

目录

前言	1.1
IDA概览	1.2
IDA快手上手	1.3
IDA通用知识	1.4
版本	1.4.1
界面布局	1.4.2
命名和含义	1.4.3
自动分析	1.4.4
搜索	1.4.5
快捷键	1.4.6
IDA功能详解	1.5
查看代码	1.5.1
汇编代码	1.5.1.1
伪代码	1.5.1.2
函数调用	1.5.1.3
结构体类定义	1.5.1.4
字符串	1.5.2
函数列表	1.5.3
导入和导出	1.5.4
插件	1.5.5
IDA使用心得	1.6
附录	1.7
文档和资料	1.7.1
参考资料	1.7.2

逆向利器：IDA

- 最新版本： v0.7
- 更新时间： 20221024

简介

介绍逆向领域中功能强大且好用的利器：IDA。先介绍IDA概览；再介绍IDA的快速上手过程；再介绍IDA中通用的基础知识，包括版本选择、界面相关比如布局等、常见命名和含义、自动分析过程、搜索、快捷键；以及介绍IDA各种功能，包括查看代码，比如汇编代码、F5伪代码、函数调用、结构体的类的定义、字符串、函数列表插件等等；再去记录IDA的使用心得；最后整理一些IDA相关的文档和资料，供参考。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- [crifan/reverse_tool_ida: 逆向利器：IDA](#)

如何使用此HonKit源码去生成发布为电子书

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

在线浏览

- [逆向利器：IDA book.crifan.org](#)
- [逆向利器：IDA crifan.github.io](#)

离线下载阅读

- [逆向利器：IDA PDF](#)
- [逆向利器：IDA ePUB](#)
- [逆向利器：IDA Mobi](#)

版权和用途说明

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

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

鸣谢

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

更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

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

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新：2022-10-24 12:16:44

IDA概览

在逆向领域，有款很功能强大且好用的工具=利器是：[IDA Pro](#)

- [IDA Pro](#)
 - 常简称：[IDA](#)
 - 常用于
 - iOS逆向
 - 静态分析：逆向二进制，研究代码逻辑
 - 常用功能：函数、F5伪代码、字符串、类的结构体定义等等
 - 动态调试：调试iOS的app
 - 主页
 - <https://hex-rays.com/>

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-24 10:58:01

IDA快手上手

- iOS逆向
 - 常用：（Mac中）用IDA分析代码
 - 下载和安装IDA
 - 找到要分析的iOS的app的二进制文件
 - 拖动二进制到IDA中
 - 等待分析完毕
 - 利用各种功能，研究代码逻辑
 - 常用功能
 - 字符串
 - 搜索感兴趣的关键字
 - 比如：越狱对应的单词： jailbreak 、 jail 、 jb 等
 - 函数
 - 搜索已找到的iOS的ObjC的类和函数
 - 然后打开查看代码逻辑
 - 伪代码
 - 当打开函数后，默认是汇编代码，按 F5 即可打开 伪代码
 - 近似于自己写的源码，人类可读的那种，就可以分析代码，搞懂函数的基本（甚至全部的）逻辑了
 - 结构体：类的定义
 - 结构体定义的来源
 - 有些是自动分析出来的
 - 如果没有错误，无需调整
 - 有些需要自己额外新增类的定义
 - 效果
 - 如果类的结构体定义是正确的话，那么伪代码中，自动会解析出，类的函数和属性的调用，即可大大增加伪代码的可读性
 - 之后的重点
 - IDA的静态分析：IDA 的 F5伪代码，查看伪代码的大概逻辑
 - Xcode的动态调试：iOS逆向的 Xcode+ + MonkeyDev + LLDB 的动态调试，通过hook等手段，确认调用了哪些函数，参数值如何等等
 - 互相配合：把Xcode中的实时的汇编代码 和 IDA中伪代码，互相对应起来，便于理解代码逻辑
 - 后续的优化代码逻辑：在逐渐搞懂更多代码逻辑后，继续给IDA中的 伪代码 去优化代码，主要是给参数、变量、函数等改名，以及更进一步的，新增或修改类的结构体定义，使得伪代码中自动解析出正确的类的函数和属性的调用等内容。
 - 偶尔用：用IDA调试二进制

下载

[IDA官网](#)有试用版可供下载：

[Download center \(hex-rays.com\)](#)

->

- [IDA Free](#)
- [IDA Evaluation](#)

IDA通用知识

TODO:

【整理】IDA中一些功能和选项设置

此处整理IDA中的基础的通用的知识。

- 版本
- 界面
 - 功能布局
 - 显示相关
- 命名和含义
- 自动分析
 - 基本流程
 - 进度
- 搜索
- 快捷键

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:42:46

版本

IDA软件的不同版本

常用结论：此处iOS逆向的话，常用的是 v7.0 的 IDA Pro (中64位的 id64)

- 概述

- IDA Teams : Collaborative reverse-engineering work
 - 团队合作时才需要
- IDA Pro : The state-of-the-art binary code analysis tool
 - 常简称为 IDA -> 大家常说的IDA，指的是IDA Pro
 - 功能最全，最强大，但价格也最贵
- IDA Home : Affordable tool for reverse engineering hobbyists
 - 功能相对没 IDA Pro 全，但对于个人（应该）基本够用，价格相对也便宜点（所以宣传是Affordable）
- IDA Free : Free binary code analysis tool to evaluate IDA's basic functionalities
 - 免费，但功能有限
- IDA Demo : 仅仅作演示用，了解功能而已

- 详解

◦

IDA的历史版本号

[IDA updates and releases](#)

- IDA 8.x
 - IDA 8.1.221006 October 6, 2022
 - IDA 8.0.220829 (SP1) August 29, 2022
 - IDA 8.0.220729 July 29, 2022
- IDA 7.x
 - IDA 7.7.220118 (SP1) January 18, 2022
 - IDA 7.7.211224 December 24, 2021
 - IDA 7.6.210427 (SP1) April 28, 2021
 - IDA 7.6.210322 March 22, 2021

- IDA 7.5.201028 (SP3) October 28, 2020
- IDA 7.5.200728 (SP2) July 28, 2020
- IDA 7.5.200619 (SP1) June 19, 2020
- IDA 7.5.200519 May 19, 2020
- IDA 7.4.191112 (SP1) November 12, 2019
- IDA 7.4.191011 October 11, 2019
- IDA 7.3.190614 June 14, 2019
- IDA 7.2.181105 November 5, 2018
- IDA 7.1.180227 February 27, 2018
- IDA 7.0.171130 (SP1) November 30, 2017
- IDA 7.0.190914 September 14, 2017
- IDA 6.x
 - IDA 6.95.160808 August 08, 2016
 - IDA 6.9.151221 December 21, 2015
 - IDA 6.8.150413 April 13, 2015
 - IDA 6.7.141229 December 29, 2014
 - IDA 6.6.140604 June 04, 2014
 - IDA 6.5.131217 December 17, 2013
 - IDA 6.4.130306 March 6, 2013
 - IDA 6.4 January 10, 2013
 - IDA 6.3 May 31, 2012
 - IDA 6.2 October 05, 2011
 - IDA 6.1 April 08, 2011
 - IDA 6.0 October 01, 2010
- IDA 5.x
 - IDA 5.7 June 25, 2010
 - IDA 5.6 December 30, 2009
 - IDA 5.5 June 12, 2009
 - IDA 5.4 January 29, 2009
 - IDA 5.3 July 14, 2008
 - IDA 5.2 November 20, 2007
 - IDA 5.1 February 21, 2007
 - IDA 5.0 March 23, 2006
- IDA 4.x
 - IDA 4.9(SP) January 27, 2006
 - IDA 4.9 September 25, 2005
 - IDA 4.8 March 15, 2005
 - IDA 4.7 August 2004
 - IDA 4.6 October 27, 2003
 - IDA 4.x August 29, 2022
- IDA 3.x
 - IDA 3.x

界面布局

TODO:

- 【整理】IDA使用心得：多种显示模式
- 【已解决】IDA中浮动窗口Output Window如何固定到底部

此处整理，IDA中关于界面显示和布局方面的内容。

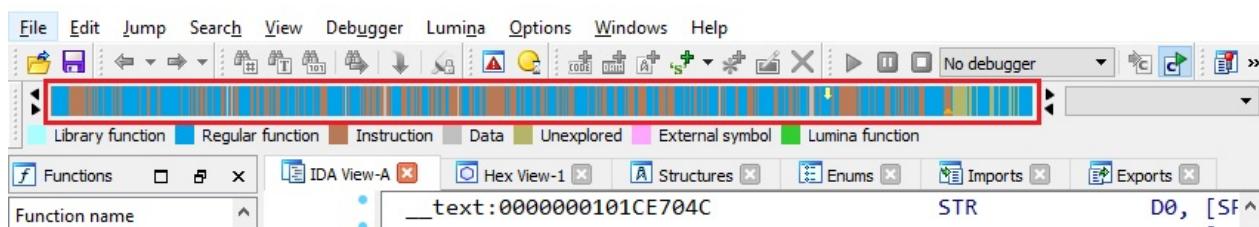
导航条=navigator

关于这个：

Navigation band = navigator = navbar = 导航栏 = 导航条

有专门的介绍

[Igor's tip of the week #49: Navigation band – Hex Rays \(hex-rays.com\)](#)



有空可以好好学习看看

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2022-10-23 22:19:30

命名和含义

TODO:

- 【整理】IDA使用心得：常见名称及含义
 - 【未解决】搞懂IDA中_D_objc_selrefs qword_38AF870 % 8的含义
-

此处整理IDA中，各处看到的，各种名称的命令规则的含义。

命名规则

IDA经常会自动生成假名字。他们用于表示子函数，程序地址和数据。根据不同的类型和值假名字有不同前缀

- IDA常见命名意义
 - sub 指令和子函数起点
 - locret 返回指令
 - loc 指令
 - off 数据，包含偏移量
 - seg 数据，包含段地址值
 - asc 数据，ASCII字符串
 - byte 数据，字节（或字节数组）
 - word 数据，16位数据（或字数组）
 - dword 数据，32位数据（或双字数组）
 - qword 数据，64位数据（或4字数组）
 - flt 浮点数据，32位（或浮点数组）
 - dbl 浮点数，64位（或双精度数组）
 - tbyte 浮点数，80位（或扩展精度浮点数）
 - stru 结构体(或结构体数组)
 - algn 对齐指示
 - unk 未处理字节
 - 字节相关
 - db=1个字节
 - dw=2个字节
 - dd=4个字节

举例

- sub
 - sub_11326A84

IDA - /Users/crifan/dev/DevRoot .framework/i .i64

Library function Regular function Instruction Data Unexplored External symbol

Functions window

```

Function name
Sub_6AB7BDC
Sub_6AB7BE4
Sub_6AB7BEC
Sub_6AB7C00
Sub_6AB7C84
Sub_6AB7D08
Sub_6AB7D50
Sub_6AB7D84
Sub_6AB7E14
Sub_6AB7F80
Sub_6AB7FC4
Sub_6AB8060
Sub_6AB806C
Sub_6AB8850
Sub_6AB88CC
Sub_6AB88D8
Sub_6AB88EC
Sub_6AB88F8
Sub_6AB890C
Sub_6AB8920
Sub_6AB892C
Sub_6AB8A7C
Sub_6AB8B18
Sub_6AB8B3C
Sub_6AB88EC
jmp_objc_autoreleasePoolPop_6AB8C10
call_anotherBlockInvoke_6AB8C18
    .eh_RARR`R

Line 150168 of 1872039
0C986A84 sub_11326A84:1(11326A84)

```

Output window

```

11470741: using guessed type _int64 __fastcall objc_alloc(_QWORD);
11470740: using guessed type _int64 __fastcall objc_autoreleasePoolPop(_QWORD);
11470854: using guessed type _int64 __fastcall objc_enumerationMutation(_QWORD);
114705C1: using guessed type _int64 __fastcall objc_retainAutorelease(_QWORD);

```

Python

AU: idle Down Disk: 243GB

- unk

- unk_5922000

IDA - /Users/crifan/dev/DevRoot .i64

Library function Regular function Instruction Data Unexplored External symbol

Functions window

```

Function name
Sub_6AB7BDC
Sub_6AB7BE4
Sub_6AB7BEC
Sub_6AB7C00
Sub_6AB7C84
Sub_6AB7D08
Sub_6AB7D50
Sub_6AB7D84
Sub_6AB7E14
Sub_6AB7F80
Sub_6AB7FC4
Sub_6AB8060
Sub_6AB806C
Sub_6AB8850
Sub_6AB88CC
Sub_6AB88D8
Sub_6AB88EC
Sub_6AB88F8
Sub_6AB890C
Sub_6AB8920
Sub_6AB892C
Sub_6AB8A7C
Sub_6AB8B18
Sub_6AB8B3C
Sub_6AB88EC
jmp_objc_autoreleasePoolPop_6AB8C10
call_anotherBlockInvoke_6AB8C18
    .eh_RARR`R

Line 150168 of 1872039
0C986C84 sub_11326A84:185(11326C84)

```

Output window

```

11470741: using guessed type _int64 __fastcall objc_alloc(_QWORD);
11470740: using guessed type _int64 __fastcall objc_autoreleasePoolPop(_QWORD);
11470854: using guessed type _int64 __fastcall objc_enumerationMutation(_QWORD);
114705C1: using guessed type _int64 __fastcall objc_retainAutorelease(_QWORD);

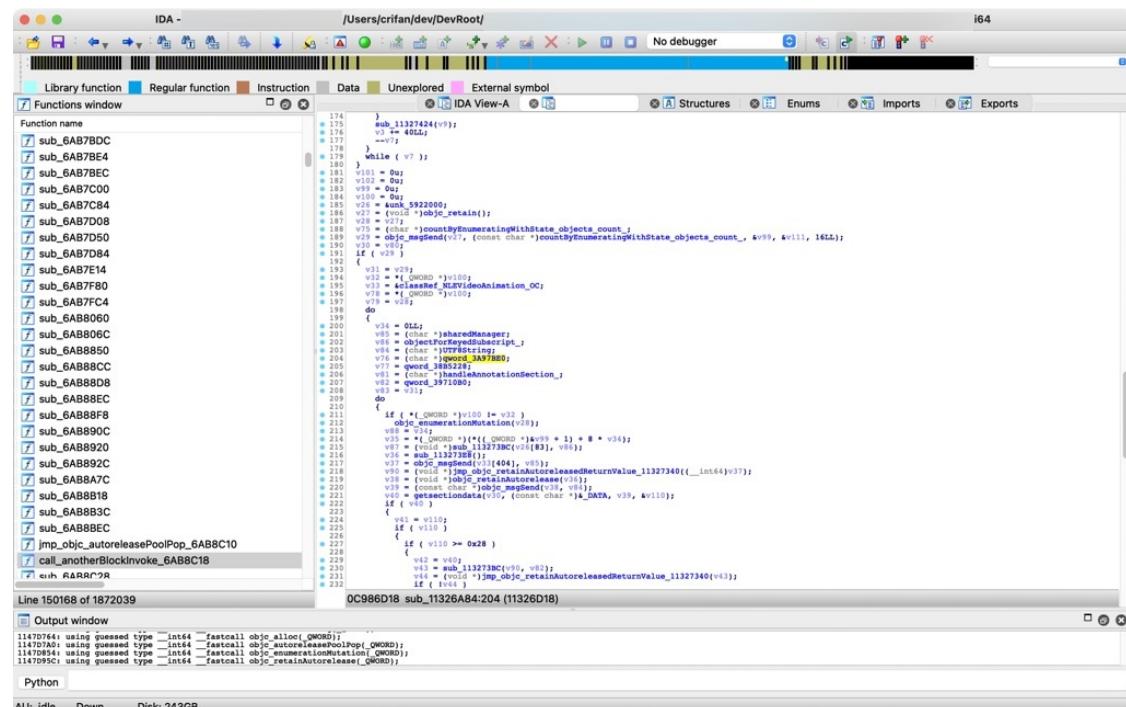
```

Python

AU: idle Down Disk: 243GB

- qword

- qword_3A97BE0



sub函数

关于sub函数的一些细节说明：

- `sub_xxx` : 普通的函数（有代码处理逻辑的）
 - 比如去改名的话，可以改名叫做：
 - `sub_BinaryOffset`
 - `sub_AddressInsideBinary`
- `nullsub_xxx` : 空函数（没有任何代码逻辑的）

此处给出实例：

举例：

【未解决】研究抖音越狱检测逻辑：`_RxAnnotationInlineLoader`的`load`

中的：

```
void __cdecl [_RxAnnotationInlineLoader load](_RxAnnotationInlineLoader_meta *self, SEL a2)
{
  ...
  j__dyld_register_func_for_add_image((void (__cdecl *)(const struct mach_header *, intptr_t))sub_11326A84);
  j__dyld_register_func_for_remove_image((void (__cdecl *)(const struct mach_header *, intptr_t))nullsub_12445);
}
```

IDA - AwemeCore.i64 (AwemeCore) /Users/crifan/dev/DevRoot/.i64

Regular function Instruction Data Unexplored External symbol

```

1 void __cdecl +[_RxAnnotationInlineLoader load](_RxAnnotationInlineLoader_meta *self, SEL a2)
2 {
3     uint32_t v2; // w0
4     uint32_t v3; // w19
5     uint32_t v4; // w20
6     const struct mach_header *v5; // x21
7
8     if ( !__OSAtomicTestAndSetBarrier(lu, &unk_5922290) )
9     {
10        qword 5922298 = ( int64 )jmp_objc_msgSend_D523EEC(&OBJC_CLASS__NSMutableDictionary, (const char *)new);
11        jmp_5922298 = release_D523EEC;
12        j_dyld_register_func_for_add_image((void __cdecl *)(const struct mach_header *, intptr_t)sub_11326A84);
13        j_dyld_register_func_for_remove_image((void __cdecl *)(const struct mach_header *, intptr_t))nullsub_12445;
14        v2 = j_dyld_image_count_0();
15
16        if ( ( v2 ) )
17        {
18            v3 = v2;
19            v4 = 0;
20            do
21            {
22                v5 = j_dyld_get_image_header(v4);
23                j_dyld_get_image_vmaddr_slide(v4);
24                sub_D523FD8((const struct mach_header_64 *)v5);
25            }
26            while ( v3 != v4 );
27        }
28    }
29    _ret_113273F0();
30 }
```

- `_dyld_register_func_for_add_image`传入的函数: `sub_11326A84`
 - 就是个普通的, 内部有代码逻辑的函数:

IDA - AwemeCore.i64 (AwemeCore) /Users/crifan/dev/DevRoot/.i64

weme.app/Frameworks/AwemeCore.framework/AwemeCore.i64

No debugger

Regular function Instruction Data Unexplored External symbol

```

1 int64 _fastcall sub_11326A84(const struct mach_header_64 *a1)
2 {
3     const struct mach_header_64 *v1; // x19
4     uint8_t *v2; // x0
5     uint8_t *v3; // x18
6     const char *v4; // x21
7     const char *v5; // x28
8     objc_class *v6; // x33
9     unsigned int64 v7; // x22
10    __int64 v8; // x0
11    __int64 v9; // x27
12    void *v10; // x27
13    __int64 v11; // x0
14    void *v12; // x0
15    void *v13; // x0
16    void *v14; // x25
17    void *v15; // x0
18    __int64 v16; // x0
19    __int64 v17; // x0
20    void *v18; // x28
21    const char *v19; // x23
22    void *v20; // x0
23    __int64 v21; // x0
24    __int64 v22; // x0
25    void *v23; // x0
26    void *v24; // x0
27    void *v25; // x24
28    void *v26; // x22
29    void *v27; // x0
30    void *v28; // x24
31    void *v29; // x0
32    const struct mach_header_64 *v30; // x20
33    void *v31; // x20
34    __int64 v32; // x26
35    objc_class ***v33; // x28
36    Signed int64 v34; // x19
37    __int64 v35; // x19
38    __int64 v36; // x19
39    void *v37; // x0
40    void *v38; // x0
41    const char *v39; // x0
42    uint8_t *v40; // x0
43    uint8_t *v41; // x21
44    uint8_t *v42; // x21
45    __int64 v43; // x0
46    void *v44; // x20
47    __int64 v45; // x0
48    __int64 v46; // x19
49    const char *v47; // x28
50    const char *v48; // x22
51    const char *v49; // x21
52    __int64 v50; // x23
53    void *v51; // x0
54    void *v52; // x0
55    __int64 v53; // x0
56    _QWORD v54; // x8
57    __int64 v55; // x19
58    void *v56; // x0
59    void *v57; // x19
```

0C986A84 sub_11326A84:11 (11326A84)

- `_dyld_register_func_for_remove_image`传入的函数: `nullsub_12445`
 - 从名字看, 就知道: 是个null的空的函数
 - 进入看, 果然是空的, 啥也没有

IDA - AwemeCore.i64 (AwemeCore) /Users/crifan/dev/DevRoot/.i64

Aweme.app/Frameworks/AwemeCore.framework/AwemeCore.i64

No debugger

Regular function Instruction Data Unexplored External symbol

```

1 void nullsub_12445()
2 {
3 }
4 }
```

具体含义

qword

对于qword:

- 常常是: 常量字符串

- 偶尔是：其他类型
 - 比如字典的指针等等

详见：

- 【已解决】IDA中抖音AwemeCore中字符串const char* qword_3893908的原始字符串
- 【已解决】iOS逆向心得：如何从对x8的adrp和ldr计算出对应的qword字符串值
 - 核心逻辑是：
 - qword_xxx的xxx是二进制内偏移量 + 二进制的ALSR = 实际（字符串的）地址
 - 去查看：[实际（字符串的）地址] = （即可查看到）保存了对应的字符串
- 【整理】iOS逆向心得：IDA中的unk的含义

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-24 10:11:56

自动分析

TODO:

- 【整理】IDA中的自动分析Autoanalysis
 - 【记录】用IDA分析加了符号表的抖音AwemeCore二进制
-

一般是用IDA去分析二进制中代码的逻辑。而二进制本身其实只有 0 和 1 二进制数据而已。

想要分析代码，即查看对应二进制对应的 汇编代码（以及后续的 伪代码），所包含的函数，所包含的字符串等等信息，则就需要：

对二进制进行充分的分析，最后才能显示出我们要的上述的各种信息。

而对于二进制加载后的分析过程，IDA叫做：

- 自动分析 = auto analysis

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2022-10-23 22:00:38

搜索

TODO:

- 【整理】IDA使用心得：search搜索查找
-

IDA中的搜索，可以用于各种地方，包括函数列表，字符串列表，全局搜索，等等。

其中和搜索相关，有些通用的逻辑，此处解释一下。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-23 22:06:14

快捷键

此处整理IDA的快捷键：

IDA_Pro_Shortcuts.pdf (hex-rays.com)

由于： 快捷键 = Shortcut = cheatsheet

所以此处是: IDA Pro Cheatsheet

IDAPro 7.5 – document updated February 09, 2021

File Operations		Edit (Data Types – etc)		Functions																																																																																																																																																																																	
Parse C header file...	Ctrl+F9	Rename	N	Create function...	P																																																																																																																																																																																
Create ASM file...	Alt+F10	Enter repeatable comment...	;	Edit function...	Alt+P																																																																																																																																																																																
Save	Ctrl+W	Enter comment...	:	Set function end	E																																																																																																																																																																																
Exit with Save	Alt+X or Alt+F4	Begin selection	Alt+L	Stack variables...	Ctrl+K																																																																																																																																																																																
Navigation		Code	C	Change stack pointer...	Alt+K																																																																																																																																																																																
Jump to operand	Enter	Data	D	Rename register...	V																																																																																																																																																																																
Jump in a new window	Alt+Enter	Struct var...	Alt+Q	Set type...	Y																																																																																																																																																																																
Jump to previous position	Esc	String	A	Lumina																																																																																																																																																																																	
Jump to next position	Ctrl+Enter	Array...	Numpad*, *	Jump to address...	G	Undefine	U	Pull all metadata	F12	Jump by name...	Ctrl+L	Enter anterior lines...	Ins	Push all metadata	Ctrl+F12	Jump to function...	Ctrl+P	Enter posterior lines...	Shift+Ins	View all metadata	Alt+F12	Jump to pseudocode	Tab	Offset (data segment)	O	Debugger		Jump to segment...	Ctrl+S	Offset (current segment)	Ctrl+O	Jump to segment register...	Ctrl+G	Offset by (any segment)...	Alt+R	Add breakpoint	F2	Jump to problem...	Ctrl+Q	Offset (user-defined)...	Ctrl+R	Start process	F9	List cross references to...	Ctrl+X	Offset (struct)...	T	Terminate process	Ctrl+F2	Jump to xref to operand...	X	Number (default)	#	Step into	F7	Jump to entry point...	Ctrl+E	Hexadecimal	Q	Step over	F8	Mark position...	Alt+M	Decimal	H	Run until return	Ctrl+F7	Jump to marked position...	Ctrl+M	Binary	B	Run to cursor	F4	Error operand	Ctrl+F	Character	R	Breakpoint list	Ctrl+Alt+B	Search		Segment	S	Stack trace	Ctrl+Alt+S	Next code	Alt+C	Enum member...	M	Dialog Boxes		Next data	Ctrl+D	Stack variable	K	Next explored	Ctrl+A	Change sign	–	Navigate	Tab, Shift+Tab	Next unexplored	Ctrl+U	Bitwise negate	~	Immediate value...	Alt+I	String literals...	Alt+A	Toggle	Space	Next immediate value	Ctrl+I	Setup data types...	Alt+D	Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter	Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12		
Jump to address...	G	Undefine	U	Pull all metadata	F12																																																																																																																																																																																
Jump by name...	Ctrl+L	Enter anterior lines...	Ins	Push all metadata	Ctrl+F12																																																																																																																																																																																
Jump to function...	Ctrl+P	Enter posterior lines...	Shift+Ins	View all metadata	Alt+F12																																																																																																																																																																																
Jump to pseudocode	Tab	Offset (data segment)	O	Debugger																																																																																																																																																																																	
Jump to segment...	Ctrl+S	Offset (current segment)	Ctrl+O	Jump to segment register...	Ctrl+G	Offset by (any segment)...	Alt+R	Add breakpoint	F2	Jump to problem...	Ctrl+Q	Offset (user-defined)...	Ctrl+R	Start process	F9	List cross references to...	Ctrl+X	Offset (struct)...	T	Terminate process	Ctrl+F2	Jump to xref to operand...	X	Number (default)	#	Step into	F7	Jump to entry point...	Ctrl+E	Hexadecimal	Q	Step over	F8	Mark position...	Alt+M	Decimal	H	Run until return	Ctrl+F7	Jump to marked position...	Ctrl+M	Binary	B	Run to cursor	F4	Error operand	Ctrl+F	Character	R	Breakpoint list	Ctrl+Alt+B	Search		Segment	S	Stack trace	Ctrl+Alt+S	Next code	Alt+C	Enum member...	M	Dialog Boxes		Next data	Ctrl+D	Stack variable	K	Next explored	Ctrl+A	Change sign	–	Navigate	Tab, Shift+Tab	Next unexplored	Ctrl+U	Bitwise negate	~	Immediate value...	Alt+I	String literals...	Alt+A	Toggle	Space	Next immediate value	Ctrl+I	Setup data types...	Alt+D	Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter	Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																														
Jump to segment register...	Ctrl+G	Offset by (any segment)...	Alt+R	Add breakpoint	F2																																																																																																																																																																																
Jump to problem...	Ctrl+Q	Offset (user-defined)...	Ctrl+R	Start process	F9																																																																																																																																																																																
List cross references to...	Ctrl+X	Offset (struct)...	T	Terminate process	Ctrl+F2																																																																																																																																																																																
Jump to xref to operand...	X	Number (default)	#	Step into	F7																																																																																																																																																																																
Jump to entry point...	Ctrl+E	Hexadecimal	Q	Step over	F8																																																																																																																																																																																
Mark position...	Alt+M	Decimal	H	Run until return	Ctrl+F7																																																																																																																																																																																
Jump to marked position...	Ctrl+M	Binary	B	Run to cursor	F4																																																																																																																																																																																
Error operand	Ctrl+F	Character	R	Breakpoint list	Ctrl+Alt+B																																																																																																																																																																																
Search		Segment	S	Stack trace	Ctrl+Alt+S																																																																																																																																																																																
Next code	Alt+C	Enum member...	M	Dialog Boxes																																																																																																																																																																																	
Next data	Ctrl+D	Stack variable	K	Next explored	Ctrl+A	Change sign	–	Navigate	Tab, Shift+Tab	Next unexplored	Ctrl+U	Bitwise negate	~	Immediate value...	Alt+I	String literals...	Alt+A	Toggle	Space	Next immediate value	Ctrl+I	Setup data types...	Alt+D	Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter	Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																																																																																														
Next explored	Ctrl+A	Change sign	–	Navigate	Tab, Shift+Tab																																																																																																																																																																																
Next unexplored	Ctrl+U	Bitwise negate	~	Immediate value...	Alt+I	String literals...	Alt+A	Toggle	Space	Next immediate value	Ctrl+I	Setup data types...	Alt+D	Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter	Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																																																																																																								
Immediate value...	Alt+I	String literals...	Alt+A	Toggle	Space																																																																																																																																																																																
Next immediate value	Ctrl+I	Setup data types...	Alt+D	Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter	Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																																																																																																																		
Text...	Alt+T	Edit segment...	Alt+S	Confirm	Enter, Alt+K, Ctrl+Enter																																																																																																																																																																																
Next text	Ctrl+T	Change segment register value...	Alt+G	Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4	Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous		Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																																																																																																																												
Sequence of bytes...	Alt+B	Struct var...	Alt+Q	Cancel	Esc, Alt+F4																																																																																																																																																																																
Next sequence of bytes	Ctrl+B	Select union member...	Alt+Y	Miscellaneous																																																																																																																																																																																	
Open Subviews		Undo	Ctrl+Z	Local types	Shift+F1	Calculator...	?	Functions	Shift+F3	Windows list (next)	Ctrl+Tab	Names	Shift+F4	Switch to window #1...9	Alt+1...9	Signatures	Shift+F5	Close window	Alt+F3	Segments	Shift+F7	Script command...	Shift+F2	Segment registers	Shift+F8	Exit	Alt+X	Structures	Shift+F9			Enumerations	Shift+F10			Type libraries	Shift+F11			Strings	Shift+F12																																																																																																																																												
Local types	Shift+F1	Calculator...	?																																																																																																																																																																																		
Functions	Shift+F3	Windows list (next)	Ctrl+Tab																																																																																																																																																																																		
Names	Shift+F4	Switch to window #1...9	Alt+1...9																																																																																																																																																																																		
Signatures	Shift+F5	Close window	Alt+F3																																																																																																																																																																																		
Segments	Shift+F7	Script command...	Shift+F2																																																																																																																																																																																		
Segment registers	Shift+F8	Exit	Alt+X																																																																																																																																																																																		
Structures	Shift+F9																																																																																																																																																																																				
Enumerations	Shift+F10																																																																																																																																																																																				
Type libraries	Shift+F11																																																																																																																																																																																				
Strings	Shift+F12																																																																																																																																																																																				

IDA功能详解

此处整理IDA中各种强大且好用的功能。

- 字符串
- 函数列表
- 查看代码
 - 汇编代码
 - 伪代码
 - F5查看伪代码
 - 导出伪代码
 - 函数调用关系
 - 结构体：设置好类的定义，伪代码自动解析出属性调用
- 导入和导出
- 插件
 - keypatch

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:44:07

查看代码

一般主要用IDA来查看和分析代码，代码主要分：

- 汇编代码：从原始二进制的字节码，反汇编得到的汇编代码
- 伪代码：从汇编代码用F5反编译得到的，很接近人类写的代码，人类能读懂代码逻辑的代码

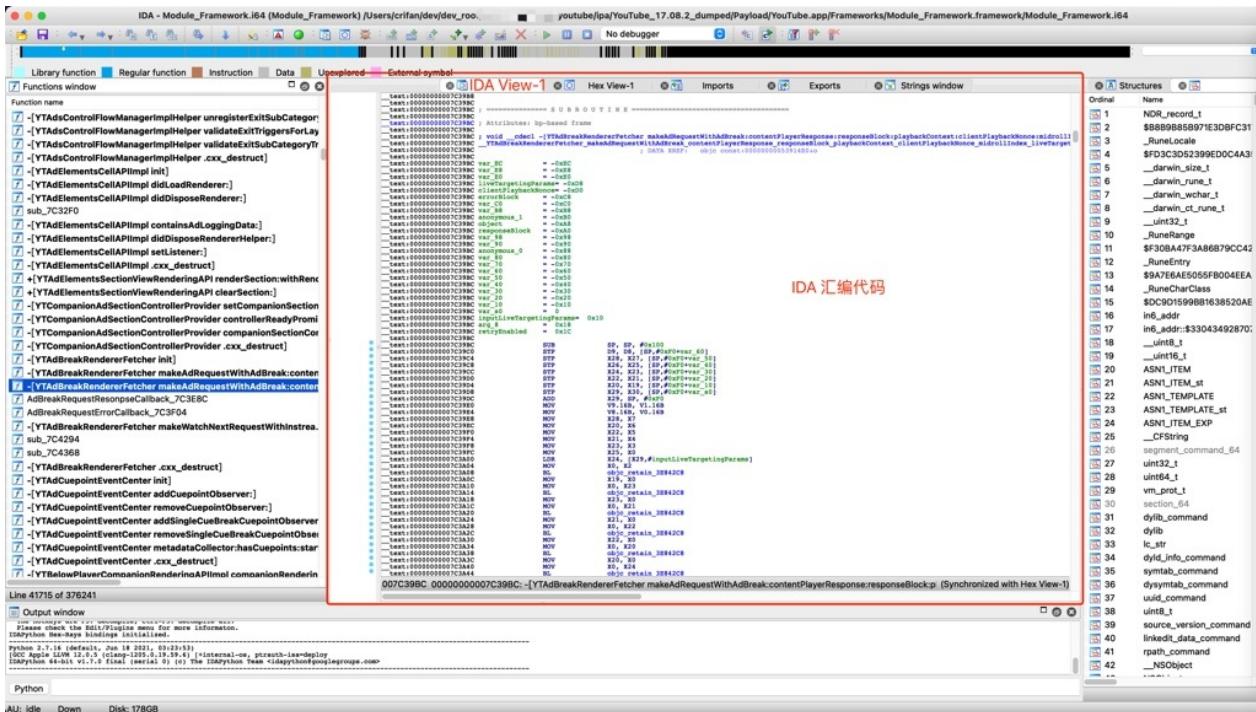
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-23 21:16:18

汇编代码

TODO:

- 【已解决】IDA使用心得：IDA汇编代码如何快速找到匹配的Xcode汇编代码
- 【已解决】IDA中查看ARM汇编的伪代码
- 【未解决】IDA中开启宏指令比如ADRP和ADD变成ADRL
- 【整理】IDA使用心得：auto comments
- 【整理】IDA使用心得：OpCode区

IDA中的 汇编代码， 是从原始二进制的字节码， 反汇编得到的汇编代码



一般来说，在逆向尝试搞懂代码逻辑时，不太需要直接查看汇编代码，因为的确很难直接看懂逻辑。

不过有些情况下，会用到汇编代码：

- iOS逆向
 - 静态分析
 - 有些汇编代码中，IDA已帮忙分析和插入了相关的解释信息，值得研究逻辑时去参考
 - 比如，YouTube逆向期间，IDA已帮忙给相关汇编加上了描述，指明了有些代码是vtable的部分
 - 便于分析和对照，寻找对应虚函数的具体实现
 - 动态调试
 - 想要找到调试期间的，Xcode中汇编代码，对应的代码逻辑
 - 往往就需要找到IDA中对应的伪代码是什么
 - 往往就需要先去找IDA中汇编代码的位置
 - 再去F5（或Tab键）跳转到对应的伪代码的位置

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-24 12:04:21

伪代码

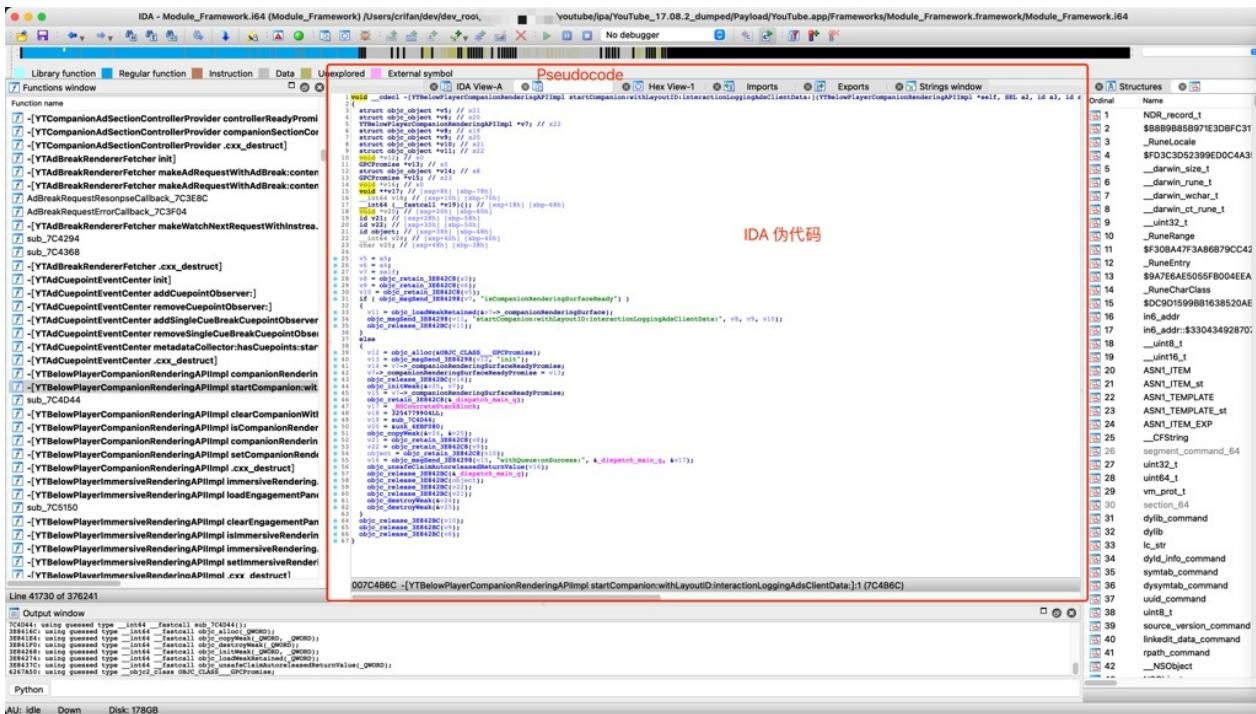
TODO:

- 【未解决】IDA使用心得：伪代码内新增变量
- 【整理】IDA使用心得：刷新当前已打开的伪代码用F5
- 【整理】IDA使用心得：伪代码中的指针+某个数值和0x开头LL结尾的数值不是一个意思
- 【整理】IDA使用心得：给伪代码添加注释
- 【整理】IDA使用心得：伪代码中的可变参数个数的函数去增加或删除参数
- 【整理】iOS逆向心得：IDA使用心得：改变值的显示格式从10进制改为16进制查看Block的flags标志位
- 【整理】iOS逆向心得：IDA Pro使用心得：给函数改名便于快速定位汇编伪代码对应关系
- 【iOS逆向心得】IDA使用心得：伪代码中在修改了别处函数定义后返回导致伪代码中函数调用参数丢失
- 【整理】IDA使用心得：反编译伪代码常见错误
 - [Failures and troubleshooting \(hex-rays.com\)](#)
- 【整理】IDA使用心得：如何理解反汇编后的伪代码的逻辑
- 【记录】IDA使用心得：objc_msgSend跳板函数重命名优化
- 【记录】IDA使用心得：伪代码改名重命名改回默认值
- 【记录】IDA使用心得：给伪代码的变量改名
- 【已解决】IDA使用心得：伪代码中如何找到对应的IDA汇编代码
- 【已解决】IDA中给汇编代码或伪代码改名
- 【整理】IDA使用心得：改名 给变量改类型
- 【整理】IDA使用心得：F5伪代码

代码逆向相关：

- 【已解决】IDA中xsp和xbp是什么意思如何定位地址
- 【整理】IDA使用心得：IDA伪代码和汇编代码 反汇编 逻辑关系 理解

IDA中，支持从 汇编代码，按 F5 快捷键去 反编译 得到的 伪代码 -> 很接近人类写的代码，人类能容易读懂代码逻辑的代码



IDA中最强的功能，应该就属这个 伪代码 了。

IDA反编译出的 伪代码：

- 质量很高：很接近原程序的代码的逻辑
- 且有很多额外好用的功能支持
 - 比如
 - 重命名: rename
 - 变量更改类型: change type
 - 增加减少参数个数
 - 自动解析出类的属性的引用
 - 等等

伪代码中，右键，支持很多功能：

- Rename
- Set type
- Set number representation
- Edit indented comment
- Edit block comment
- Hide/unhide statements
- Split/unsplit expression
- Force call type
- Set call type
- Add/del variadic arguments
- Del function argument
- Add/delete function return type
- Jump to cross reference
- Jump to cross reference globally
- Generate HTML file
- Mark/unmark as decompiled
- Copy to assembly
- Show/hide casts

而根据当前元素类型，（可能）会显示额外菜单=功能=选项：

- 局部变量
 - Reset pointer type
 - Convert to struct *
 - Create new struct type
 - Map to another variable
 - Unmap variable(s)
 - Force new variable
- 联合体union
 - Select union field
- 括号类：圆括号、中括号、花括号
 - Jump to paired paren
- 文本
 - Copy快捷键: Ctrl+C
- C表达式关键字
 - Collapse/uncollapse item

具体细节详见：

[Interactive operation \(hex-rays.com\)](http://hex-rays.com)

导出全部伪代码

IDA中，一般来说，伪代码都是针对单个函数的：反编译再查看单个函数的伪代码。

后来发现，想要导出全部伪代码，也是可以的。

详见：

- 【未解决】用插件导出IDA的YouTube的Module_Framework的全部反汇编的源码伪代码
- 【已解决】IDA中用idat64的Batch Mode尝试反编译导出YouTube的Module_Framework全部代码伪代码

标记为已编译 Mark/unmark as decompiled

This command marks the current function as decompiled. It is a convenient way to track decompiled functions. Feel free to use it any way you want. Marking a function as decompiled will change its background color to the value specified by the MARK_BGCOLOR parameter in the configuration file. The background color will be used in the pseudocode window, in the disassembly listing, and in the function list.

拷贝到汇编 Copy to assembly

This command copies the pseudocode text to the disassembly window. It is available from the popup right-click menu. Please note that only "meaningful" lines are copied. Lines containing curly braces, else/do keywords will be omitted. The copied text is represented as anterior comments in the disassembly. Feel free edit them the way you want. The copied text is static and will not change if the pseudocode text changes.

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-24 12:04:40

函数调用

TODO:

- 【整理】iOS逆向心得：IDA中以列表形式列出函数被调用的地方
 - 【整理】iOS逆向心得之IDA使用心得：查看函数被调用的所有地方即被调用函数的列表
 - 【整理】iOS逆向心得之IDA使用心得：如何快速找到真正的函数的被调用的列表函数名
- 【整理】IDA 使用心得：交叉引用以列表方式显示

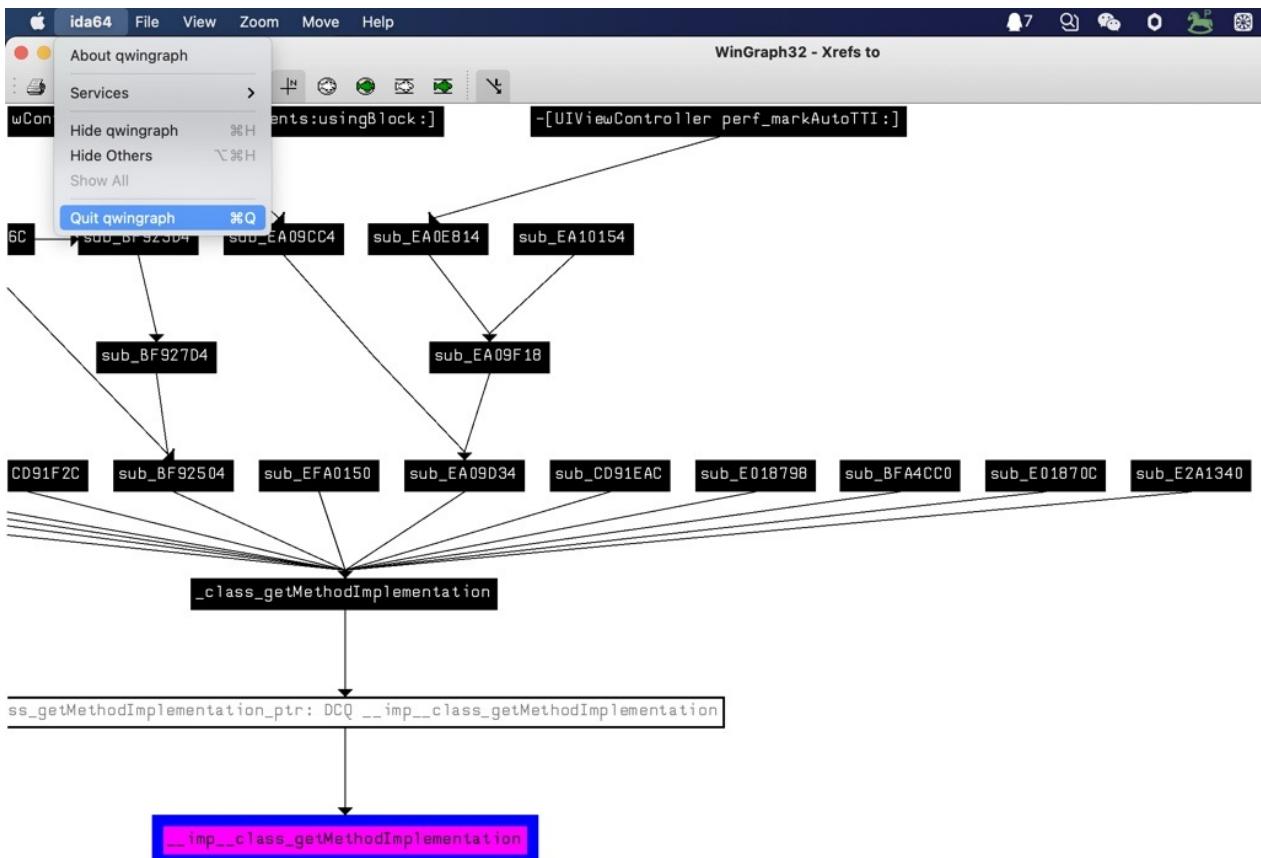
IDA中代码分析方面，对于函数的调用的关系，也有很好的支持。

iOS逆向期间，往往涉及到，想要搞懂一个函数，被其他哪些地方调用到了等等，和函数调用关系相关的内容。IDA对此支持的都很好。

Xrefs to=有哪些地方引用到了此函数

显示界面的底层实现所用的库

IDA中函数调用的graph，通过 ida64 的quit，看到的是：qwingraph



而 qwingraph，其实是一个插件，底层可视化插件

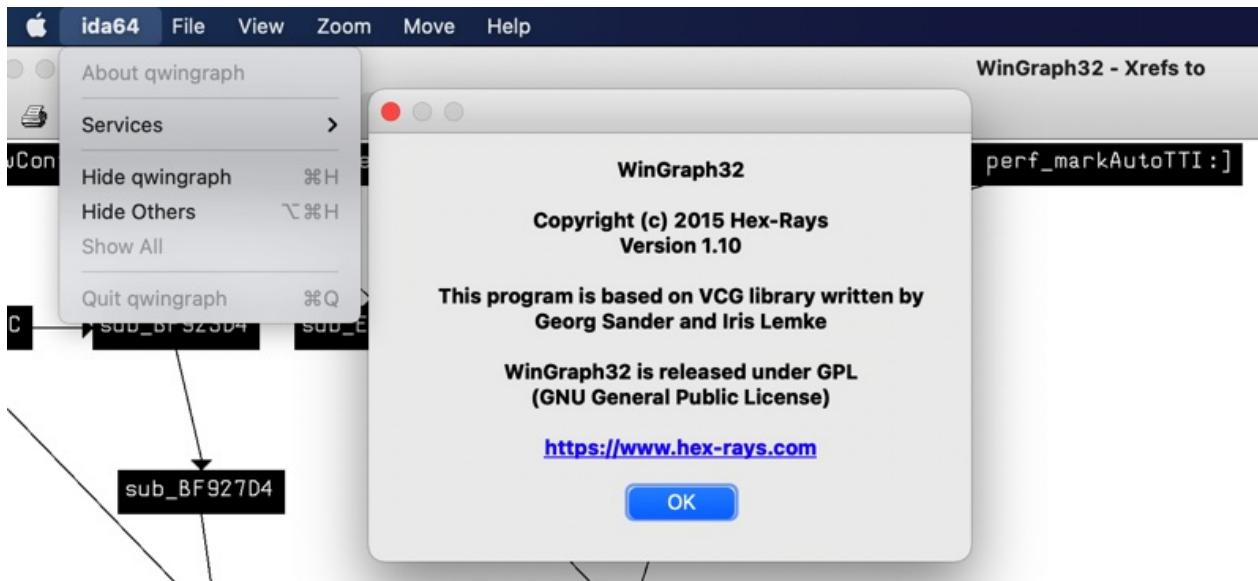
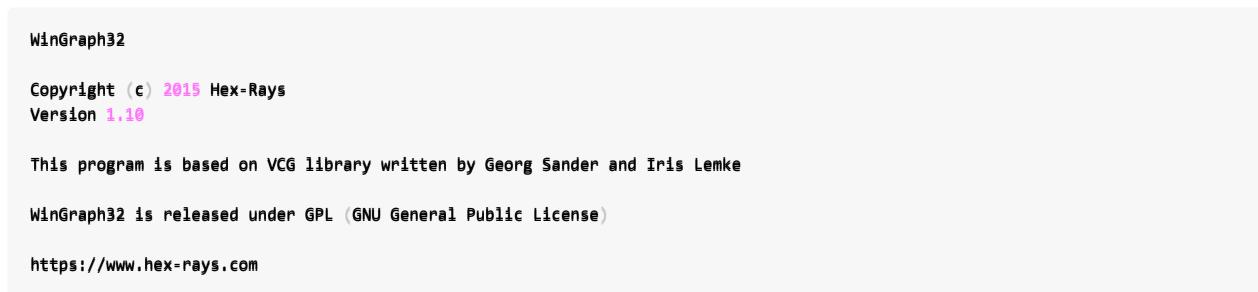
可以从

[Download center \(hex-rays.com\)](http://hex-rays.com)

找到：

- Qwingraph v1.10
 - Source code the Wingraph we use and modified (GPL)
 - https://hex-rays.com/products/ida/support/freefiles/qwingraph_src.zip

而 WinGraph32 本身关于的信息是：



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新：2022-10-23 22:49:25

结构体定义

TODO:

- 新增Structure结构体定义 + 且双击后，可以导入到数据库中
 - 【已解决】IDA中如何给Local Types中struct MLServerABRLoader加上嵌入的struct结构体定义
 - 【整理】iOS逆向心得：IDA使用心得：修改变量类型Set Ivar Type后IDA可以自动解析结构体的属性和字段
 - 【整理】IDA使用心得：类的部分字段无法解析，导致伪代码中类的属性错误，需要手动修复结构体定义
 -
-

IDA中，支持把类的原始定义，通过结构体的形式写出来（甚至自动分析出来对应结构体定义），从而后续的汇编代码和伪代码中，自动解析出类的属性和函数的调用，很是方便。

此处的类的结构体定义，主要涉及到两方面：

- `Structures`
- `Local Types`
-

◦

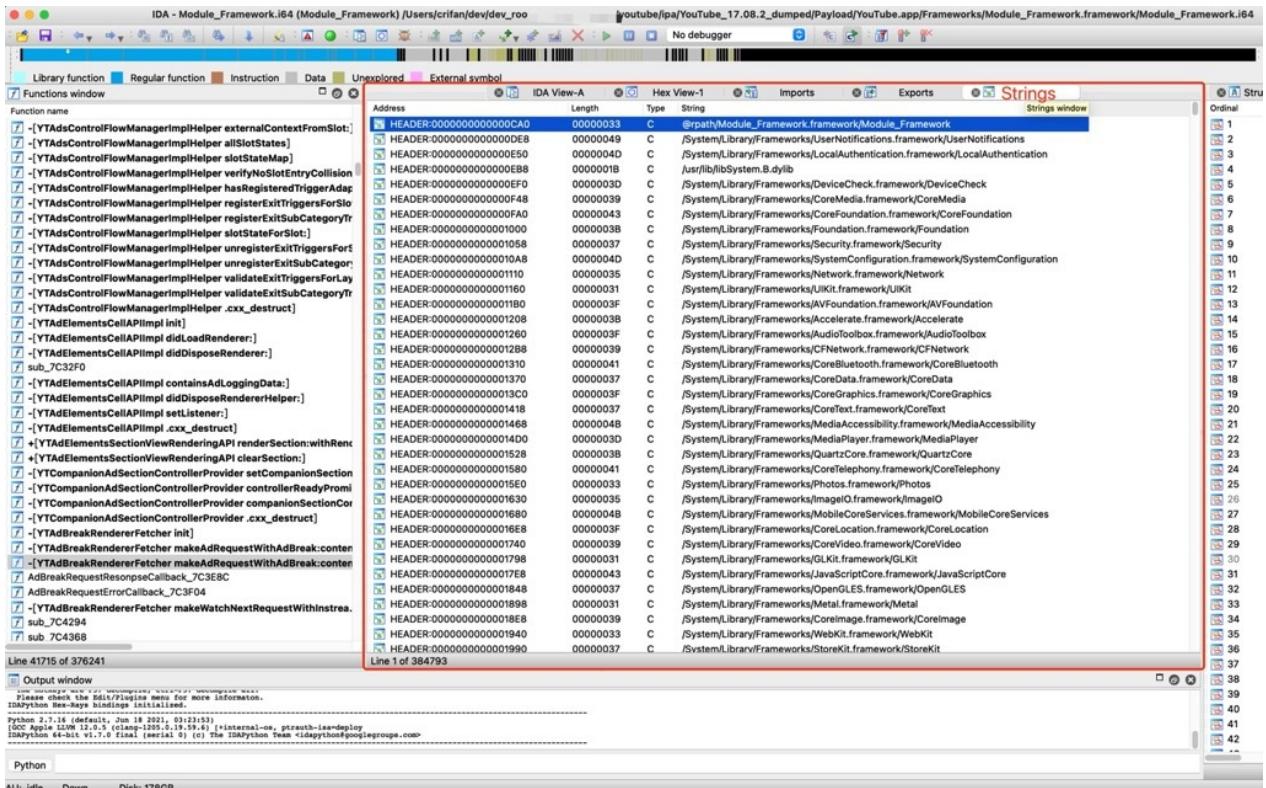
且也支持新增自定义的结构体，更改已有类的结构体的字段定义等，很强大好用的功能。

- 创建结构体

◦

字符串

IDA也能自动分析出，二进制中有哪些字符串，放到一个单独视图 Strings，供分析和研究。



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:59:05

函数列表

IDA可以分析出，二进制中的所有的函数，并且列出函数的列表，供查找和定位，以及后续代码逻辑的研究。

ida64 File Edit Jump Search View Debugger Options Window

IDA - Module_Framework.i64 (Module_Framework) /User

Library function Regular function Instruction Data Unexplored

f Functions window

Function name	Segment
-[GPBBoolArray yt_array]	__text
-[GPBBoolArray yt_addNumber:]	__text
+[GPBEnumArray arrayWithCapacity:]	__text
+[GPBEnumArray yt_arrayWithArray:]	__text
-[GPBEnumArray yt_numberAtIndex:]	__text
-[GPBEnumArray yt_array]	__text
-[GPBEnumArray yt_addNumber:]	__text
+[GPBUInt32UInt32Dictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32UInt32Dictionary yt_dictionary]	__text
-[GPBUInt32UInt32Dictionary yt_enumerateKeysAndValuesUsingBlo...]	__text
sub_1B33B80	__text
-[GPBUInt32UInt32Dictionary yt_setValue:forKey:]	__text
+[GPBUInt32UInt32Dictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32UInt32Dictionary yt_dictionary]	__text
-[GPBUInt32UInt32Dictionary yt_enumerateKeysAndValuesUsingBlo...]	__text
sub_1B33EFC	__text
-[GPBUInt32UInt32Dictionary yt_setValue:forKey:]	__text
+[GPBUInt32UInt64Dictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32UInt64Dictionary yt_dictionary]	__text
-[GPBUInt32UInt64Dictionary yt_enumerateKeysAndValuesUsingBlo...]	__text
sub_1B3427C	__text
-[GPBUInt32UInt64Dictionary yt_setValue:forKey:]	__text
+[GPBUInt32UInt64Dictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32UInt64Dictionary yt_dictionary]	__text
-[GPBUInt32UInt64Dictionary yt_enumerateKeysAndValuesUsingBlo...]	__text
sub_1B345FC	__text
-[GPBUInt32UInt64Dictionary yt_setValue:forKey:]	__text
+[GPBUInt32BoolDictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32BoolDictionary yt_dictionary]	__text
-[GPBUInt32BoolDictionary yt_enumerateKeysAndValuesUsingBloc...]	__text
sub_1B3497C	__text
-[GPBUInt32BoolDictionary yt_setValue:forKey:]	__text
+[GPBUInt32FloatDictionary yt_dictionaryWithDictionary:]	__text
-[GPBUInt32FloatDictionary yt_dictionary]	__text
-[GPBUInt32FloatDictionary vt_enumerateKevsAndValuesUsinaBlo...]	text

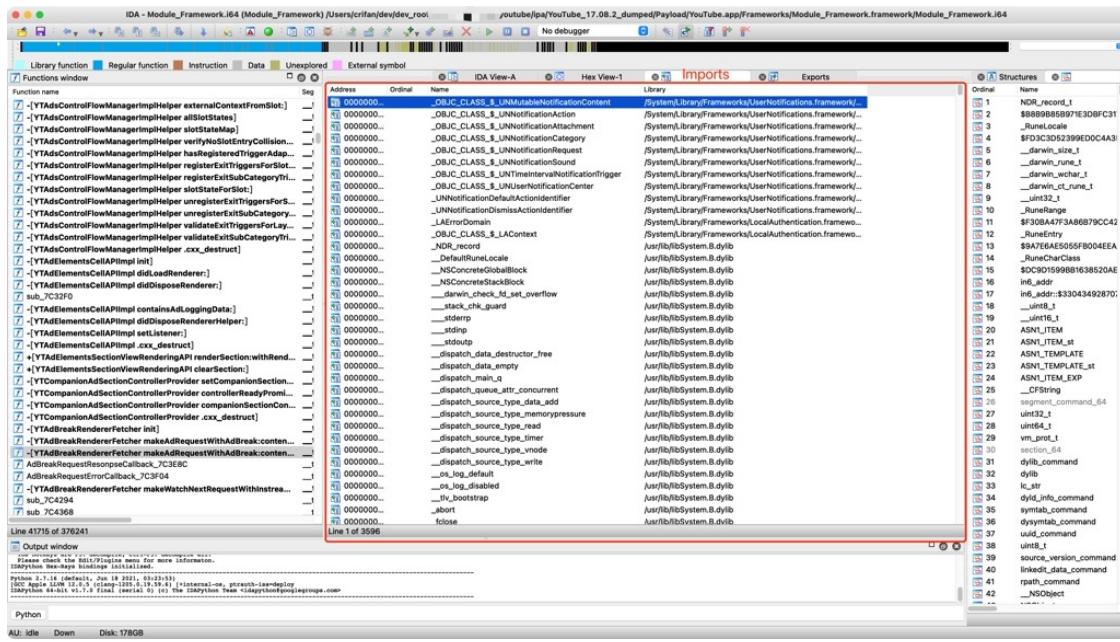
Line 159826 of 376241

导入和导出

IDA中还有2个独立的窗口是：

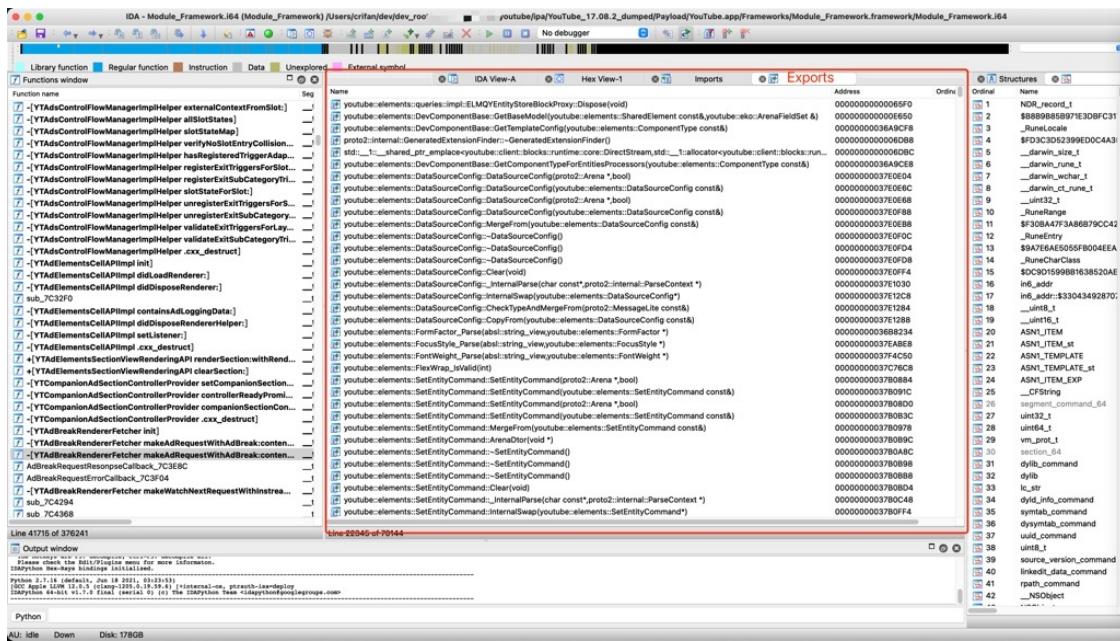
- Import=导入

- 当前二进制，导入了哪些函数
 - = 引用了外部的，别的库的哪些函数
- 举例



- Export=导出

- 当前二进制，导出了哪些函数
 - 供别处（比如自己程序的另外的二进制中去）使用
- 举例



在逆向分析时，可以根据，导出和导入，找到一些相关线索

比如，在iOS逆向的越狱检测和反越狱检测中，就可以去找，当前二进制是否导入了，常用于越狱检测的一些系统函数。

详见：

【整理Book】iOS逆向开发：越狱检测和反越狱检测

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:55:17

插件

IDA还支持插件的机制，可以扩展支持更多更强的各种功能。

常见的插件有：

- keypatch

有机会参考

iOS重打包绕过签名校验防护 | La0s

使用Keypatch插件patch两条汇编 MOV X0, #0 && RET

IDA View-A Pseudocode-A Hex View-1 Structures Enums

```
text:00000001005ED6D4 ; bool __cdecl -[UIDevice isJailbroken](UIDevice *self, SEL)
text:00000001005ED6D4 __UIDevice_isJailbroken_ ; DATA XREF:objc const:00000001016211E0+o
text:00000001005ED6D4 var_150 = -0x150
text:00000001005ED6D4 var_138 = -0x138
text:00000001005ED6D4 var_130 = -0x130
text:00000001005ED6D4 var_128 = -0x128
text:00000001005ED6D4 var_118 = -0x118
text:00000001005ED6D4 var_108 = -0x108
text:00000001005ED6D4 var_F8 = -0xF8
text:00000001005ED6D4 var_78 = -0x78
text:00000001005ED6D4 var_68 = -0x68
text:00000001005ED6D4 var_58 = -0x58
text:00000001005ED6D4 var_50 = -0x50
text:00000001005ED6D4 var_40 = -0x40
text:00000001005ED6D4 var_30 = -0x30
text:00000001005ED6D4 var_20 = -0x20
text:00000001005ED6D4 var_10 = -0x10
text:00000001005ED6D4 var_s0 = 0
text:00000001005ED6D4
text:00000001005ED6D4 MOV X0, #0 ; Keypatch modified this from:
text:00000001005ED6D4 ; SUB SP, SP, #0x160
text:00000001005ED6D8 RET | ; Keypatch modified this from:
text:00000001005ED6D8 ; STP X28, X27, [SP,#0x100]
text:00000001005ED6DC ; --
text:00000001005ED6DC STP X26, X25, [SP,#0x110]
text:00000001005ED6E0 STP X24, X23, [SP,#0x120]
text:00000001005ED6E4 STP X22, X21, [SP,#0x130]
text:00000001005ED6E8 STP X20, X19, [SP,#0x140]
text:00000001005ED6EC STP X29, X30, [SP,#0x150]
text:00000001005ED6F0 ADD X29, SP, #0x150
text:00000001005ED6F4 ADRP X8, #__stack_chk_guard_ptr@PAGE
text:00000001005ED6F8 LDR X8, [X8,#__stack_chk_guard_ptr@PAGEOFF]
```

去试试：

- 用keypatch插件，patch打补丁，插入（汇编）指令

其他插件

Third-party plugins

2

[onethawt/idapugins-list](https://github.com/onethawt/idapugins-list): A list of IDA Plugins

cifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved. powered by Gitbook最后更新: 2022-10-24 11:21:42

IDA使用心得

TODO:

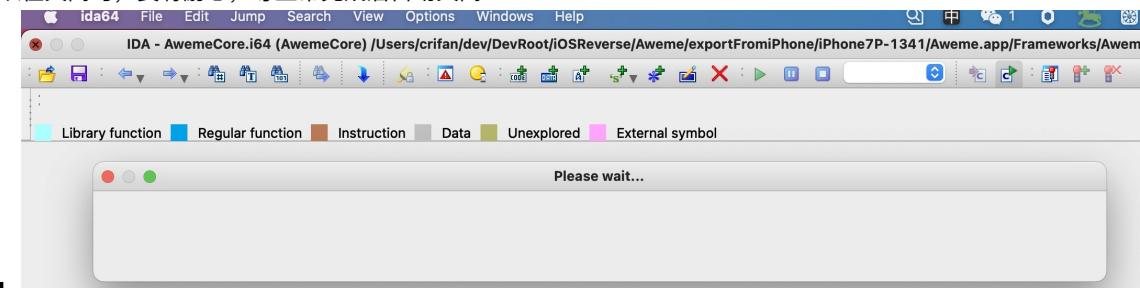
- 【记录】IDA Pro使用记录

使用IDA逆向分析和调试期间，有很多心得，整理如下，供参考。

打开和关闭

IDA，对于打开比较大（比如100多MB）的二进制的话：

- 打开：
 - 首次打开：需要的解析时间很长
 - 再次打开：也需要点时间（大概几十秒，根据已解析的数据库大小决定）
- 关闭：保存改动，写入数据库，也会耗时较长
 - 所以在关闭时，要有耐心，等正常完成后自动关闭



打开

双击 .i64 文件，可以调用IDA（中的 ida64）去打开：



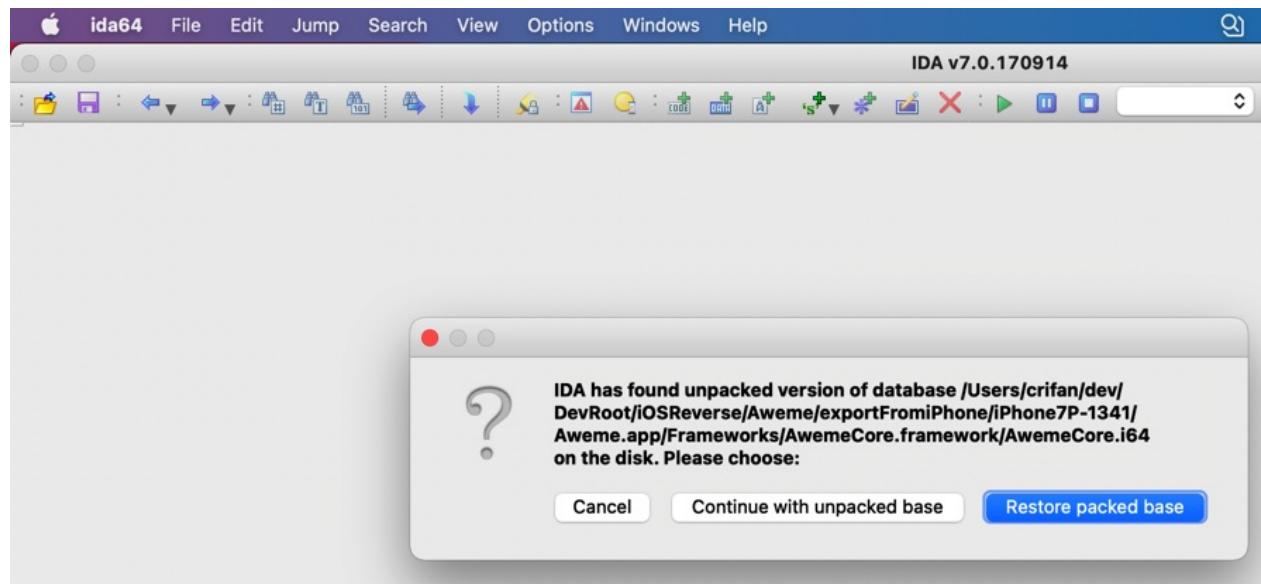
即可正常打开。

如果之前没有正常关闭，则会提示：

IDA has found unpacked version of database on the disk. Please choose:

- Cancel
- Continue with unpacked base
- Restore packed base

一般选： Restore packed base



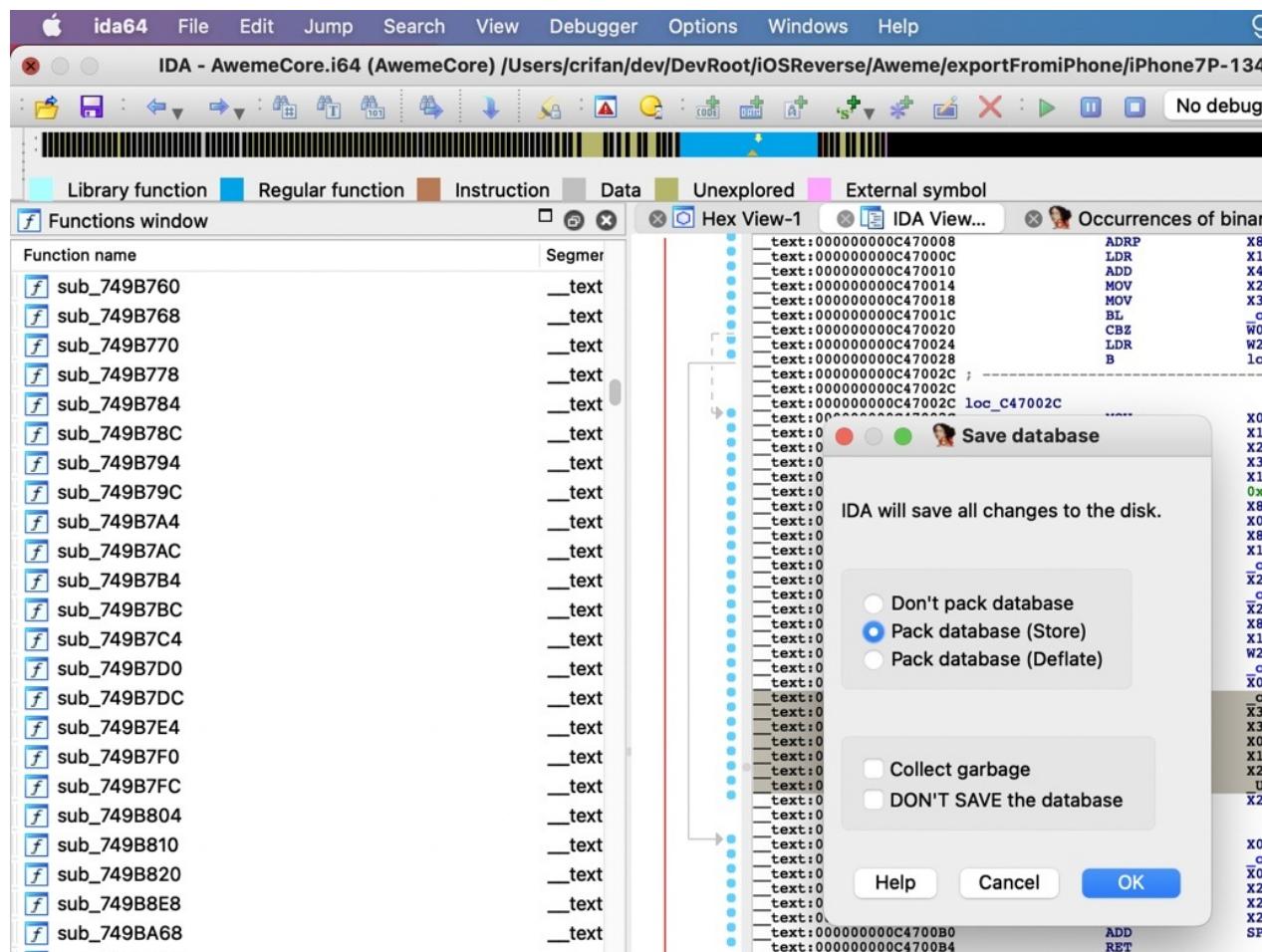
即可。

关闭

去点击关闭时，一般会有提示：

IDA will save all changes to the disk.

- Don't pack database
- Pack database (Store)
- Pack database (Deflate)



一般选默认的： Pack database (Store)

然后保存出的是单个文件： .i64 :



关闭的心得

- 没有特殊情况时，千万不要轻易在关闭时，强制杀掉进程
 - 否则可能会导致：之前的数据库被损坏，再次打开后，之前分析的数据丢失了
 - 比如自己的优化改动，比如给函数变量重命名等

比如之前自己就遇到过：

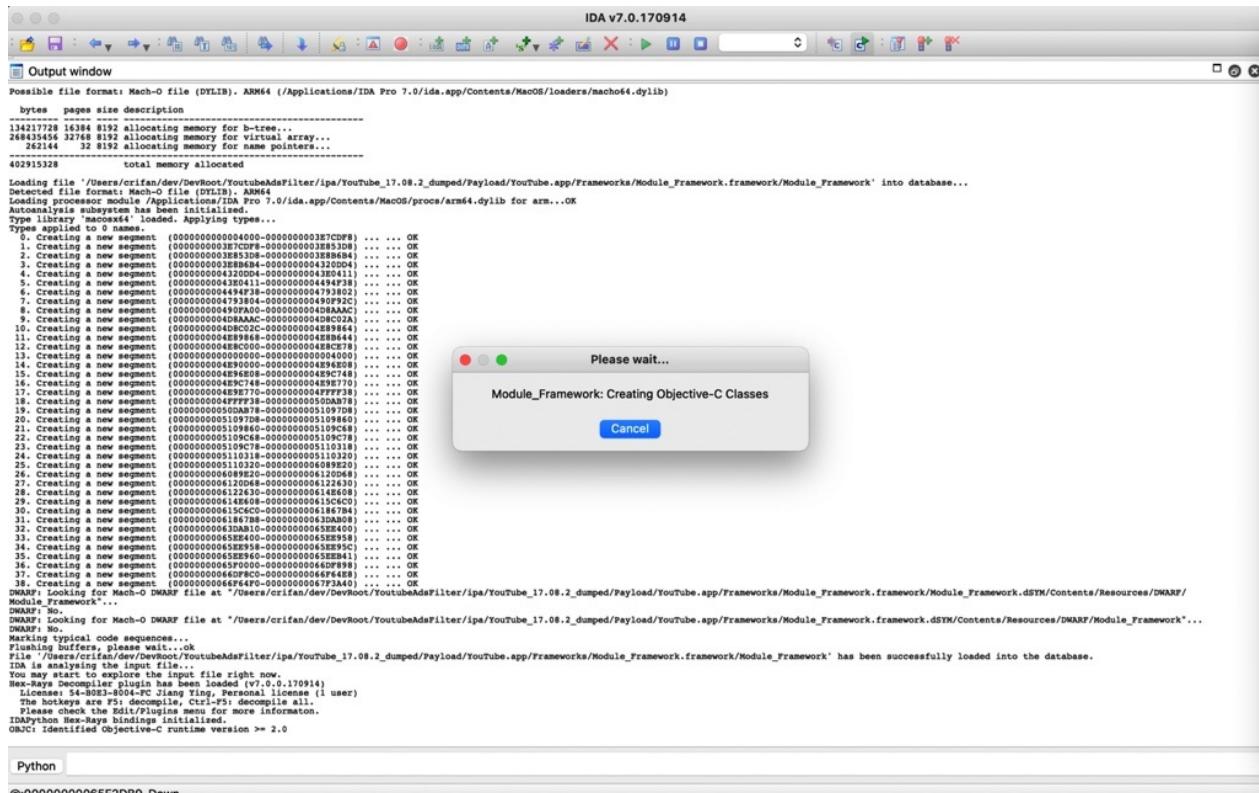
折腾：

【未解决】IDA中用idat64的Batch Mode尝试反编译导出YouTube的Module_Framework全部代码伪代码

期间，对于IDA的text mode的导出过程，觉得是卡死了，强制杀掉了IDA进程

再去打开IDA的GUI mode，即正常通过IDA图标双击打开，结果报错了

只好去，删除残留文件，全新的加载打开



界面布局

- 【整理】IDA使用心得：把类结构Structures的窗口放在右边方便对比查看

代码分析

*(_DWORD *) 的含义

```
v7 = (*(_DWORD *))(v7 + 8);
```

==

```
v7 = (v7 + 8);
```

==

汇编的：

```
MOV v7, DWORD PTR v7 + 0x8
```

含义解释：

v7 is assigned with the value located at address v7+8sizeof(DWORD). For example, if v7 = 0abcd0123 then v7 + 8sizeof(DWORD) = 0abcd0143. Whatever is located at 0abcd0143 will be assigned to v7.

函数跳转

- Jump跳转
 - 【整理】IDA使用心得：根据selector跳转到函数

- 【整理】IDA使用心得：跳转到历史列表的函数位置

函数

- 【整理】IDA使用心得：通过给函数Set Item Type去修正函数的参数的个数和类型和返回值类型

类

- 【整理】IDA心得：自定义的类的属性偏移量和自动生成的偏移量不匹配

bug

- 【整理】IDA使用心得：iOS的ObjC伪代码反编译翻译的有错误不够准确

如何处理数组Array

Working with array

<https://hex-rays.com/wp-content/uploads/2021/10/igor-tip-of-the-week-S01.pdf>

Arrays are used in IDA to represent a sequence of multiple items of the same type: basic types (byte, word, dword etc.) or complex ones (e.g. structures).

Creating an array

To create an array:

1. Create the first item;
2. Choose "Array..." from the context menu, or press *;
3. Fill in at least the Array size field and click OK.

Step 1 is optional; if no data item exists at the current location, a byte array will be created.

Hint: if you select a range before pressing *, Array size will be pre-filled with the number of items which fits into the selected range.

Quick menu navigation

Array parameters affect how the array is displayed in the listing and can be set at the time the array is first created or any time later by pressing *.

- Array size: total number of elements in the array;
- Items on a line: how many items (at most) to print on one line. 0 means to print the maximum number which fits into the disassembly line;
- Element print width: how many characters to use for each element. Together with the previous parameter can be used for formatting arrays into nice-looking tables.

For example: 8 items per line, print width -t:

```
db 1, 2, 3, 4, 5, 6, 7, 8
db 9, 10, 11, 12, 13, 14, 15, 16
db 17, 18, 19, 20, 21, 22, 23, 24
db 25, 255, 255, 255, 255, 255, 255, 26
db 27, 28, 29, 30, 31, 32, 33, 34
db 35, 36, 37, 38, 39, 40, 41, 42
```

print width 0:

```
db 1, 2, 3, 4, 5, 6, 7, 8
db 9, 10, 11, 12, 13, 14, 15, 16
db 17, 18, 19, 20, 21, 22, 23, 24
db 25, 255, 255, 255, 255, 255, 255, 26
db 27, 28, 29, 30, 31, 32, 33, 34
db 35, 36, 37, 38, 39, 40, 41, 42
```

print width 5:

```
db 1, 2, 3, 4, 5, 6, 7, 8
db 9, 10, 11, 12, 13, 14, 15, 16
db 17, 18, 19, 20, 21, 22, 23, 24
db 25, 255, 255, 255, 255, 255, 255, 26
db 27, 28, 29, 30, 31, 32, 33, 34
db 35, 36, 37, 38, 39, 40, 41, 42
```

• Use "dup" construct: for assemblers that support it, repeated items with the same value will be collapsed into a dup expression instead of printing each item separately.
 dup off: db 0FFh, 0FFh, 0FFh, 0FFh, 0FFh
 dup on: db 6 dup(0FFh)

• Signed elements: integer items will be treated as signed numbers;
 • Display indexes: for each line, first item's array index will be printed in a comment.
 • Create as array: if unchecked, IDA will convert the array into separate items.

附录

下面列出相关参考资料。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-03-17 20:39:28

文档和资料

TODO:

- 【整理】学习IDA教程：The IDA Pro Book
-

此处整理IDA相关的文档、教程、书籍等有价值的参考资料。

IDA官网

IDA的官网：

<https://www.hex-rays.com>

本身就有很多不错的教程和资料。

IDA官网在线文档

- 文档总入口：<https://hex-rays.com/documentation/>
 - Help总入口：<https://hex-rays.com/products/ida/support/idadoc/index.shtml>
 - 反编译器：<https://hex-rays.com/products/decompiler/manual/index.shtml>
 - 其中最常用的一些：
 - 伪代码中右键菜单中的各种功能
 - [Interactive operation](#)
 - 常见问题和错误
 - [Failures and troubleshooting](#)

IDA的tip of week

[tip of week index \(hex-rays.com\)](#)

Usage: basic and advanced usage of IDA features

- #01: Lesser-known keyboard shortcuts in IDA
- #03: Selection in IDA
- #04: More selection!
- #05: Highlight
- #09: Reanalysis
- #13: String literals and custom encodings
- #14: Comments in IDA
- #15: Comments in structures and enums
- #28: Functions list
- #30: Quick views
- #31: Hiding and Collapsing
- #34: Dummy names
- #35: Demangled names
- #36: Working with list views in IDA
- #37: Patching
- #46: Disassembly operand representation
- #47: Hints in IDA

Navigation: moving around the database

- #16: Cross-references
- #17: Cross-references 2
- #20: Going places
- #23: Graph view
- #38: Hex view
- #48: Searching in IDA
- #49: Navigation band
- #50: Execution flow arrows

Types: working with types

- #10: Working with arrays

Hidden: hidden gems, not widely known but useful functionality

- #06: IDA Release notes
- #19: Function calls
- #21: Calculator and expression evaluation feature in IDA
- #24: Renaming registers
- #39: Export Data
- #41: Binary file loader
- #44: Hex dump loader

Decompiler: related to the Hex-Rays decompiler

- #18: Decompiler and global cross-references
- #27: Fixing the stack pointer
- #40: Decompiler basics
- #42: Renaming and retyping in the decompiler
- #43: Annotating the decompiler output
- #45: Decompiler types

Automation: automating repetitive tasks

- #07: IDA command-line options cheatsheet
- #08: Batch mode under the hood
- #32: Running scripts

Customization: customizing IDA UI to better suit your workflow

- #02: IDA UI actions and where to find them
- #22: IDA desktop layouts
- #25: Disassembly options
- #26: Disassembly options 2
- #29: Color up your IDA
- #33: IDA's user directory (IDAUSR)

有很多有用提示

有空可以看看

IDA官网的下载中心

IDA官网的下载中心：

[Download center \(hex-rays.com\)](https://hex-rays.com/download-center/)

有很多相关内容，可供学习和使用：

- SDK and Utilities
 - Some downloads are only available to IDA Pro users and require a password which can be found in the latest download email.
 - IDA SDK 7.7
 - Develop processor modules, loaders and extensions - extended with the source of 30+ modules and 20+ loaders.
 - Please check out the SDK documentation online (or download the zip file for offline use).
 - Flair 7.7
 - Add your own compiler libraries to the FLIRT engine
 - IDAClang 7.7
 - A type library generator based on libclang. Use this when parsing complex C++ code that tilib cannot handle.
 - Tilib 7.7
 - Create your own type libraries

- Loadint 7.7
 - Create your own disassembler comment databases
- idsutils 7.7
 - Create your own IDS files from DLLs
- ios_deploy
 - iOS helper utility to manipulate iOS devices
- PIN tool
 - The source code of our PIN tool. It creates a debugger backend out of Intel's PIN framework
 - See: PIN framework
- TVision 2015 library
 - For the IDA text interface (source code)
- Qwingraph v1.10
 - Source code the Wingraph we use and modified (GPL)
- Sample plugins
 - Stealth
 - Stealth against anti-debugging tricks
 - findcrypt
 - Identifies some frequently used block ciphers
 - highlighter
 - Highlights code that has been single stepped through in a debugging session
 - unispector
 - Extracts unicode strings from an IDA database
 - IDA Pro, Python and Qt
 - Migrating PySide code to PyQt5
 - Using custom viewers from IDAPython
 - Augmenting IDA UI with your own actions
 - Plugin contest pages
 - Our plugins contest pages offer many useful plugins!
 - By year: 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019
- User contributions
 - Plugins
 - COM Interface Plugin
 - By Dieter Spaar
 - Sobek
 - A simple data-flow analysis plugin by JF Michel
 - PDBPlus
 - By Dean Ashton
 - IDB_2_PAT
 - By J.C Roberts
 - strucrec
 - By Halvar Flake
 - Class Informer plugin
 - To reconstruct C++ classes using the RTTI info
 - By Sirmabus
 - IDC scripts
 - Visual Basic Disassembly IDC script
 - To assist in the disassembly of VB5/VB6 hostile code
 - By Reginald Wong
 - IDC Delphi script
 - Delphi constants and class definitions
 - By Dietrich Teickner
 - h2enum
 - converts C/C++ header files to IDA enums - Nice if you have no TIL files

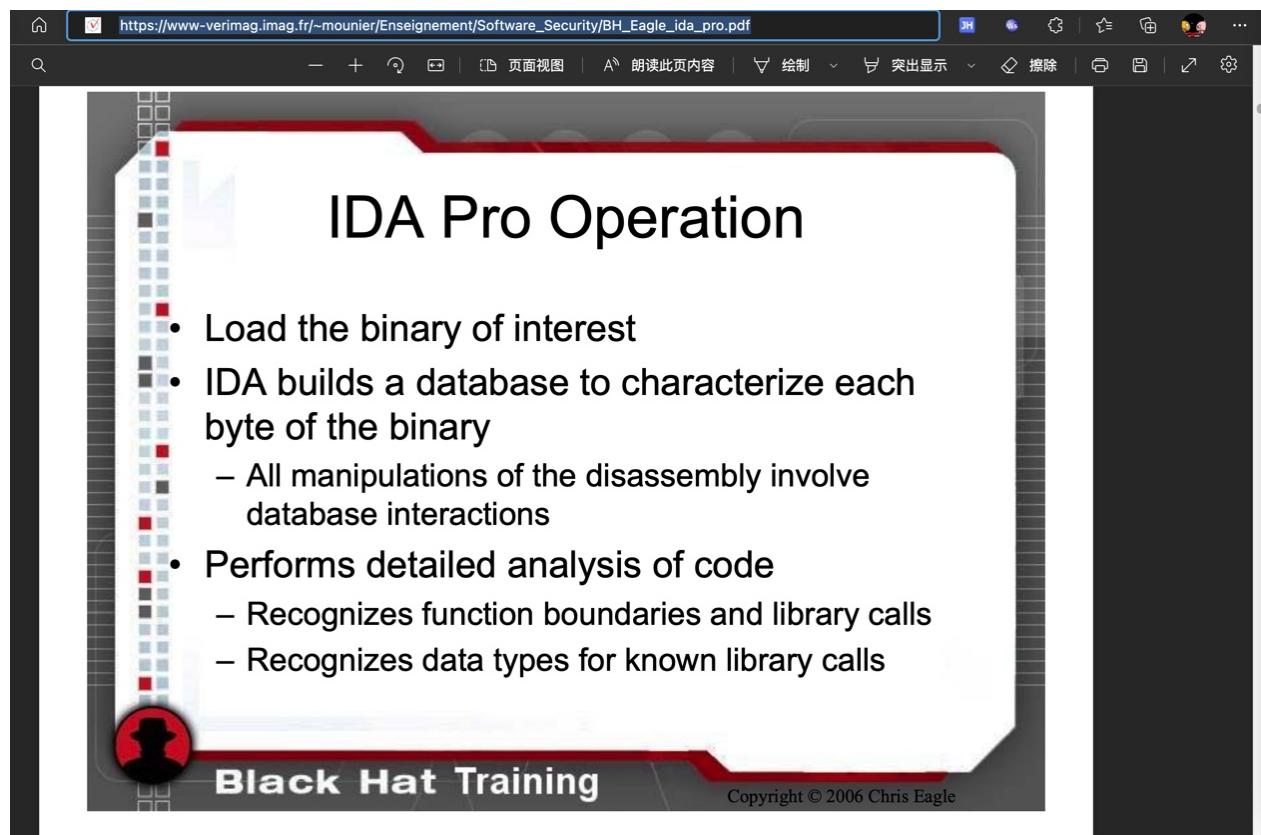
- By Leonid Lisovsky
- IDC PDR script
 - Useful in the analysis of PDR files (port drivers)
 - By Huang Yu
- Loader script
 - For VC++ and Borland C++
 - By Toshiyuki Tega
- Microchipss 16F84 PIC script
 - Defines SFR and bit names for Microchip's 16F84 PIC processor but can be used as a template for other processors
 - By an anonymous contributor
- dumpinfo
 - By JC Roberts
- Pentica-B script
 - Pentica-B import script for the H8-8300
 - By Tom Hayes
- H8 script
 - Improves the initial H8 autoanalysis
 - By Tom Hayes
- Processor modules
 - AMD 29K processor module
 - By Arne Wichmann
 - NEC V830 processor module
 - By Ben Byer
 - Samsung SAM8
 - By Andrew de Quincey. Also available is a plugin that generates files compatible with the SAMA assembler.
- Miscellaneous
 - symload
 - By Dainis Jonitis
 - PE utilities
 - A set of extremely useful PE utilities
 - By Atli Mar Gudmundsson
 - H8 utilities
 - A few utilities that could be useful to H8 developers
 - By Tom Hayes

抽空可以好好找找看看，有哪些值得好好利用的东西。

BH_Eagle_ida_pro.pdf

无意间看到的别人整理的IDA的内容：

[BH_Eagle_ida_pro.pdf](#)



下载到此处 [BH_Eagle_ida_pro.pdf](https://www-verimag.imag.fr/~mounier/Enseignement/Software_Security/BH_Eagle_ida_pro.pdf) 供下载和学习。

其他待学习和待研究的资料

- [IDA software reverse engineering - Code World \(codetd.com\)](#)
 - IDA的用法，内容很多，值得看看。
- [IDA Pro Tips to Add to Your Bag of Tricks – PT SWARM \(ptsecurity.com\)](#)
 - 这里有很多感觉有用的内容，值得参考的内容，抽空去研究学习
-

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:19:55

参考资料

- 【未解决】研究抖音越狱检测逻辑: _RxAnnotationInlineLoader的load
- 【未解决】IDA中用idat64的Batch Mode尝试反编译导出YouTube的Module_Framework全部代码伪代码
-
- [Failures and troubleshooting \(hex-rays.com\)](#)
- [Igor's tip of the week #49: Navigation band – Hex Rays \(hex-rays.com\)](#)
- [tip of week index \(hex-rays.com\)](#)
- [Interactive operation \(hex-rays.com\)](#)
- [IDA_Pro_Shortcuts.pdf \(hex-rays.com\)](#)
- [Download center \(hex-rays.com\)](#)
- [IDA Free](#)
- [IDA Evaluation](#)
- [BH_Eagle_ida_pro.pdf](#)
- [IDA software reverse engineering - Code World \(codetd.com\)](#)
- [IDA Pro Tips to Add to Your Bag of Tricks – PT SWARM \(ptsecurity.com\)](#)
- [Hex-Rays interactive operation: Mark/unmark as decompiled](#)
- [Hex-Rays interactive operation: Copy to assembly](#)
- [ida - What is the meaning of \(_DWORD \) - Reverse Engineering Stack Exchange](#)
- [iOS重打包绕过签名校验防护 | LaOs](#)
- [Interactive operation](#)
- [Failures and troubleshooting](#)
- [Third-party plugins](#)
- [onethawt/idaplugins-list: A list of IDA Plugins](#)
- [IDA Help: The Interactive Disassembler Help Index](#)
-

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-24 11:22:19