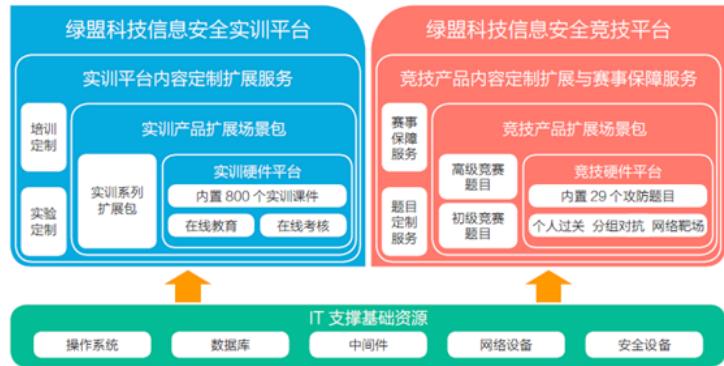


图1.1 绿盟科技信息安全攻防实训与竞技产品结构图

- 整体方案采用B/S架构对外提供新安全课件培训、实验训练和攻防保障服务。使用人员可以结合自身情况，灵活选取远程在线和线下面对面的实时教学形式。
- 两个平台可以支持进行课件培训、实验训练和攻防保障三大功能。课件培训主要以课件宣讲为主，实现信息安全知识的直接传递；实验训练以安全攻防模拟操作实验为主，使得被培训对象对安全技术的建立直观印象；攻防保障主要为被培训对象提供攻防的虚拟环境，实际检验被培训对象的安全水平。
- IT支撑基础资源主要包括安操作系统、数据库、中间件、网络设备和安全设备等，通过与虚拟技术相结合，用以保障上层的实训场景和竞技场景。

○ 绿盟信息安全实训系统

- ISTS -Information Security Trainning System
 - 信息安全培训、教学及科研提供一个完整的、一体化的实验教学环境



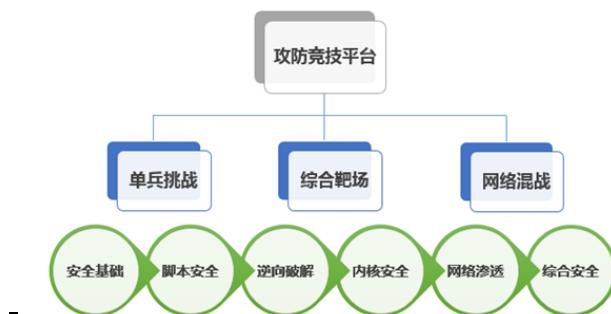
○ 绿盟信息安全竞技系统

- ISCS -Information Security Competition System
 - 围绕信息安全理论和知识，组建的信息安全对抗技能实战赛。也可以承载信息安全对抗攻防演练项目，考察攻防演练者网络安全理论知识与实际问题处理能力，旨在借助攻防演练培养一批具备信息安全素养的优秀实战专业的安全人员。
- 模式：
 - 安全攻防竞赛平的攻防竞赛模式可以分为三种形式
 - 单兵作战挑战赛=个人挑战模式
 - 个人挑战模式每个赛题（关卡）是一个单独的靶场，并提供了七大类赛题，WEB、密码学、隐写、溢出、逆

向、编程、综合，全面考察参赛选手的安全能力，题目由易到难。



- 综合靶场
- 网络混战



看雪CTF

- 看雪CTF
 - <https://ctf.pediy.com>
 - 看雪CTF（简称KCTF）是圈内知名度最高的技术竞技，从原CrackMe攻防大赛中发展而来，采取线上PK的方式，规则设置严格周全，题目涵盖Windows、Android、iOS、Pwn、智能设备、Web等众多领域

crifan.com, 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：2020-08-04 22:36:40

正向安全

- 正向安全 防护手段
 - 混淆
 - Android
 - 加入 花指令
 - 用于干扰的垃圾代码
 - Mac
 - iOS
 - OC的混淆
 - 加壳
 - 加固=防护 的一种手段
 - 不同系统=平台
 - Windows
 - vmprotect
 - 强壳
 - Linux
 - upx
 - Android
 - Mac
 - App Store的ipa
 - SO的保护
 - 是什么
 - APP的核心（认证逻辑， 加密等） 算法采用C/C++编写并编译为SO文件
 - 目的 = 为何要SO防护
 - 增加被破解=反编译的难度
 - 增加黑客利用其业务的难度
 - 优势
 - SO中为原生ARM汇编， 难以还原原始代码
 - 对比： DEX文件很容易被各种反编译工具直接还原成通俗易懂的Java代码
 - SO调试成本高
 - 对比： Java写的程序更容易被调试
 - 如使用SmaliIdea、 Jeb、 IDA、 Xposed、 插桩打日志等多种方式
 - SO难以在x86生产环境中黑盒调用
 - 对比： DEX文件可转换成class文件，在生产环境中使用JNI直接传参调用
 - 手段
 - 反调试
 - 区块加密
 - OLLVM 混淆
 - OLLVM 混淆是逆向人员的噩梦， 这招确实能有效提高 so 代码的安全性
 - 破解手段

- unicorn
- ARM VMP
 - ARM VMP 兼容性问题比较多，还无法商业化

crifan.com, 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：2020-08-10 09:07:45

漏洞编号

网络已存在的公开漏洞，一般都有个 漏洞编号

目前主要有2个漏洞编号相关组织：

- CVE
- CNVD

crifan.com, 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：2020-08-06 22:58:16

CVE

- CVE
 - 名称
 - CVE = Common Vulnerabilities and Exposures = 通用漏洞披露
 - 别称： 常见弱点与漏洞
 - 是什么： 一个与信息安全有关的数据库
 - 注意
 - 通常我们谈到的 CVE 指的是 CVE 编号 = CVE 漏洞编号
 - 是分配给每个安全漏洞的 CVE ID 编号
 - 作用： 收集各种信息安全弱点及漏洞并给予编号以便于公众查阅
 - 列出了已公开披露的各种计算机安全漏洞
 - 目的： 帮助 IT 专业人员协调自己的工作， 轻松地确定漏洞的优先级并加以处理， 从而提高计算机系统的安全性
 - 运营： 现由美国非营利组织 MITRE 所属的 National Cybersecurity FFRDC 所营运维护

CVE漏洞编号

- CVE 漏洞编号
 - 每一个通用漏洞披露都赋予一个专属的编号
 - 格式：
 - CVE-YYYY-NNNN
 - CVE： 固定的前缀字
 - YYYY： 西元纪年
 - NNNN： 流水编号
 - 原则上为四位数字， 但必要时可编到五位数或更多位数
 - 例如
 - 于 2014 年发现的心脏出血漏洞 编号为 CVE-2014-0160

CVE 申请需要满足什么条件？

- 只有满足一系列特定条件的漏洞才会分配 CVE ID
 - 这些漏洞必须满足以下条件：
 - - 1. 可以单独修复。
 - 2. 该漏洞可以独立于所有其他错误进行修复。
 - 2.
 - 已得到相关供应商的确认。
 - 软件或硬件供应商已确认错误，并承认其会对安全性造成负面影响。
 - 或者
 - 已记录在案。
 - 错误报告者已共享了一份相关的漏洞报告，表明错误会造成负面影响，且有悖于受影响系统的安全策略。

1. 会影响某个代码库。
2. 如果漏洞会对多个产品造成影响，则会获得单独的 CVE。对于共享的库、协议或标准，只有在使用共享代码会容易受到攻击时，该漏洞才会获得单个 CVE。否则，每个受影响的代码库或产品都会获得一个唯一的 CVE

CVE漏洞编号是谁颁发的？

CVE开始是由 MITRE Corporation 负责日常工作的。但是随着漏洞数量的增加，MITRE 将漏洞编号的赋予工作转移到了其 CNA = CVE Numbering Authorities 成员。

CNA 涵盖5类成员，目前共有69家成员单位：

1. MITRE：可为所有漏洞赋予 CVE 编号；
2. 软件或设备厂商：如 Apple、Check Point、ZTE 为所报告的他们自身的漏洞分配 CVE ID。该类成员目前占总数的80%以上。
3. 研究机构：如 Airbus，可以为第三方产品漏洞分配编号；
4. 漏洞奖金项目：如 HackerOne，为其覆盖的漏洞赋予 CVE 编号；
5. 国家级或行业级CERT：如 ICS-CERT、CERT/CC，与其漏洞协调角色相关的漏洞。

crifan.com，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：2020-08-06 23:04:44

CNVD

- CNVD
 - = China National Vulnerability Database =中国 国家信息安全漏洞共享平台
 - 一句话介绍：中国版的 CVE
 - 简介
 - 由 国家计算机网络应急技术处理协调中心 （中文简称 国家互联应急中心，英文简称 CNCERT ）联合国内重要信息系统单位、基础电信运营商、网络安全厂商、软件厂商和互联网企业建立的国家网络安全漏洞库
 - 资料
 - 官网
 - 国家信息安全漏洞共享平台
 - <https://www.cnvd.org.cn>

crifan.com, 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：2020-08-06 23:02:07

Web安全

此处整理和Web网络相关的安全相关知识。

- Web安全
 - 根据攻防角度分
 - 进攻
 - 名字和概念
 - 漏洞扫描
 - 端口扫描
 - Web攻击 = Web漏洞攻击
 - Web挖掘 = Web漏洞挖掘
 - Web渗透
 - 攻击方式
 - SQL注入
 - XSS
 - CSRF
 - 越权
 - 文件包含
 - 文件上传
 - 命令执行
 - WAF绕过
 - URL跳转
 - 钓鱼
 - 社工 = 社会工程学
 - 防守
 - 代码审计 = 安全代码审计 = 安全审计
 - 目的：写出高质量的漏洞少的代码
 - 日志分析 = 安全日志分析 = 日志关联分析
 - 深度包检测
 - 程序行为监视
 - 防护设备
 - 防火墙
 - WAF = Web应用程序防火墙
 - IDS = Intrusion Detection Systems = 入侵检测系统
 - IPS = Intrusion Prevention Systems = 入侵防御系统
 - 相关组织和标准
 - 组织：OWASP
 - 标准：OWASP10
 - 主要工作方向和内容
 - 渗透测试
 - 漏洞挖掘
 - 安全开发
 - 代码审计
 - 代码审计 = 代码安全审计 = 安全编码审计 = 源代码审计 = 源代码安全分析
 - 网络安保

渗透测试

详见独立教程：

- [潜入你的网络：渗透测试](#)

crifan.com, 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：2021-05-24 21:57:13

常用工具

此处整理Web安全中常用的一些工具。

crifan.com, 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: 2020-08-02 20:24:53

安全操作系统

crifan.com, 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: 2020-08-02 20:24:38

Kali Linux

- = Kali Linux
- 是什么：一个Linux操作系统，专门用于渗透测试，
- Kali
 - = Kali Linux
 - 旧称： BackTrack Linux
 - 是什么：一个操作系统
 - 用途：专门用于安全、逆向、破解的
 - 特点：自带大量相关工具
 - 被称为
 - 网络安全人员的专用系统
 - 资料
 - 主页
 - Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution
 - <https://www.kali.org>
 - 包含工具
 - 首页
 - Penetration Testing Tools - Kali Linux
 - <https://tools.kali.org>
 - Kali Linux Tools Listing | Penetration Testing Tools
 - <https://tools.kali.org/tools-listing>
 - 根据分类
 - Information Gathering
 - ace-voip
 - Amap
 - APT2
 - arp-scan
 - Automater
 - bing-ip2hosts
 - braa
 - CaseFile
 - CDPSnarf
 - cisco-torch
 - copy-router-config
 - DMitry
 - dnmap
 - dnsenum
 - dnsmap
 - DNSRecon
 - dnstracer
 - dnswalk
 - DotDotPwn
 - enum4linux
 - enumIAX

- EyeWitness
- Faraday
- Fierce
- Firewalk
- fragroute
- fragrouter
- Ghost Phisher
- GoLismero
- goofile
- hping3
- ident-user-enum
- InSpy
- InTrace
- iSMTP
- lbd
- Maltego Teeth
- masscan
- Metagoofil
- Miranda
- nbtscan-unixwiz
- Nikto
- Nmap
- ntop
- OSRFramework
- p0f
- Parseo
- Recon-ng
- SET
- SMBMap
- smtp-user-enum
- snmp-check
- SPARTA
- sslaudit
- SSLsplit
- sslstrip
- SSLyze
- Sublist3r
- THC-IPV6
- theHarvester
- TLSSLed
- twofi
- Unicornscan
- URLCrazy
- Wireshark
- WOL-E
- Xplico
- Vulnerability Analysis
- BBQSQL

- BED
- cisco-auditing-tool
- cisco-global-exploiter
- cisco-ocs
- cisco-torch
- copy-router-config
- Doona
- DotDotPwn
- HexorBase
- jSQL Injection
- Lynis
- Nmap
- ohrwurm
- openvas
- Oscanner
- Powerfuzzer
- sfuzz
- SidGuesser
- SIPArmyKnife
- sqlmap
- Sqlninja
- sqsus
- THC-IPV6
- tnscmd10g
- unix-privesc-check
- Yersinia
- Exploitation Tools
- Armitage
- Backdoor Factory
- BeEF
- cisco-auditing-tool
- cisco-global-exploiter
- cisco-ocs
- cisco-torch
- Commix
- crackle
- exploitdb
- jboss-autopwn
- Linux Exploit Suggester
- Maltego Teeth
- Metasploit Framework
- MSFPC
- RouterSploit
- SET
- ShellNoob
- sqlmap
- THC-IPV6
- Yersinia

- Wireless Attacks
 - Airbase-ng
 - Aircrack-ng
 - Airdecap-ng and Airdecloak-ng
 - Aireplay-ng
 - airgraph-ng
 - Airmon-ng
 - Airodump-ng
 - airodump-ng-oui-update
 - Airolin-ng
 - Airserv-ng
 - Airtun-ng
 - Asleap
 - Besside-ng
 - Bluelog
 - BlueMaho
 - Bluepot
 - BlueRanger
 - Bluesnarfer
 - Bully
 - coWPAtty
 - crackle
 - eapmd5pass
 - Easside-ng
 - Fern Wifi Cracker
 - FreeRADIUS-WPE
 - Ghost Phisher
 - GISKismet
 - Gqrx
 - gr-scan
 - hostapd-wpe
 - ivstools
 - kalibrate-rtl
 - KillerBee
 - Kismet
 - makeivs-ng
 - mdk3
 - mfcuk
 - mfoc
 - mfterm
 - Multimon-NG
 - Packetforge-ng
 - PixieWPS
 - Pyrit
 - Reaver
 - redfang
 - RTLSDR Scanner
 - Spooftooth

- Tkiptun-ng
- Wesside-ng
- Wifi Honey
- wifiphisher
- Wifitap
- Wifite
- wpaclean
- Forensics Tools=取证工具
 - Binwalk
 - bulk-extractor
 - Capstone
 - chntpw
 - Cuckoo
 - dc3dd
 - ddrescue
 - DFF
 - diStorm3
 - Dumpzilla
 - extundelete
 - Foremost
 - Galleta
 - Guymager
 - iPhone Backup Analyzer
 - p0f
 - pdf-parser
 - pdfid
 - pdgmail
 - peepdf
 - RegRipper
 - Volatility
 - Xplico
- Web Applications
 - apache-users
 - Arachni
 - BBQSQL
 - BlindElephant
 - Burp Suite
 - CutyCapt
 - DAVTest
 - deblaze
 - DIRB
 - DirBuster
 - fimap
 - FunkLoad
 - Gobuster
 - Grabber
 - hURL
 - jboss-autopwn

- joomscan
- jSQL Injection
- Maltego Teeth
- Nikto
- PadBuster
- Paros
- Parsero
- plecost
- Powerfuzzer
- ProxyStrike
- Recon-ng
- Skipfish
- sqlmap
- Sqlninja
- sqlsus
- ua-tester
- Uniscan
- w3af
- WebScarab
- Webshag
- WebSlayer
- WebSploit
- Wfuzz
- WhatWeb
- WPScan
- XSSer
- zaproxy
- Stress Testing
 - DHCPig
 - FunkLoad
 - iaxflood
 - Inundator
 - inviteflood
 - ipv6-toolkit
 - mdk3
 - Reaver
 - rtpflood
 - SlowHTTPTest
 - t50
 - Termineter
 - THC-IPV6
 - THC-SSL-DOS
- 其他方面对Kali的支持
 - Hopper Disassembler
 - Hopper - Download
 - <https://www.hopperapp.com/download.html?>
 - 专门提供Kali Linux的zip压缩包

组织和标准

crifan.com, 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: 2020-08-01 09:30:53

OWASP10

- 在Web安全领域，有个组织叫：
 - OWASP
 - = Open Web Application Security Project
 - = 开源Web应用安全项目
 - = 开源Web应用安全组织
- 该组织每年会推出一个 标准： OWASP 10
 - OWASP列出了最重要的10个方面的安全攻击
 - 说明
 - 列出排名前10的攻击类型
 - 每年都会出一个报告
 - 最早： 2003年
 - 最新： 2017年
- 2017年的OWASP 10
 - Injection=注入攻击
 - 涉及方面
 - SQL
 - SQL Injection = SQL注入
 - NoSQL
 - OS
 - LDAP
 - LDAP Injection
 - 坏结果
 - 运行了不该运行的（恶意的）代码
 - Expression Language (EL) Injection
 - Command Injection
 - 获取了不该获取的数据=盗取数据
 - 心得
 - 编写接受数据的模块时要非常小心
 - 举例
 - request.getParameter()
 - request.getCookie()
 - request.getHeader()
 - Broken Authentication
 - =失效的身份
 - Sensitive Data Exposure
 - XXE
 - XML External Entities
 - Broken Access Control
 - =访问控制缺失
 - Security Misconfiguration
 - =安全配置错误
 - XSS
 - =Cross-Site Scripting
 - Insecure Deserialization

- Using Components with Known Vulnerabilities
 - =使用含有已知漏洞的组件
- Insufficient Logging & Monitoring
- 中文主页
 - Welcome to OWASP CHINA — OWASP-CHINA
 - <http://www.owasp.org.cn>

crifan.com, 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: 2020-08-04 22:04:53

设备端安全

和Web网络相对应的，可以统称为 设备端的安全。

主要包括：

- PC端
 - Windows
 - Mac
 - Linux
- 移动端
 - Android
 - iOS
- IoT =物联网设备

下面根据不同维度详细介绍。

可执行文件 逆向工程 工具

- Windows 的 PE 格式的 exe文件 、 dll文件
 - OllyDBG
 - IDA PRO
 - 二进制分析
 - Hiew
 - 反汇编 + 16进制编辑器
 - 命令行, 无GUI
- Linux 的 ELF 格式的文件
 - GDB
 - IDA PRO
 - Hopper
 - Disassembler + Pseudo C decompiler
 - Evan's debugger
 - Linux中类似于OllyDBG的工具
 - Insight
 - GDB的GUI
 - 有点过时了
- Mac 的 MACH-O 格式的文件
 - Hopper
 - IDA PRO
 - LLDB
 - MachOView
- Android 的 dex 格式的文件 (apk文件内的)
 - APK TOOL
 - Disassembler and Assembler (SMALI)
 - JEB
 - Android disassembler (SMALI) and decompiler (JAVA)
 - IDA PRO

- iOS 的可执行文件
 - IDA Pro
 - Hopper
 - otool

TODO:

【未解决】Mac中有哪些常用的破解逆向方面的工具软件

crifan.com, 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: 2020-08-06 22:31:37

计算机安全

- PC端：计算机安全
 - 多平台
 - IDA
 - radare2
 - Windows
 - Windows安全
 - 调试工具
 - OD = OllyDbg = Olly DBG
 - WinDBG
 - LLDB
 - Mac
 - 工具
 - Hopper Disassemble
 - Linux
 - LLDB
 - GDB
- 对比
 - 静态分析：[IDA](#)
 - 支持插件：
 - 最强大的：[Hex-rays](#)
 - 把汇编语言转换成C语言伪代码
 - 动态调试-》调试器：[WinDGB](#)、[OllyDBG](#)

crifan.com, 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：2020-08-08 22:00:06

Windows安全

- CAIN
 - 是什么：Oxid.it开发的一个针对Microsoft操作系统的免费口令恢复和网络嗅探测试工具
 - 特点：在口令破解上很有一套技术
 - 功能
 - 网络嗅探，网络欺骗，破解加密口令、解码被打乱的口令、显示口令框、显示缓存口令和分析路由协议，甚至还可以监听内网中他人使用VOIP拨打电话等。

软件逆向

- 软件逆向
 - =软件逆向工程
 - 定义
 - 通过反汇编和调试等手段，分析计算机程序的二进制可执行代码从而获得程序的算法细节和实现原理的技术。
 - 研究对象
 - 没有公开源代码的计算机程序
 - 主要是已经经过编译的二进制可执行代码
 - 举例
 - win32平台上
 - PE文件
 - 常见文件格式
 - exe
 - dll
 - 分类
 - 系统级逆向
 - 大范围分析观察，整体把握
 - 代码级逆向
 - 程序二进制码中提取设计理念和算法
 - 步骤
 - 研究保护方法，去除保护功能
 - 解码/反汇编（目标二进制代码）
 - 反汇编目标软件，定位功能函数
 - 中间语言翻译（汇编或类汇编代码）
 - 分析汇编代码
 - 数据流分析（各级中间语言）
 - 修改汇编代码或还原高级源代码
 - 其他分析和优化（高级抽象代码）
 - 工具
 - OllyDbg：动态追踪工具，插件较多较多
 - Windbg：用户态和内核态调试工具
 - IDA：交互式反汇编器
 - PEID：著名的查壳工具

- C32Asm : 反汇编程序，可直接修改软件内部代码，有十六进制编辑模式
- 主要应用
 - 软件破解：破解软件的版权让用户不支付授权费用就可以使用软件的全部功能。
 - 病毒和恶意程序的分析：恶意程序的传播机制和危害并设计出解，分析病毒解决办法。
 - 系统漏洞分析：分析漏洞原理，设计补丁程序或者编写利用程序（Exploit）
 - 分析不公开的文件格式，协议等
 - 分析windows或mac平台上的硬件驱动程序编写linux下的相应驱动
 - 挖掘消费电子产品的潜能
 - 挖掘操作系统未文档化的API，发现更多内幕
 - 计算机犯罪取证

crifan.com, 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：2020-08-05 21:27:21

反汇编器

- `disassembler` = 反汇编
 - 反汇编引擎/框架
 - 名称
 - 反汇编引擎 = `disassembler engine`
 - 反汇编框架 = `disassembler framework`
 - 常见
 - `llvm`
 - `capstone`
 - `OllyDbg` 的 `ODDisasm`
 - `BeaEngine`
 - `udis86`
 - `XDE`
 - 对比
 - `udis86 vs BeaEngine vs capstone`
 - 细节
 - 性能: `udis86 > BeaEngine > capstone`
 - 解码能力: `capstone > BeaEngine > udis86` (`udis86`不支持寄存器分析,其余解码能力是相近的)
 - 平台支持: `capstone > (udis86 = BeaEngine)`
 - X86扩展指令集: `capstone > (udis86 ≈ BeaEngine)`
 - 结论
 - 如果你需要的是一个X86/64下的性能又好同时解码能力又强的反汇编引擎, 并且不需要寄存器分析这种特技的话, 那么`udis86`合适你
 - 如果你还需要带寄存器分析功能的话, 那么`BeaEngine`与`capstone`合适你; 如果你还需要ARM构架支持的话, `capstone`应该会更适合你

crifan.com, 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: 2020-08-08 23:03:02

Capstone

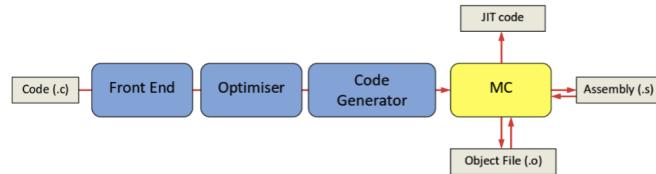
- Capstone
 - 一句话描述:
 - a lightweight multi-platform, multi-architecture disassembly framework
 - 一个轻量级的支持多平台和多架构的反汇编框架
 - 特点:
 - lightweight 轻量级
 - 简洁的API
 - Clean/simple/lightweight/intuitive architecture-neutral API
 - 多种语言接口Bindings
 - Python, Ruby, Go, NodeJS等
 - multi-platform 支持多平台
 - Windows & *nix (with Mac macOS, iOS, Android, Linux, *BSD & Solaris confirmed)
 - multi-architecture 支持多种架构
 - Arm, Arm64 (Armv8), BPF, Ethereum Virtual Machine, M68K, M680X, Mips, MOS65XX, PowerPC, RISCV, Sparc, SystemZ, TMS320C64X, Web Assembly, XCore & X86 (include X86_64)
 - 用途=应用领域
 - 安全领域
 - 二进制分析 binary analysis
 - 逆向 reversing
 - 号称
 - Next Generation Disassembler Engine
 - 更好的下一代反汇编引擎
 - 特别说明
 - 著名的开源逆向工具 Radare2 以及商业逆向工具 IDA Pro 的三方插件都基于 Capstone
 - 主页
 - 官网
 - The Ultimate Disassembly Framework – Capstone – The Ultimate Disassembler
 - <http://www.capstone-engine.org>
 - GitHub
 - aquynh/capstone: Capstone disassembly/disassembler framework: Core (Arm, Arm64, BPF, EVM, M68K, M680X, MOS65xx, Mips, PPC, RISCV, Sparc, SystemZ, TMS320C64x, Web Assembly, X86, X86_64, XCore) + bindings.
 - <https://github.com/aquynh/capstone>

Capstone vs LLVM

- capstone源自 LLVM 编译器框架中的MC模块

- MC 模块中有个反汇编引擎叫做 MCDisassembler

- MC = Machine Code
- 机制：



- 而 llvm 甚至还有个工具叫做： llvm-mc

- 可以用于反汇编输入的二进制文件

- capstone 才用了 MCDisassembler 作为核心内容
 - 但又经过了大量优化改动，以适配自己的设计
 - capstone 在 MCDisassembler 在基础上加了其他的大量的功能
 - -> MCDisassembler 能做的 capstone 都能做
- capstone 和 llvm-mc 的区别
 - 详见
 - https://www.capstone-engine.org/beyond_llvm.html

其他相关

- capstone转llvm

- chubbymaggie/capstone2llvmir: Library for Capstone instruction to LLVM IR translation
 - <https://github.com/chubbymaggie/capstone2llvmir>

- 成套工具

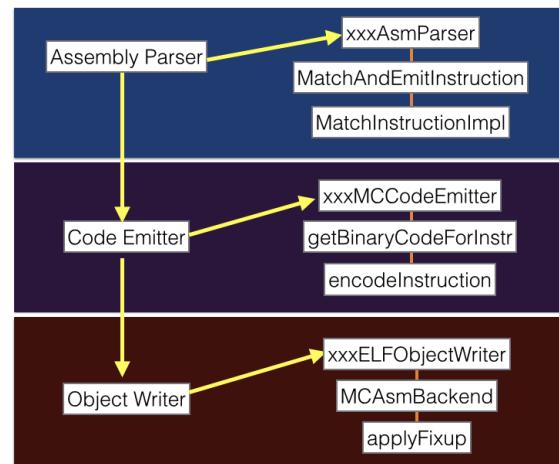
- 3件套

- Logo



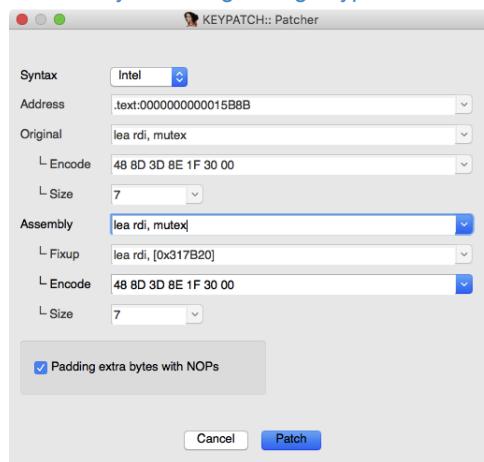
- 3个项目

- Capstone disassembler
 - Next Generation Disassembler Engine
 - <http://capstone-engine.org>
- Unicorn emulator
 - Next Generation CPU Emulator
 - <http://unicorn-engine.org>
- Keystone assembler
 - <http://keystone-engine.org>
 - 流程



■ IDA插件

- Keypatch – Keystone – The Ultimate Assembler
 - <https://www.keystone-engine.org/keypatch/>

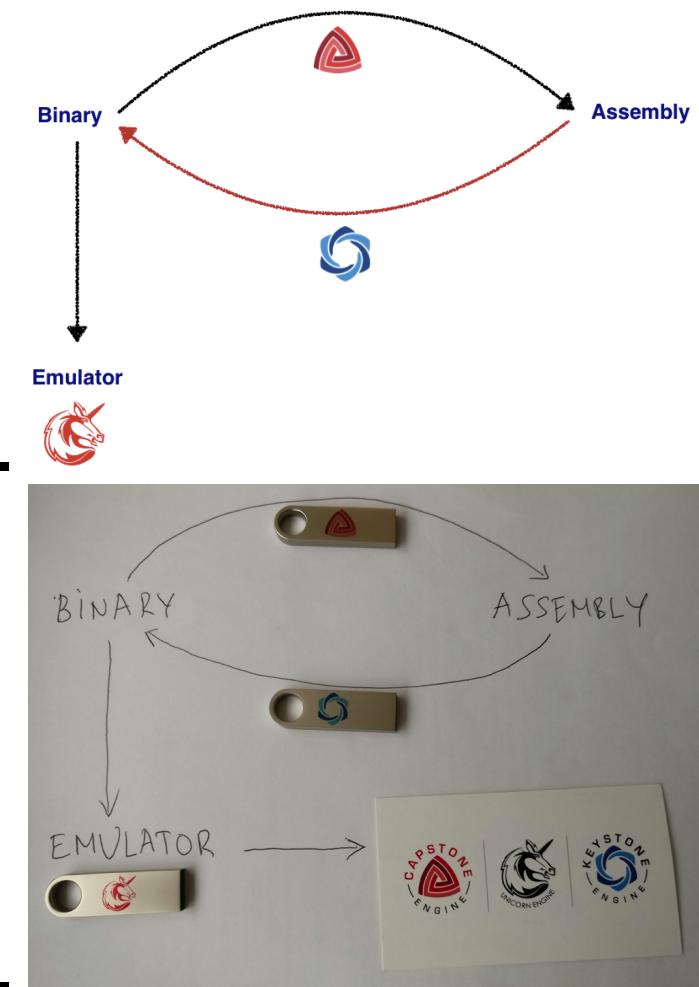


■ 竞品

- Radare2
 - Unix-like reverse engineering framework and commandline tools
- Pwnypack
 - CTF toolkit with Shellcode generator Ropper: Rop gadget and binary information tool
- GEF
 - GDB plugin with enhanced features
- Uservcorn
 - Versatile kernel+system+userspace emulator
- X64dbg
 - An open-source x64/x32 debugger for windows
- Liberation
 - code injection library for iOS
- Demovfuscator
 - Deobfuscator for movfuscated binaries

■ 效果：

- Fundamental frameworks for Reverse Engineering



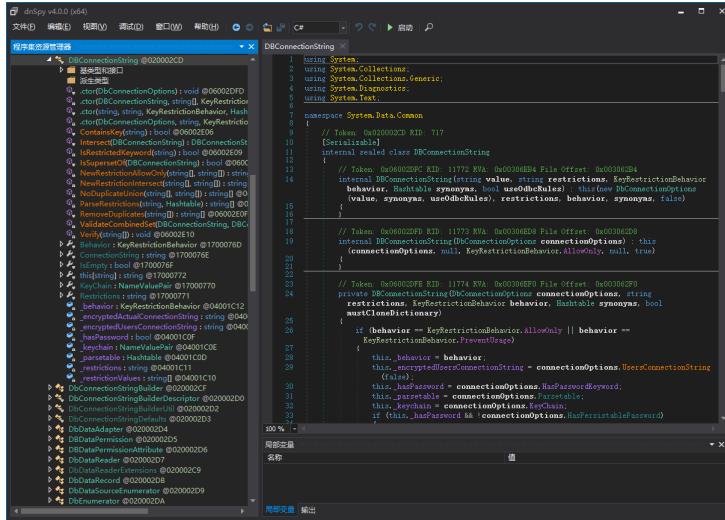
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2020-08-09 09:46:19

反编译器

crifan.com, 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: 2020-08-08 21:32:44

dnSpy

- dnSpy
 - 一句话描述: a debugger and .NET assembly editor
 - 特点: 无需源码也能修改 .NET 程序
 - 功能
 - 功能组件
 - 编译器
 - 调试器
 - 汇编编辑器
 - 支持编写插件扩展功能
- 截图



- 资料

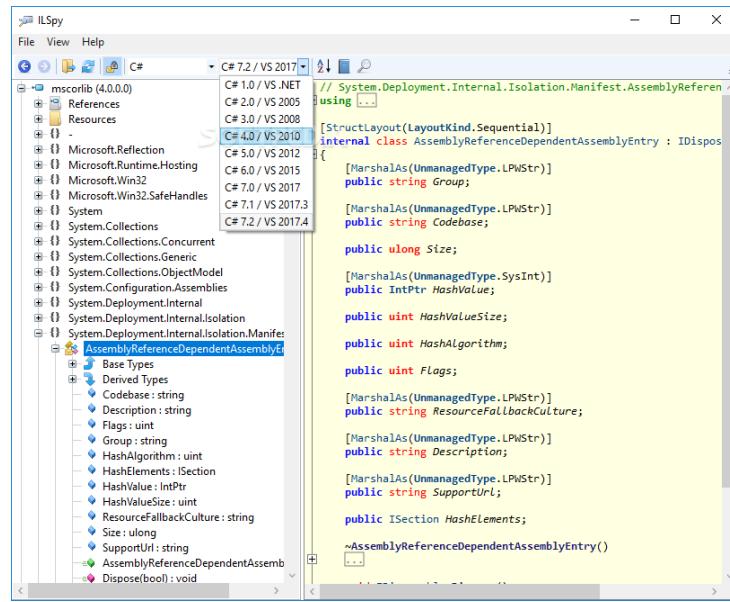
- Github
 - <https://github.com/0xd4d/dnSpy>

crifan.com, 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: 2020-08-08 21:44:02

ILSpy

- ILSpy

- 截图



- 资料

- Github

- [icsharpcode/ILSpy: .NET Decompiler with support for PDB generation, ReadyToRun, Metadata \(&more\) - cross-platform!](https://github.com/icsharpcode/ILSpy)
- <https://github.com/icsharpcode/ILSpy>

crifan.com, 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: 2020-08-08 22:06:18

静态安全检工具

crifan.com, 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: 2020-08-08 12:23:26

winchecksec

- winchecksec
 - 作用: Windows静态安全检测工具
 - 支持检测如下 安全特性
 - ASLR
 - /DYNAMICBASE with stripped relocation entries edge-case
 - /HIGHENTROPYVA for 64-bit systems
 - Code integrity/signing:
 - /INTEGRITYCHECK
 - Authenticode-signed with a valid (trusted, active) certificate
(currently unsupported on Linux)
 - DEP
 - 别称: W^X , NX
 - Manifest isolation
 - /ALLOWISOLATION
 - SEH 和 SafeEH
 - SEH = Structured Exception Handling
 - Control Flow Guard 和 Return Flow Guard instrumentation
 - Stack cookie
 - /GS
 - 资料
 - Github
 - trailofbits/winchecksec: Checksec, but for Windows: static detection of security mitigations in executables
 - <https://github.com/trailofbits/winchecksec>

crifan.com, 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: 2020-08-08 12:26:52

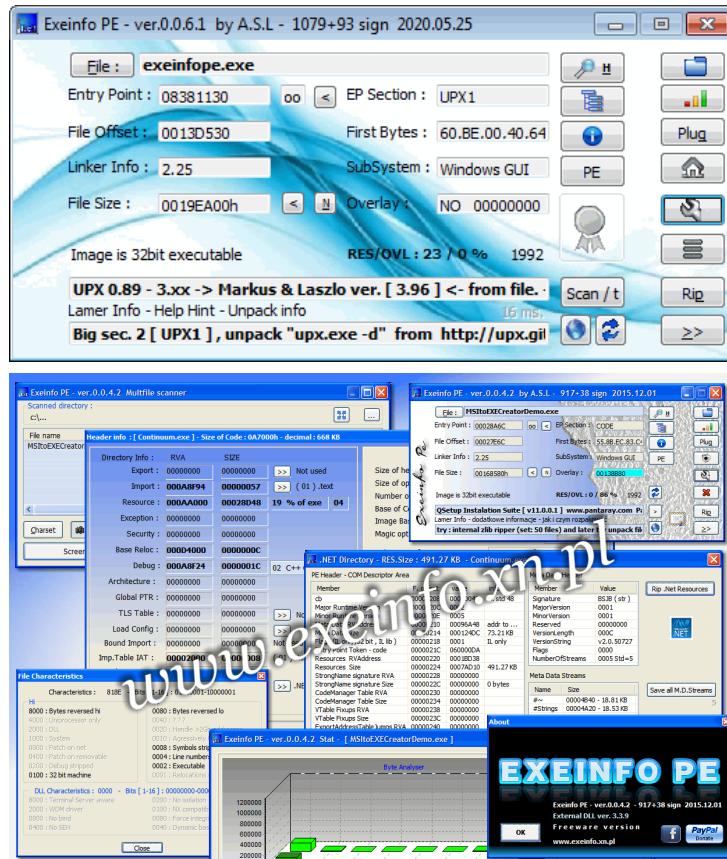
可执行文件工具

crifan.com, 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: 2020-08-05 22:02:10

Exeinfo PE

- Exeinfo PE

- 一句话描述: Packer compressor detector / unpack info / internal exe tools
- 支持平台
 - Android
 - Linux
 - Mac
- 截图



- 资料

- 官网

- Exeinfo PE by A.S.L - compression detector and data detector
 - <http://www.exeinfo.xn.pl>
 - mirror
 - <http://exeinfo.byethost18.com>

crifan.com, 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: 2020-08-05 22:09:49

查壳工具

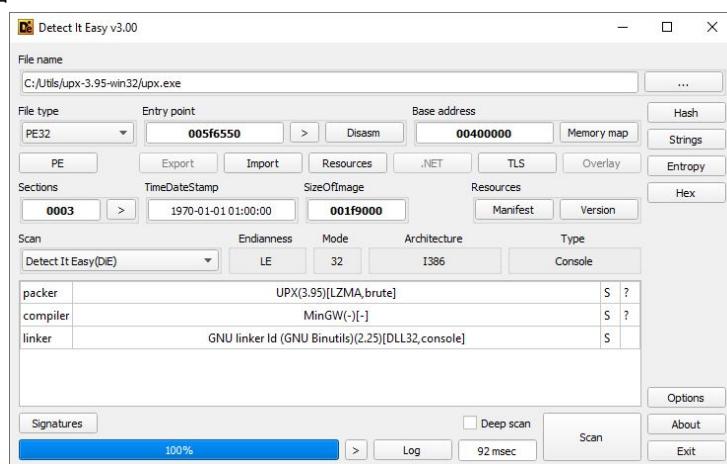
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2020-08-05 21:40:04

Detect It Easy

- Detect It Easy

- 简称: DIE

- 截图



- 资料

- GitHub

- horsicq/Detect-It-Easy: Program for determining types of files for Windows, Linux and MacOS.

- <https://github.com/horsicq/Detect-It-Easy>

- 官网

- ..:NTInfo:..

- <http://ntinfo.biz/index.html>

crifan.com, 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: 2020-08-05 22:10:44

PEiD

crifan.com, 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: 2020-08-05 21:46:49

逆向工具

crifan.com, 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: 2020-08-05 21:26:59

IDA

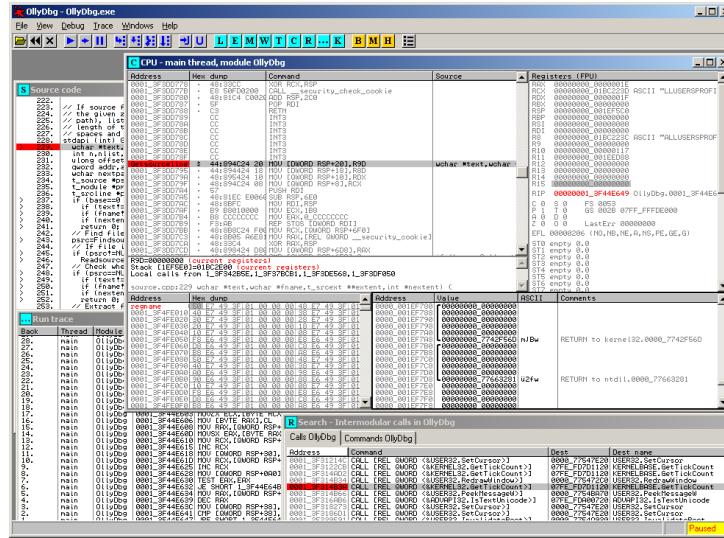
- IDA
 - = IDA Pro
 - 支持平台
 - Windows
 - Mac
 - Linux
 - 多个维度看
 - IDA Pro is a disassembler
 - A disassembler is a piece of software used to translate machine code into a human readable format called assembly language
 - IDA Pro is a debugger
 - A debugger is a computer program that assists in the detection and correction of errors in other computer programs
 - IDA Pro is interactive
 - IDA Pro is programmable
 - A plugin architecture allows a program to call external code at certain points without knowing all the details of that code in advance, therefore adding functionalities to the calling program
 - 用途
 - Hostile Code analysis
 - Vulnerability research
 - Commercial-off-the-shelf (COTS) validation
 - Privacy protection
 - Other uses
 - 评价
 - IDA是一个世界顶级的Ring3交互式反汇编工具
 - 是最强大的静态分析工具(也可动态调试)
 - 在分析整体程序流程、算法方面非常有优势
 - 应用举例
 - 游戏破解
 - 即使不能过掉反外挂系统，也可以用dump下游戏内存静态分析出游戏明文收发包等重要函数

crifan.com, 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: 2020-08-06 22:22:50

Ollydbg

TODO:

- 在看雪上有人翻译了国外人写的<使用ollydbg从零开始cracking>, 还是挺详细的介绍了od的使用, 推荐阅读和动手跟一遍。
 - 使用OllyDbg从零开始Cracking(已完结)-『外文翻译』 -看雪安全论坛
 - <https://bbs.pediy.com/thread-184679.htm>
 - Ollydbg
 - 一句话描述: OllyDbg is a 32-bit = x86 assembler level analysing debugger for Microsoft® Windows®. Emphasis on binary code analysis makes it particularly useful in cases where source is unavailable
 - 功能和特点
 - It traces registers, recognizes procedures, API calls, switches, tables, constants and strings, as well as locates routines from object files and libraries.
 - It has a user friendly interface, and its functionality can be extended by third-party plugins
 - 名字
 - ollie 来源于作者: Oleh Yuschuk
 - 评价
 - OllyDbg是windows平台下Ring 3级调试器
 - 界面友好
 - 非常容易上手
 - 支持插件扩展功能
 - 是当今最流行最强大的动态调试工具
 - 截图



- 应用举例
 - 游戏破解
 - 目前做辅助的基本均用此调试器分析需要的资源结构、内存数据，功能函数用于开发辅助功能

- 资料
 - 官网
 - v1
 - OllyDbg v1.10
 - <http://www.ollydbg.de>
 - v2
 - OllyDbg 2.0
 - <http://www.ollydbg.de/version2.html>

crifan.com, 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: 2020-08-08 22:04:54

WinDbg

- WinDbg
 - 评价
 - 是windows平台下强大的Ring3和Ring0调试工具
 - 应用举例
 - 在游戏漏洞挖掘中，主要用于调试分析反外挂驱动的保护点，以便写出反反外挂驱动
 -

crifan.com, 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：2020-08-05 22:17:14

内存修改工具

crifan.com, 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: 2020-08-05 22:22:36

CE

- CE
 - = Cheat Engine
 - 一句话简介：内存修改编辑工具
 - 说明
 - a development environment focused on modding games and applications for personal use
 - 资料
 - GitHub
 - cheat-engine/cheat-engine: Cheat Engine. A development environment focused on modding
 - <https://github.com/cheat-engine/cheat-engine>
 - 官网
 - Cheat Engine
 - <https://www.cheatengine.org>

crifan.com, 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：2020-08-05 22:43:27

MHS

- MHS

- = Memory Hacking Software

- 截图



- 主要功能

- Searching
 - View RAM In Real-Time
 - Debugging
 - Disassembling
 - Inject Code
 - Inject DLL's
 - Scripting
 - Hotkeys
 - Real-Time Expression Evaluator

- 资料

- 官网
 - L. Spiro's Memory Hacking Software
 - <http://memoryhacking.com>
 - 下载
 - <http://memoryhacking.com/download.php>
 -

ModifyMemory

crifan.com, 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: 2020-08-05 22:40:20

Mac安全

crifan.com, 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: 2020-08-05 19:59:14

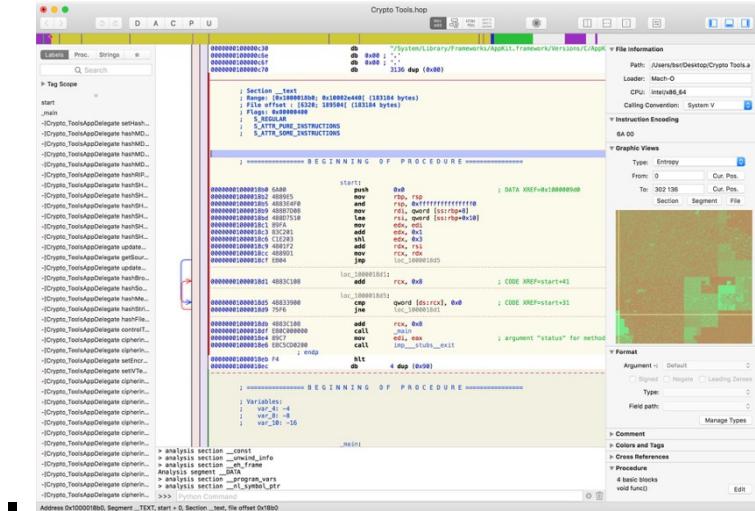
逆向工具

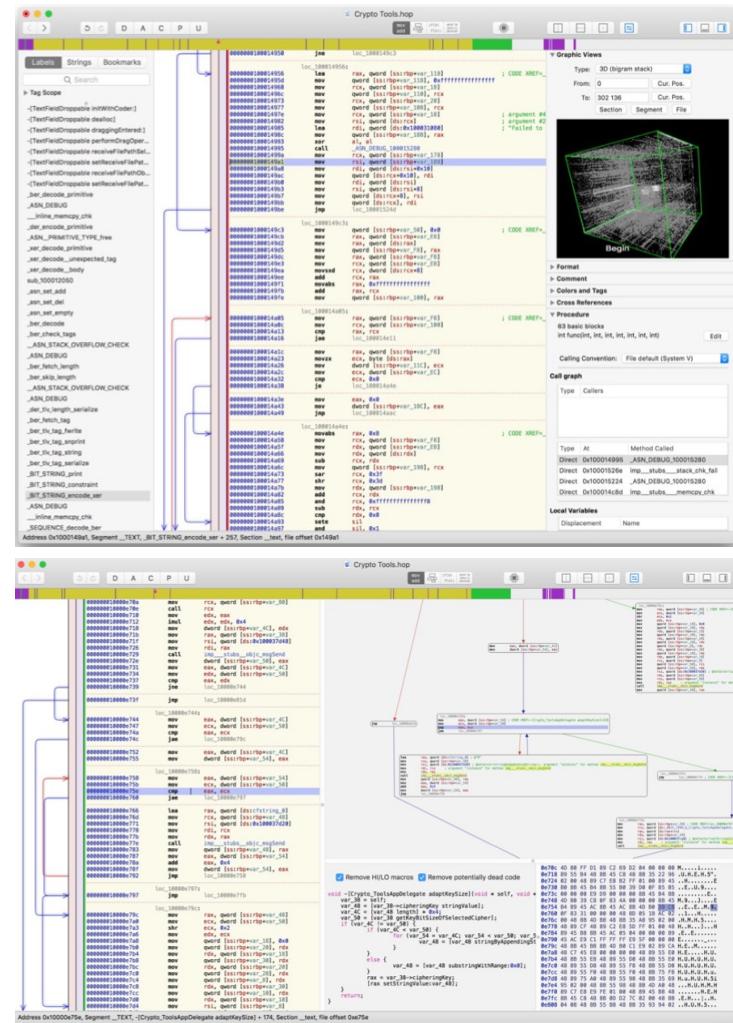
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2020-08-05 19:59:24

Hopper Disassembler

- Hopper Disassemble

- = 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

crifan.com, 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: 2020-08-05 20:02:21

编译套件

crifan.com, 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: 2020-08-10 09:07:45

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

crifan.com, 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: 2020-08-10 09:07:45

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 llbd 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: O LLVM
 - 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>

crifan.com, 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: 2020-08-10 09:07:45

调试器

crifan.com, 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: 2020-08-10 09:07:45

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>

crifan.com, 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：2020-08-10 09:07:45

逆向工具

crifan.com, 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: 2020-08-06 20:56:57

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/>

crifan.com, 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：2020-08-06 20:58:12

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

The screenshot shows the radare2 interface with two main panes. The top pane displays assembly code from memory address 0x100001206 to 0x10000120f. The bottom pane shows a detailed control flow graph (CFG) for the same code region, highlighting various branches and jumps.

```
[0x100001206]> pd 10
0x100001206 4156      push r14
0x100001208 4155      push r13
0x10000120a 4154      push r12
0x10000120c 53        push rbx
0x10000120d 4881ec180600. sub rsp, 0x618
0x100001214 4989f7    mov r15, rsi
0x100001217 4189fe    mov r14d, edi
0x10000121a 488d85c0fdff. lea rax, [rbp - 0x240]
0x100001221 488945d0    mov qword [rbp - 0x30], rax
0x100001225 4585f6    test r14d, r14d
[0x100001206]> pd 4 @..0d
0x10000120d 4881ec180600. sub rsp, 0x618
0x100001214 4989f7    mov r15, rsi
0x100001217 4189fe    mov r14d, edi
0x10000121a 488d85c0fdff. lea rax, [rbp - 0x240]
[0x100001206]> pd 4 @..225
0x100001225 4585f6    test r14d, r14d
0x100001228 7105      jg 0x10000122f
0x10000122a e8d1310000  call sym.func.100004400
0x10000122f 488d35b03800. lea rsi, 0x100004af0 ; section.4._TEXT._cstring
[0x100001206]>
```

Below the assembly code, the CFG diagram illustrates the control flow between the various instructions. Nodes represent specific assembly blocks, and edges represent conditional jumps (e.g., jg, je) and returns. The nodes are color-coded and numbered (e.g., 0x1000011a6, 0x1000011e3, etc.) to track the flow through the program's logic.

- 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
- rasim2 : 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 proj] [-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 [proj]   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.usestr=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

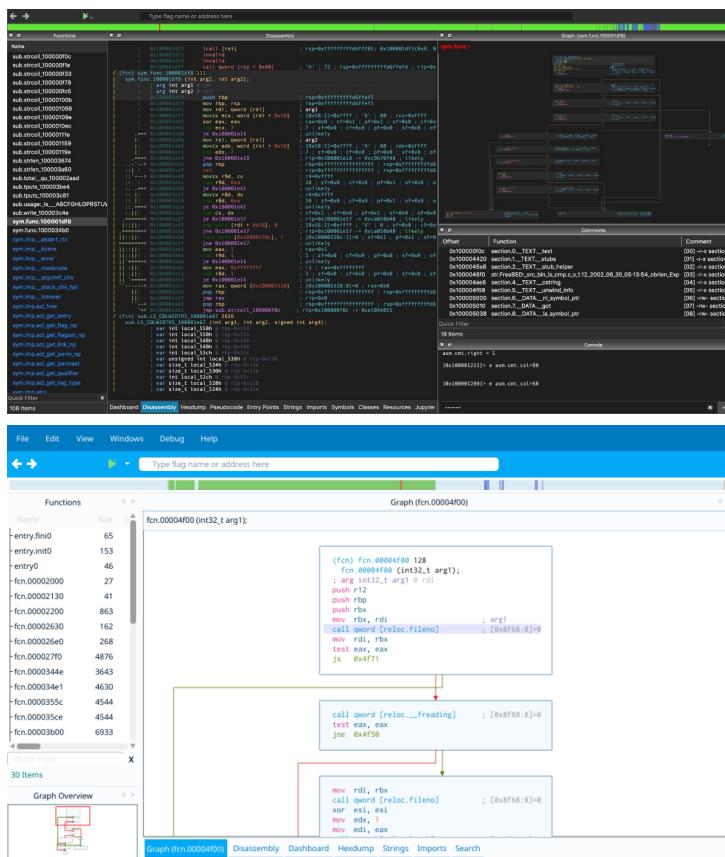
- Windows

- 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>

crifan.com, 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: 2020-08-06 22:08:58

移动安全

- 移动端：移动设备安全
 - Android
 - Apk逆向工具
 - Apktool
 - jd-gui
 - dex2jar
 - apk反编译
 - apk
 - 脱壳
 - 加壳
 - Smali/Baksmali代码
 - Android
 - Hook技术
 - Xposed
 - 虚拟化技术
 - VirtualApp
 - DroidPlugin
 - iOS

crifan.com, 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：2020-07-31 23:07:32

Android安全

详见独立教程：

[安卓应用的安全和破解](#)

crifan.com, 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：2020-08-06 11:21:02

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
 - Cycrypt
 - 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

crifan.com, 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：2020-08-04 21:15:23

正向工具

crifan.com, 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: 2020-08-04 21:04:41

混淆

crifan.com, 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: 2020-08-04 21:04:46

ios-class-guard

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

crifan.com, 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: 2020-08-04 21:09:29

常用工具

crifan.com, 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: 2020-08-04 19:58:23

调试器

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

crifan.com, 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: 2020-08-10 09:07:45

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"

crifan.com, 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： 2020-08-10 09:07:45

插件开发

crifan.com, 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: 2020-08-04 21:19:15

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、Cycrypt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>

crifan.com, 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：2020-08-04 21:21:35

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。

现在很多可以处理 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

crifan.com, 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: 2020-08-06 22:41:40

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://stevenyngard.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/XCTTestDriver.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

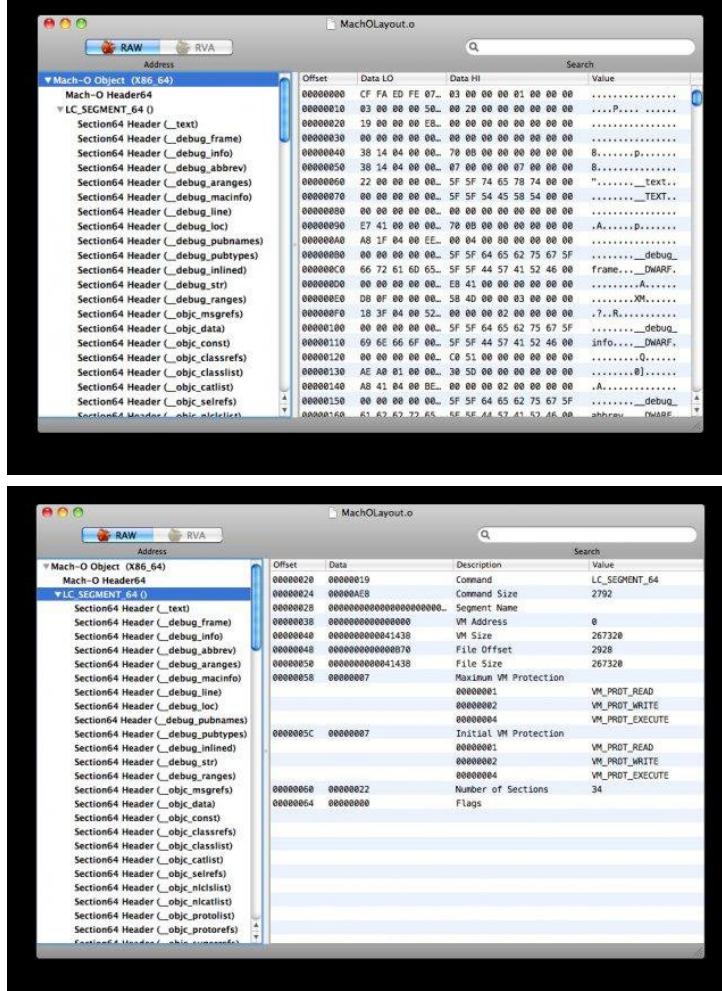
```
class-dump 3.5 (64 bit)  
Usage: class-dump [options] <mach-o-file>  
  
where options are:  
-a          show instance variable offsets  
-A          show implementation addresses  
--arch <arch> choose a specific architecture from a universal binary  
-C <regex>   only display classes matching regular expression  
-f <str>     find string in method name  
-H          generate header files in current directory, or director  
-I          sort classes, categories, and protocols by inheritance  
-o <dir>    output directory used for -H  
-r          recursively expand frameworks and fixed VM shared libra  
-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/Platfo  
--sdk-mac    specify Mac OS X version (will look in /Developer/SDKs/1  
--sdk-root   specify the full SDK root path (or use --sdk-ios/--sdk-
```

crifan.com, 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: 2020-08-04 20:48:25

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>

crifan.com, 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：2020-08-10 09:07:45

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>

crifan.com, 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: 2020-08-04 22:46:22

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)  
armeabi - 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 disassemble 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/
```

crifan.com, 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: 2020-08-04 22:49:27

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

crifan.com, 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：2020-08-04 20:40:26

bfinject

crifan.com, 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: 2020-08-04 20:16:09

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
```

crifan.com, 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：2020-08-04 20:25:13

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>

crifan.com, 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: 2020-08-04 20:04:50

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>

crifan.com, 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：2020-08-04 21:10:08

越狱工具

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

crifan.com, 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: 2020-08-04 22:01:38

Cydia Substrate

crifan.com, 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: 2020-08-04 21:23:43

frida

crifan.com, 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: 2020-08-04 21:11:21

Electra

crifan.com, 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: 2020-08-04 22:01:01

unc0ver

crifan.com, 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: 2020-08-04 22:01:14

工控物联网安全

工控安全，从概念上，基本上等价于最新出现的物联网安全

内容较多，且自成体系，现已整理至独立教程：

[工控安全概览](#)

crifan.com，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：2020-10-29 20:04:37

特定设备安全

crifan.com, 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: 2020-08-01 09:28:57

WiFi安全

- WiFi安全
 - Aircrack-ng
 - Wifiphisher
 -

crifan.com, 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: 2020-08-03 22:33:24

Aircrack-ng

- Aircrack-ng
 - 一句话描述：一种802.11 WEP和WPA-PSK密钥破解黑客工具
 - 可以在捕获到足够的数据包时恢复密码
 - 资料
 - 官网
 - <http://www.aircrack-ng.org>

crifan.com, 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：2020-08-03 22:32:14

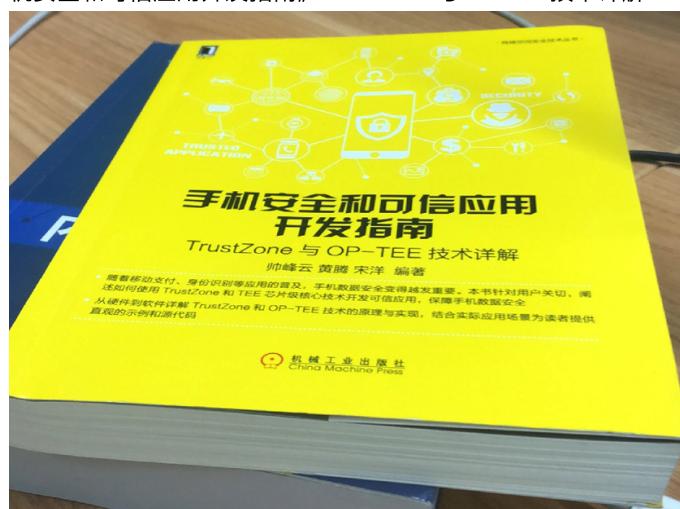
Wifiphisher

- Wifiphisher
 - 一句话描述：Wifiphisher是个伪造恶意接入点的工具，可针对WiFi网络发起自动化网络钓鱼攻击。
 - 于任务范围，Wifiphisher可致凭证获取或实际的感染

crifan.com, 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：2020-08-03 22:33:55

信息存储安全

- 信息存储安全
 - 应用场景和领域
 - 生物特征数据存储
 - 指纹
 - 虹膜
 - 信用卡PIN码（保存）
 - 私有密码（存储）
 - 客户数据（存储）
 - 受 DRM = Digital Rights Management = 数字版权管理 保护的媒体
 - 相关书籍
 - 《手机安全和可信应用开发指南》 TrustZone与OP-TEE技术详解



- 相关技术
 - 硬件层面
 - TrustZone
 - 软件层面
 - TEE

crifan.com, 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: 2020-08-04 22:57:47

硬件

crifan.com, 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: 2020-08-04 22:55:58

TrustZone

- TrustZone
 - ARM
 - 提出了 TrustZone 技术
 - 为了确保数据安全
 - 用一根 安全总线 (称为 NS 位) 来判断当前处于 secure world 还是 non-secure world 状态
 - 状态的切换由 ATF = ARM Trusted Firmware 来完成

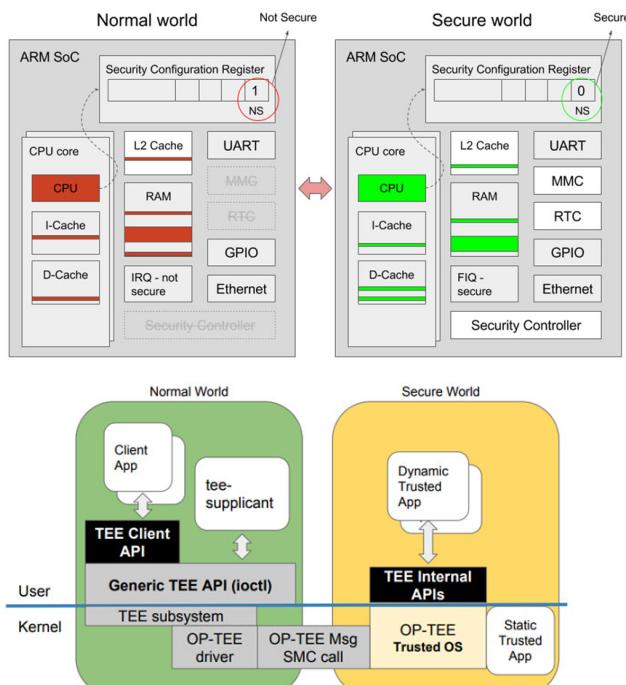
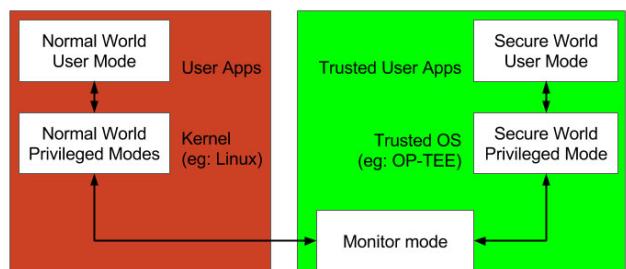
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2020-08-04 22:57:11

软件

crifan.com, 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: 2020-08-04 22:56:03

TEE

- TEE = OP-TEE
 - 名称
 - TEE = Trusted Execution Environment = 信任执行环境 = 可信任执行环境
 - OP-TEE = Open Portable Trusted Execution Environment = Open-Source Portable Trusted Execution Environment = 开放可移植的可信任执行环境
 - 一句话描述
 - 基于TrustZone技术搭建的安全执行环境
 - designed as companion to a non-secure Linux kernel running on Arm
 - 注: Cortex-A cores using the TrustZone technology
 - 用途=目的=为什么
 - 为了更安全
 - 处理那些需要和安全密切相关的、需要保密处理的信息
 - 历史
 - 最早是ST-Ericsson开发的
 - <http://www.stericsson.com/>
 - 2013年, ST-Ericsson实现了兼容 GlobalPlatform
 - <https://globalplatform.org/>
 - 2013年之后, ST和Ericsson分开了
 - 现在TEE属于STMicroelectronics
 - https://www.st.com/content/st_com/en.html
 - 2013年后期, Linaro成立了 SWG = Security Working Group = 安全工作组
 - 其最重要的任务之一就是继续开发TEE
 - 在开源TEE之前, 花了很多个月去把之前部分私有模块, 换成开源实现
 - 包括: 密码库, 安全监控, 编译系统及其他
 - 2014-06-12, TEE开源了, 叫做OP-TEE
 - 目前现状主要是:
 - 项目属于STMicroelectronics
 - 但是Linaro和STMicroelectronics联合在开发
 - 2015年, 项目所有权从STMicroelectronics转给Linaro了
- 资料
 - 官网
 - <https://www.op-tee.org>
 - GitHub
 - OP-TEE/optee_os: Trusted side of the TEE
 - https://github.com/OP-TEE/optee_os
 - 技术文档
 - OP-TEE Documentation — OP-TEE documentation documentation
 - <http://optee.readthedocs.io>
- 主要设计目标
 - Isolation

- the TEE provides isolation from the non-secure OS and protects the loaded Trusted Applications (TAs) from each other using underlying hardware support,
- Small footprint
 - the TEE should remain small enough to reside in a reasonable amount of on-chip memory as found on Arm based systems
- Portability
 - the TEE aims at being easily pluggable to different architectures and available HW and has to support various setups such as multiple client OSes or multiple TEEs.
- OP-TEE 包含内容
 - Secure world OS= optee_os
 - 现有实现:
 - OP-TEE OS , Trusty , 高通的 QSEE , SierraTEE
 - 注: 所有方案的外部接口都会遵循 GP = Global Platform 标准
 - 对比: Normal world os
 - 普通操作系统: Linux、Android等
 - 问: 各家厂商和组织的 TEE OS 到底有何区别?
 - 答: TA 的添加和加载时的校验有所区别
 - 系统架构
 - 
 - 相关概念
 - 

- TA = Trusted Application =可信应用
- CA = Client Application =客户端应用
- 原理
 - 产品开发团队负责开发一个运行在 Linux 上的 CA 和一个运行在 OP-TEE 上的 TA
 - CA 使用 TEE client API 与 TA 通信，并且从 TA 获取安全服务
 - CA 和 TA 使用 共享内存 进行通信
- 运行机制
 - 当处于 secure world 状态，那么就会执行 TEE OS 部分的代码
 - 当处于 non-secure world 状态时，就执行 linux kernel 部分的代码
- 举例

- 芯来科技和瓶钵信息合作开发基于RISC-V的TEE方案



- 其中也是符合TEE的架构
 - 可信区
 - 不可信区
- Normal world Client= optee_client
- test suite = optee_test/xtest
- linux驱动
- 常见问题
 - Linux内核
 - Linux内核能直接访问TEE部分的资源吗？
 - Linux kernel不能直接访问TEE部分的资源
 - Linux 内核如何才能访问TEE部分的资源呢？
 - Linux kernel能通过特定的 TA 和 CA 来访问TEE部分特定的资源

crifan.com, 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: 2020-10-29 20:32:16

其他安全相关

安全方面的公司

- Veracode
 - Veracode提供一个基于云的应用程序安全测试平台
 - 无需购买硬件，无需安装软件，使用客户马上就可以开始测试和补救应用程序，另外Veracode提供自动化的静态和动态应用程序安全测试软件和补救服务

crifan.com, 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: 2020-08-08 22:12:11

代码审计类

- 工具

- CxEnterprise
 - = Checkmarx CxEnterprise
- Armorize CodeSecure
- Fortify
 - = Fortify SCA
- RIPS

crifan.com, 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: 2020-08-08 22:12:37

Checkmarx CxEnterprise

- Checkmarx CxEnterprise
-

crifan.com, 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: 2020-08-08 22:13:38

Armorize CodeSecue

- Armorize CodeSecue
-

crifan.com, 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: 2020-08-08 22:14:38

Fortify

- Fortify
 -

crifan.com, 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: 2020-08-08 22:14:57

RIPS

- RIPS
 -

crifan.com, 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: 2020-08-08 22:15:15

相关技术和工具

crifan.com, 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: 2020-08-01 12:27:15

抓包工具

- 网络流量分析 = 网络报文监听 = 网络协议分析
 - Wireshark

crifan.com, 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: 2020-08-03 22:28:11

Wireshark

crifan.com, 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: 2020-08-03 22:28:24

破解密码

- John the Ripper
- Hashcat
- Hydra

crifan.com, 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: 2020-08-03 22:37:42

John the Ripper

- John the Ripper
 - 用GPU算力离线破解密码

crifan.com, 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: 2020-08-03 22:30:03

Hashcat

- Hashcat
 - 评价
 - 世界上最快、最先进的密码恢复实用程序
 - 破解哈希的首选渗透测试工具
 - 功能
 - 支持多种猜测密码的蛮力攻击
 - 包括字典和掩码攻击
 - 说明
 - Hashcat在现代GPU显卡上运行最好
 - 传统的hashcat仍支持CPU上的哈希破解
 - 但是要提醒用户的是，这比显卡的处理能力要慢得多

crifan.com, 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: 2020-08-03 22:37:17

Hydra

- Hydra
 - 可用于在线破解密码
 - 举例：SSH或FTP登录、IMAP、IRC、RDP等

crifan.com, 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：2020-08-03 22:37:45

代理工具

- 常见代理工具
 - Fiddler
 - 常用的抓包工具，有XSS自动化扫描插件
 - parosproxy
 - 一个对Web应用程序的漏洞进行评估的代理程序

crifan.com, 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: 2020-08-02 19:41:03

远程操作工具

- `scp`
 - 终端命令，把远程设备的文件复制到另一个设备

crifan.com, 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：2020-08-04 20:56:26

附录

下面列出相关参考资料。

crifan.com, 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: 2020-03-17 09:11:34

资料和文档

安全领域相关论坛

- 常见安全相关网站
 - tools
 - 简介
 - 十年民间网络安全老牌社区，聚合安全领域最优秀的人群，低调研究潜心学习讨论各类网络安全知识，为推动中国网络安全进步与技术创新贡献力量！
 - 当前国内为数不多的民间网络信息安全研究团队之一
 - wooyun=乌云
 - 最新：已关闭
 - 简介
 - 一个位于中国大陆的企业与安全研究人员（白帽子）之间的安全漏洞报告平台，并提供最新的研究资讯。
 - 2016年7月20日凌晨，乌云官网突然关闭，仅显示一张“升级通告”的图片，并附言“与其听信谣言，不如相信乌云”。据外界推测可能是内部整顿
 - 有多方消息表示多名乌云高管被警方带走，但同时也有人辟谣称是谣言。截至2020年3月，网站依然展示升级公告。
 - freebuf
 - 简介：国内领先的互联网安全新媒体，同时也是爱好者们交流与分享安全技术的社区。
 - 官网
 - FreeBuf互联网安全新媒体平台
 - <https://www.freebuf.com>
 - 安全客
 - 简介：安全客 - 安全资讯平台
 - 网站：<https://www.anquanke.com/>
 - Seebug
 - 简介：一个权威的漏洞参考、分享与学习的安全漏洞平台，是国内权威的漏洞库，在国内和国际都享有知名度，于2006年上线。
 - 官网
 - 知道创宇 Seebug 漏洞平台 - 洞悉漏洞，让你掌握第一手漏洞情报！
 - <https://www.seebug.org>
 - exploit-db.com
 - 简介：一个面向全世界黑客的漏洞提交平台
 - 官网
 - Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers
 - <https://www.exploit-db.com>
 - 吾爱破解
 - 简介：吾爱破解论坛致力于软件安全与病毒分析的前沿，丰富的技术版块交相辉映，由无数热衷于软件加密解密及反病毒爱好者共同维护

- 网站: <https://52pojie.cn>
- Paper(知道创宇)
 - 简介: 安全技术精粹
 - 网站: <https://paper.seebug.org/>
- CTFWIKI
 - 简介: CTF Wiki
 - 网站: <https://ctf-wiki.github.io/ctf-wiki/>
- CTFtime
 - 简介: Capture The Flag, CTF teams, CTF ratings, CTF archive, CTF writeups
 - 网站: <https://ctftime.org/>
- 先知社区
 - 简介: 先知社区, 先知安全技术社区
 - 网站: <https://xz.aliyun.com/>

crifan.com, 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: 2020-07-31 21:36:29

参考资料

- 【已解决】Mac中用otool查看IDA依赖的库
- 工控安全概览
-
- CTF.GS_CTF网站_CTF网址_CTF网址导航_CTF练习平台_CTF练习平台收集
- CTF大本营 - 网络安全竞赛服务平台-i春秋
- Hacker101 CTF
- CTFtime.org / All about CTF (Capture The Flag)
- optee开源项目的学习_fanguannan0706的专栏-CSDN博客_optee
- Open Portable Trusted Execution Environment - OP-TEE
- 什么是OPTEE-OS - 江召伟 - 博客园
- About OP-TEE — OP-TEE documentation documentation
- 【渗透测试工程师招聘】_暗泉信息招聘-BOSS直聘
- 漏洞利用 - 维基百科, 自由的百科全书
- 漏洞 - 维基百科, 自由的百科全书
- 计算机安全 - 维基百科, 自由的百科全书
- 网络安全 - 维基百科, 自由的百科全书
- 国内、国外网站安全渗透测试、漏洞扫描产品 | Venhow's Blog
- 渗透测试专业人员使用的11种工具 - FreeBuf互联网安全新媒体平台
- 谈谈我对逆向的一些认识 - 简书
- 「移动安全工程师招聘」_苏州极光无限信息...招聘-BOSS直聘
- 漏洞扫描原理——将主机扫描、端口扫描以及OS扫描、脆弱点扫描都统一放到了一起 - bonelee - 博客园
- 【知识科普】安全测试OWASP ZAP简介 - 知乎
- OWASP ZAP安全测试 - 简书
- 安全性测试：OWASP ZAP使用入门指南 - 哔哩哔哩
- Web安全测试-WebScarab工具介绍-云栖社区-阿里云
- 「网络安全」安全设备篇（防火墙-IDS-IPS） - 知乎
- 使用peach进行模糊测试从入门到放弃 - 安全客, 安全资讯平台
- Cain & Abel v4.9.44发布 - FreeBuf网络安全行业门户
- Layer子域名挖掘机5.0 SAINTSEC更新版 - 安全工具 - 互联网之家
- Layer子域名挖掘机 - guojia000 - 博客园
- 逆向分析之常见的汇编指令 - FreeBuf网络安全行业门户
- 十大黑客常用渗透测试工具 - 知乎
- 渗透测试专业人员使用的11种工具 - FreeBuf网络安全行业门户
- iOS App的加固保护原理 - 知乎
- iOS App 安全加固方案调研 - iOS - 掘金
- 对 iOS app 进行安全加固 - 我的学习历程
- iOS 应用加固方法 - 简书
- 为了保护公司的 App 安全, 我用遍了市面上的加固产品 - V2EX
- iOS逆向工程 介绍 | iOS 安全 Wiki
- iOS (十五) 几种App砸壳工具对比 ~ gandalf
- iOS逆向工具之砸壳工具(MacOS&iOS)介绍 - 简书

- iOS 攻防（四）：使用 Dumpdecrypted 砸壳 & class-dump 导出头文件 | 曹雪松de博客|CoderBoy's Blog
- iOS 攻防（六）：使用 Cycrypt 一窥运行程序的神秘面纱(入门篇) | 曹雪松de博客|CoderBoy's Blog
- class-dump 的安装和使用 - 简书
- iOS 攻防——（四）class-dump 与 Dumpdecrypted 使用 | 周小鱼の CODE_HOME
- class-dump 的安装和使用 - 简书
- class dump 使用方式和原理 - 简书
- Class-dump: class-dump-x 和 class-dump-z 如何分析 dylib 等文件？ - Discussion | 技术讨论 - iOSRE
- iOS 逆向之旅（进阶篇） — 工具(class-dump) - 掘金
- iOS 逆向之 class-dump - LeeGof - 博客园
- Tutorial · KJCracks/Clutch Wiki
- iOS 攻防 - （六）iOS 应用使用 Clutch 脱壳_移动开发_VictorZhang-CSDN 博客
- iOS 逆向工程之 Clutch 砸壳(图文多) - 简书
- iOS 逆向之 Clutch 砸壳 - 简书
- iOS（十五）几种 App 砸壳工具对比 ~ gandalf
- sqlcipher/sqlcipher: SQLCipher is an SQLite extension that provides 256 bit AES encryption of database files.
- iOS 安全攻与防(总篇) - 简书
- tianjifou/iOS-security-attack-and-prevent: iOS 安全攻与防, 详细的列出了, 在 iOS 开发中, 项目会存在的安全漏洞以及解决办法。
- 《iOS 应用逆向与安全》读后感 - 掘金
- iOS（十四）高版本越狱的坑 & killed 9 ~ gandalf
- OWASP Top Ten
- OWASP 10 大 Web 安全问题在 JEE 体系完全失控 - OneASP 技术分享 - SegmentFault 思否
- What is OWASP? What Are The OWASP Top 10? | Cloudflare
- Akamai 如何增强您的安全实践以缓解 OWASP 10 大风险
- vmeyet/owasp10: Open Web Application Security Project Top10 (2017) - Presentation with demos
- OWASP - Wikipedia
- 【从零开始学习 CTF】1、什么是 CTF - 知乎
- CTF.GS_CTF 网站_CTF 网址_CTF 网址导航_CTF 练习平台_CTF 练习平台收集
- CTF 大本营 - 网络安全竞赛服务平台-i春秋
- Hacker101 CTF
- CTFtime.org / All about CTF (Capture The Flag)
- iOS（十四）高版本越狱的坑 & killed 9 ~ gandalf
- macos - Editing assembly on Mac OS X - Stack Overflow
- iOS 逆向（八）逆向工具 otool 介绍 - 掘金
- otool 一些用途 - 简书
- otool 命令查看 App 动态库 - 简书
- iOS 逆向---otool 命令入门_嵌入式_ParadiseDuo-CSDN 博客
- iOS 程序逆向 Mac 下常用工具——Reveal、HopperDisassemble、IDA - 时间已静止 - 博客园
- Hopper Alternatives and Similar Software - AlternativeTo.net
- Detect-It-Easy: 一款跨平台的 PE 查壳工具 - 体验盒子 - 关注网络安全

- [Exeinfo PE 0.0.5.1 - 下载](#)
- [漏洞挖掘需要哪些基础知识? - 知乎](#)
- [零基础如何学习挖漏洞? - 知乎](#)
- [人工找漏洞是怎么找到的? 需要什么必要基础知识? - 知乎](#)
- [二进制漏洞挖掘从理论到实践按照顺序有哪些知识需要学习, 哪些书籍值得去读? - 知乎](#)
- [有Android逆向基础如何学习Android漏洞挖掘? - 知乎](#)
- [零基础, 如何进行漏洞挖掘 - 知乎](#)
- [从ctf入门漏洞挖掘_网络_tangsilian的博客-CSDN博客](#)
- [网站漏洞挖掘 - 云+社区 - 腾讯云](#)
- [\[思路/技术\] 如何入门漏洞挖掘, 以及提高自己的挖掘能力。 \(干货\) - CanMeng'Blog - 一个WEB安全渗透的技术爱好者](#)
- [谈高效漏洞挖掘之Fuzzing的艺术 - FreeBuf互联网安全新媒体平台](#)
- [\[求助\]怎样学习漏洞挖掘?-『经典问答』-看雪安全论坛](#)
- [\[原创\]各类漏洞挖掘方法辨析-『二进制漏洞』-看雪安全论坛](#)
- [\[原创\]游戏漏洞挖掘概述-『二进制漏洞』-看雪安全论坛](#)
- [mhs6.2汉化版下载|MHS\(内存修改工具\)下载v6.2 汉化 版_IT猫扑网](#)
- [\[转载\] 漏洞挖掘小白入坑指南 - 个人文章 - SegmentFault 思否](#)
- [漏洞挖掘 | 安全脉搏](#)
- [【技术分享】漏洞挖掘高级方法 - 安全客, 安全资讯平台](#)
- [内存搜索、修改器 \(附VC6源码\) BeanJoy的专栏-CSDN博客内存搜索修改器](#)
- [radare2 Alternatives and Similar Software - AlternativeTo.net](#)
- [Radare2 学习笔记: 从入门到精通 1. Radare2 简介, 及安装_Tangent's blog-CSDN博客_radare2 安装](#)
- [老司机带你玩转Radare2 - 简书](#)
- [Macho文件浏览器---MachOView - 简书](#)
- [一文带你快速理解CVE是什么意思? 从CVE ID分配到CVE漏洞处理](#)
- [提交CVE漏洞是一种怎样的体验? - 知乎](#)
- [聊聊CVE漏洞编号和正式公开那些事 | 技术博客](#)
- [通用漏洞披露 - 维基百科, 自由的百科全书](#)
- [通用漏洞披露 \(CVE\)](#)
- [国家信息安全漏洞库](#)
- [申请CVE的姿势总结 - FreeBuf互联网安全新媒体平台](#)
- [FreeBuf互联网安全新媒体平台](#)
- [全球最新漏洞库 - 安全客, 安全资讯平台](#)
- [逼格or随性?看我是如何混一波CVE编号! - FreeBuf专栏·OSPTECH攻防团队](#)
- [极光无限安全团队集结令 \(苏州\) - FreeBuf网络安全行业门户](#)
- [Microsoft Visual C++ - 维基百科, 自由的百科全书](#)
- [windows - How to disable buffer overflow checking in the Visual C++ Runtime? - Stack Overflow](#)
- [Intel® C++ Compiler 19.1 Developer Guide and Reference](#)
- [/GS \(Buffer Security Check\) | Microsoft Docs](#)
- [MSVC Compiler Options | Microsoft Docs](#)
- [GS buffer](#)
- [fstack-protector](#)
- [OllyDbg使用入门 | M0rk's Blog](#)
- [漏洞挖掘工程师](#)
- [strict_gs_check pragma | Microsoft Docs](#)

- [trailofbits/winchecksec: Checksec, but for Windows: static detection of security mitigations in executables](#)
- [c++ - How to check whether an EXE has /GS security protection on Windows? - Stack Overflow](#)
- [IMAGE_LOAD_CONFIG_DIRECTORY32 \(winnt.h\) - Win32 apps | Microsoft Docs](#)
- [GS cookie protection – effectiveness and limitations - Microsoft Security Response Center](#)
- [Security Features in MSVC | C++ Team Blog](#)
- [/SAFESEH \(Image has Safe Exception Handlers\) | Microsoft Docs](#)
- [MASM for x64 \(ml64.exe\) | Microsoft Docs](#)
- [On the effectiveness of DEP and ASLR - Microsoft Security Response Center](#)
- [exploit - How do ASLR and DEP work? - Information Security Stack Exchange](#)
- [DEP/ASLR 原理及攻击_运维_forevertingting的博客-CSDN博客](#)
- [windows - How to bypass DEP and ASLR at the same time? - Information Security Stack Exchange](#)
- [buffer overflow - How does SEH based exploit bypass DEP and ASLR? - Information Security Stack Exchange](#)
- [exploit - Stack Overflows - Defeating Canaries, ASLR, DEP, NX - Information Security Stack Exchange](#)
- [buffer overflow - Bypass Full ASLR+DEP exploit mitigation - Information Security Stack Exchange](#)
- [/DYNAMICBASE \(Use address space layout randomization\) | Microsoft Docs](#)
- [Whitepaper on Bypassing ASLR/DEP](#)
- [ASLR/DEP绕过技术概览 – ArkTeam](#)
- [Why it's important to turn on DEP and ASLR Windows security features](#)
- [安全保护技术ASLR绕过启示录 - FreeBuf互联网安新新媒体平台](#)
- [栈溢出基本ROP绕过ASLR和NX保护_网络_ditto的博客-CSDN博客](#)
- [“优雅”的Linux漏洞：用罕见方式绕过ASLR和DEP保护机制 - 云+社区 - 腾讯云](#)
- [Windows溢出保护原理与绕过方法概览 | riuskksk's blog](#)
- [Abysssec Information Security and Vulnerability Research Group](#)
- [IE漏洞学习笔记（一）Heap Spray - 安全客，安全资讯平台](#)
- [PWN入门系列（四）：栈终结篇 - 安全客，安全资讯平台](#)
- [pwn入门之栈溢出练习 - FreeBuf专栏·春秋学院](#)
- [Linux pwn入门教程\(1\)——栈溢出基础 - 知乎](#)
- [ctf中pwn入门指南 | ditto's blog](#)
- [\[原创\]GoParser——Yet Another Golang binary parser for IDAPro-『软件逆向』 -看雪安全论坛](#)
- [\[原创\]使用 mitmproxy 快速搭建软件网络验证API——以某音频处理软件为例-『软件逆向』 -看雪安全论坛](#)
- [神器如 dnSpy，无需源码也能修改 .NET 程序_walterlv - 吕毅-CSDN博客_dnspy](#)
- [dnSpy - 一款 .NET 程序逆向工具\]](#)
- [使用dnSpy对目标程序\(EXE或DLL\)进行反编译修改并编译运行 - jack_Meng - 博客园](#)
- [工具推荐：逆向破解利器OllyDbg - 知乎](#)
- [OllyDbg - Wikipedia](#)

- [Download OllyDbg 2.01](#)
- [七周年礼物第五弹之一：吾爱破解专用版Ollydbg 2016年1月21日更新](#)
- [OllyDbg Download \(2020 Latest\) for Windows 10, 8, 7](#)
- [OllyDbg使用入门 | M0rk's Blog](#)
- [极光无限安全团队集结令（苏州） - FreeBuf网络安全行业门户](#)
- [嵌入式设备漏洞研究员](#)
- [ReFirmLabs/binwalk: Firmware Analysis Tool](#)
- [Binwalk工具的详细使用说明运维子曰小玖的博客-CSDN博客](#)
- [binwalk windows安装和使用方法 - pcat - 博客园](#)
- [自动提取文件系统---binwalk\(一\) - blacksunny - 博客园](#)
- [Binwalk：固件分析利器 – ArkTeam](#)
- [Binwalk | Penetration Testing Tools](#)
- [BinWalk安装和命令参数详解 - 云+社区 - 腾讯云](#)
- [/NXCOMPAT \(Compatible with Data Execution Prevention\) | Microsoft Docs](#)
- [Control Flow Guard - Win32 apps | Microsoft Docs](#)
- [safebuffers | Microsoft Docs](#)
- [内存保护机制及绕过方案——通过覆盖虚函数表绕过/GS机制 - zhang293 - 博客园](#)
- [Windows安全机制---栈保护：GS机制_每昔的博客-CSDN博客_charshellcode](#)
- [绕过GS的栈溢出攻击原理 | Kumqu's Blog](#)
- [绕过GS安全编译的方法 | Introspelliam](#)
- [通关栈溢出（四）：缓冲区溢出的防御技术及绕过 - Hello! CytQ](#)
- [visual studio - safeseh gs on g++ - Stack Overflow](#)
- [SafeSEH利用（DEP/ASLR disabled） - 简书](#)
- [gdbinit/MachOView: MachOView fork](#)
- [The Ultimate Disassembly Framework – Capstone – The Ultimate Disassembler](#)
- [Capstone & LLVM – Capstone – The Ultimate Disassembler](#)
- [各种开源汇编、反汇编引擎的非专业比较 - simpower的个人空间 - OSCHINA](#)
- [Capstone引擎正式支持RISC-V架构 - 51CTO.COM](#)
- [KEYSTONE: Next Generation Assembler Framework](#)
- [OLLVM代码混淆移植与使用 | Heroims的博客](#)
- [利用ollvm进行代码混淆 | m4bln](#)
- [Low Level Virtual Machine \(LLVM\)](#)
- [LLDB \(debugger\) - Wikipedia](#)
- [LLDB Homepage — The LLDB Debugger](#)
- [Tutorial — The LLDB Debugger](#)
- [深入了解GDB和LLDB - 简书](#)
- [LLDB 知多少 - 掘金](#)
- [iOS（十六）一次通过lldb绕过越狱检测&反反调试实践 ~ gandalf](#)
- [debugserver - iPhone Development Wiki](#)
- [使用lldb+debugserver动态调试iOS应用 | La0s](#)
- [iOS 逆向指南：动态分析 – 小专栏](#)
- [iOS逆向, 基础工具之LLDB和debugserver - 简书](#)
- [一步一步用debugserver + lldb代替gdb进行动态调试 - Blog | 干货分享 - iOSRE](#)
- [Remote Debugging — The LLDB Debugger](#)
- [安全解决方案 - RISC-V IP方案](#)

•
crifan.com, 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: 2020-10-29 20:29:01