



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



恶意代码分析与防治技术

## 第6章 IDA Python

王志

zwang@nankai.edu.cn

南开大学 网络空间安全学院

2023-2024学年



允公允能 日新月异

# 本章知识点

- IDAPython
- 函数
- 操作数
- 基本块
- 交叉引用
- 搜索
- 重点：手工病毒分析的自动化编程



南开大学  
Nankai University



允公允能 日新月异

- Chapter 4 A Crash Course in x86 Disassembly
- Chapter 5 IDA Pro
- Chapter 6 Recognizing C Constructs in Assembly



南開大學  
Nankai University



南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異

A light blue world map is centered in the background of the slide.

IDA Python

IDA Pro的静态分析能力已经很强了，为什么还要学习IDA Python编程？

作答



允公允能 日新月异

## Using IDAPython to Make Your Life Easier: Part 3

30,798 people reacted

👍 3

5 min. read

SHARE



By Josh Grunzweig  
January 4, 2016 at 9:00 AM  
Category: Unit 42  
Tags: IDA Pro, IDAPython

While debugging in IDA Pro, there are often situations where an analyst wishes to break on a specific address, but only when a certain condition occurs. An example of this might include breaking on a call to a particular function, but only when a specific argument is passed to it. Another instance I personally run into is breaking when a specific library is loaded into my analysis virtual machine. Today, I'm going to look at this specific problem and discuss ways to handle it with IDAPython.

- <http://researchcenter.paloaltonetworks.com/2016/01/using-idapython-to-make-your-life-easier-part-3/>
- <http://researchcenter.paloaltonetworks.com/2016/01/using-idapython-to-make-your-life-easier-part-4/>



南开大学  
Nankai University



允公允能 日新月异

# IDA Python

- 编写IDA自动分析脚本
  - IDC
  - IDAPython
- IDAPython
  - IDA的所有API函数
  - Python的功能模块
  - 编写功能强大的自动分析脚



南开大学  
Nankai University



允公允能 日新月异

# IDAPython

- IDAPython源代码
  - <https://github.com/idapython/src>
- IDAPython的文档
  - [http://www.hex-rays.com/idapro/idapython\\_docs/](http://www.hex-rays.com/idapro/idapython_docs/)
- IDAPython教材
  - **The Beginner's Guide to IDAPython**
  - [https://github.com/ExpLife0011/IDAPython\\_Note/blob/master/IDAPython-Book.pdf](https://github.com/ExpLife0011/IDAPython_Note/blob/master/IDAPython-Book.pdf)

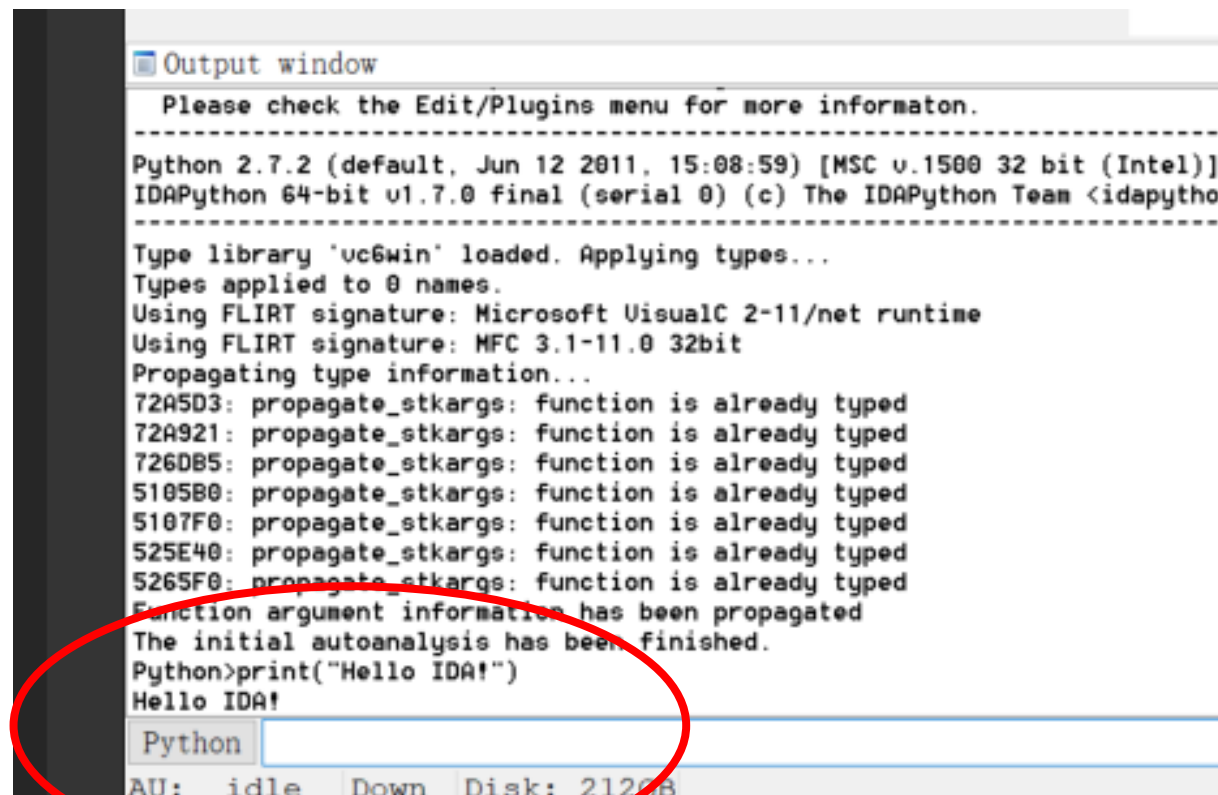
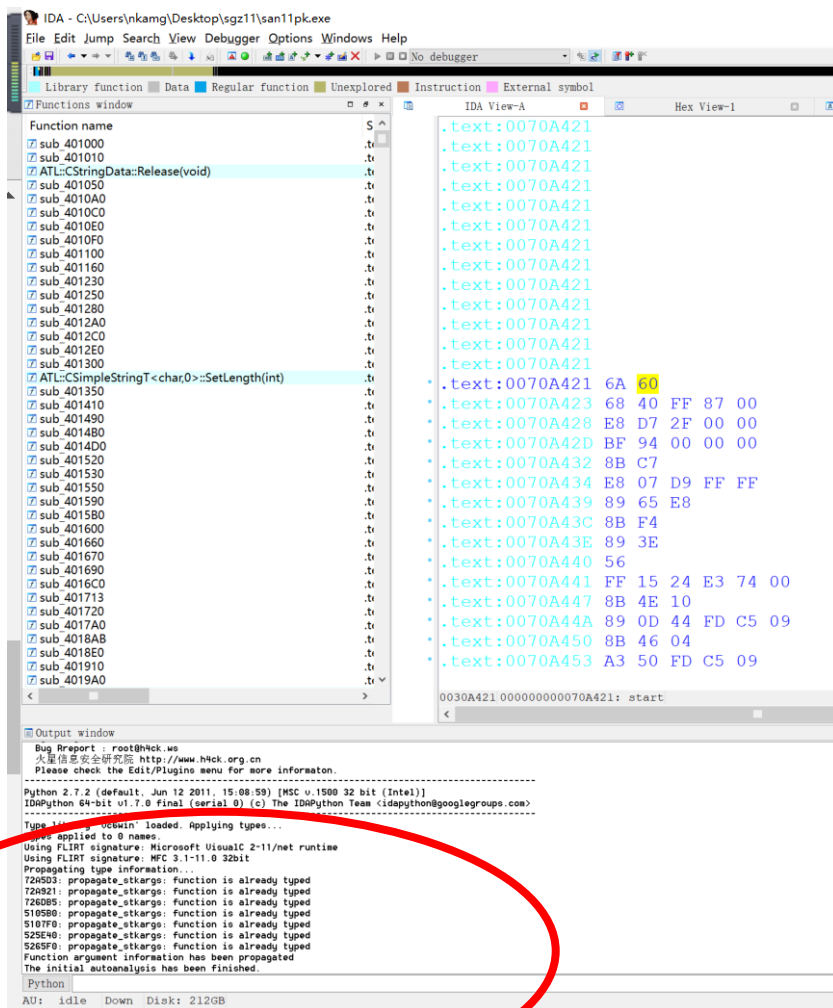


南开大学  
Nankai University



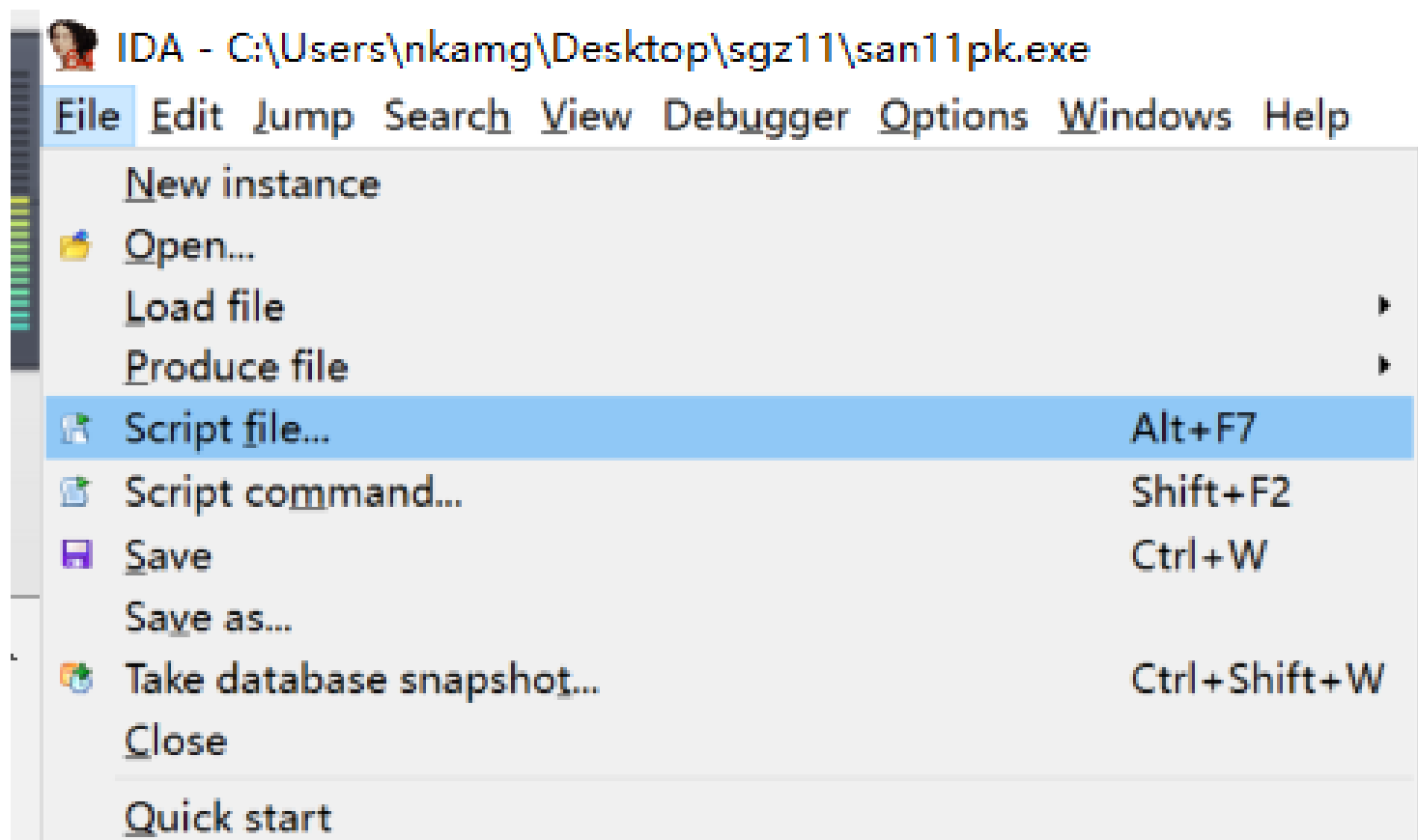
允公允能 日新月异

# IDAPython 命令输入框

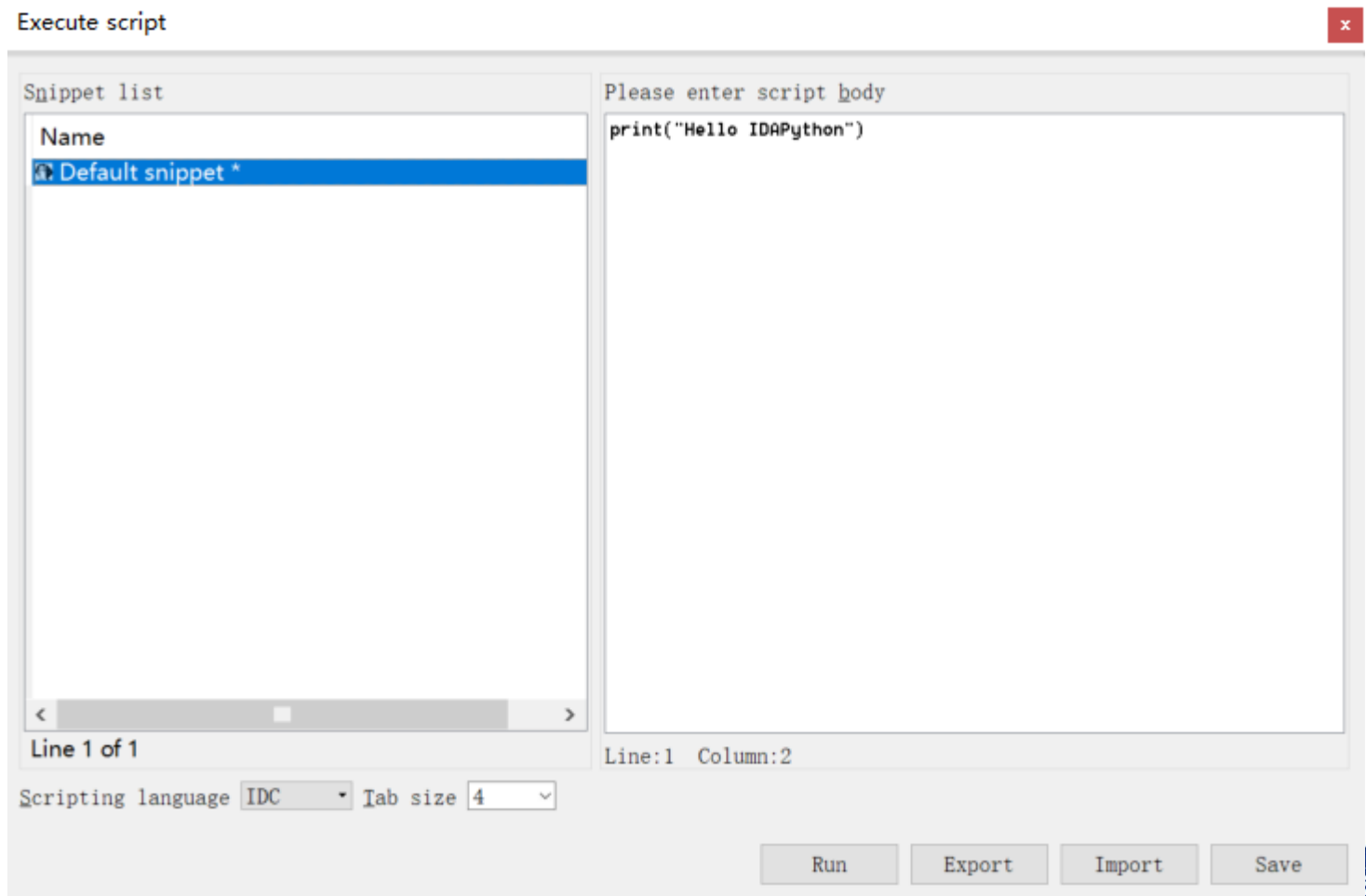
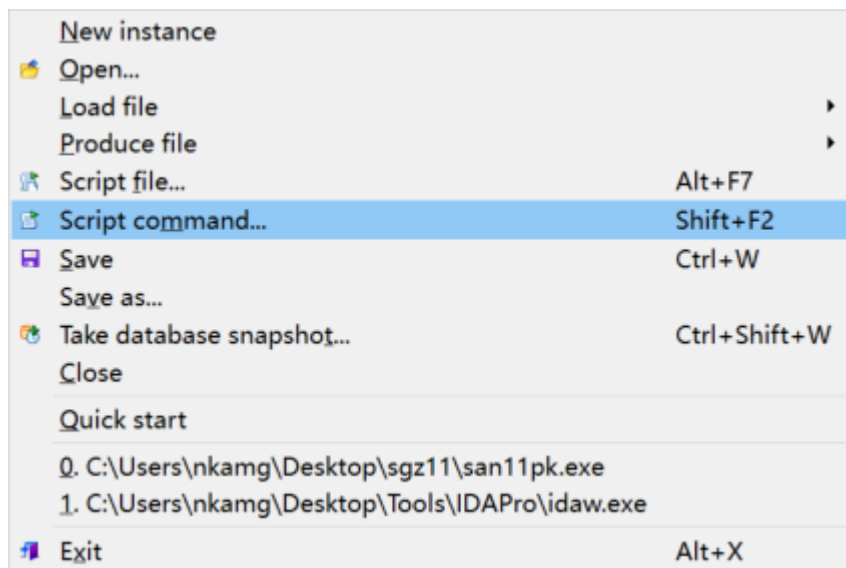


南开大学  
Nankai University

# 运行IDAPython脚本文件



# IDA中直接编写IDAPython脚本





允公允能 日新月异

# IDA基本信息

```
.text:00713CD6 8B 45 E4          mov     eax, [ebp+var_1C]
```

**.text** 是节的名称（section name）

**00713CD6** 是内存地址，十六进制的格式，没有0x前缀

**8B 45 E4** 是指令的机器码

**mov**是指令的助记符（mnemonic）

**eax**是指令的第一个操作数（operand）

**[ebp + var\_1C]**是指令的第二个操作数（operand）





允公允能 日新月异

# IDA基本信息

- IDAPython中，使用ea表示内存地址
  - 获得当前光标所在位置汇编语句的内存地址

- `idc.get_screen_ea()`

```
Python>ea = idc.get_screen_ea()  
Python>print("0x%x %s" % (ea, ea))
```

```
0x713cd6 7421142
```

- `here()`

```
Python>ea = here()
```

```
Python>print("0x%x %s" % (ea, ea))
```

```
0x713cd6 7421142
```

```
.text:00713CD6 8B 45 E4
```

```
mov     eax, [ebp+var_1C]
```





允公允能 日新月异

# IDA基本信息

- 读取节信息

- `idc.get_segm_name(ea)`

```
Python>idc.get_segm_name(ea)
.text
```

- 读取汇编语句

- `idc.generate_disasm_line(ea, 0)`

```
Python>idc.generate_disasm_line(ea, 0)
mov     eax, [ebp+var_1C]
Python>idc.print_insn_mnem(ea)
mov
```

- 读取汇编指令（助记符）

- `idc.print_insn_mnem(ea)`

```
.text:00713CD6 8B 45 E4
```

```
mov     eax, [ebp+var_1C]
```





允公允能 日新月异

# IDA基本信息

- 读取第一个操作符
  - `idc.print_operand(ea,0)`
- 读取第二个操作符
  - `idc.print_operand(ea,1)`

```
Python>idc.print_operand(ea,0)
eax
Python>idc.print_operand(ea,1)
[ebp+var_1C]
```

Python

```
.text:00713CD6 8B 45 E4          mov     eax, [ebp+var_1C]
```





允公允能 日新月异

# 内存地址的检测

- 检测内存地址是否可访问
  - idaapi.BADADDR
  - idc.BADADDR
  - BADADDR

```
.....  
Python>if BADADDR != here(): print("Valid address")  
Valid address
```

Python




南开大学  
Nankai University



# 段信息

Execute script

Snippet list

Name
 Default snippet *

Please enter script body

```

1 import idutils
2
3 for seg in idutils.Segments():
4     print('%s, 0x%x, 0x%x' % (idc.get_segm_name(seg),
5                               idc.get_segm_start(seg), idc.get_segm_end(seg)))

```

Line 1 of 1

Scripting language Python

Tab size 4

Run Export Import

```

HEADER, 0x400000, 0x401000
.text, 0x401000, 0x74e000
.idata, 0x74e000, 0x74e684
.rdata, 0x74e684, 0x89d000
.data, 0x89d000, 0x8e22fc
.idata, 0x8e22fc, 0x8e2304
.data, 0x8e2304, 0x9c62000
stxt774, 0x9c7c000, 0x9c7f000
stxt371, 0x9c7f000, 0x9c83000

```

Python



IDA Python可以读取以下哪些信息？

- ☒ A 当前内存地址
- ☒ B 节 (segment) 信息
- ☒ C 指令助记符
- ☒ D 指令操作数 (operand)



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异

函数



允公允能 日新月异

# 遍历函数

Please enter script body

```
1 import idutils
2
3 for func in idutils.Functions():
4     print("0x%x, %s" % (func, idc.get_func_name(func)))
5
```

Line:4 Column:5

Tab size 4

Run Export Import

c\_name

0x654100, sub\_654100  
0x654150, sub\_654150  
0x6541c0, sub\_6541c0  
0x654210, sub\_654210  
0x654240, sub\_654240  
0x654270, sub\_654270  
0x654310, sub\_654310  
0x654370, sub\_654370  
0x6543e0, sub\_6543E0  
0x654530, sub\_654530  
0x654610, sub\_654610

Python



南开大学  
Nankai University



允公允能 日新月异

# 函数信息

- `idautils.Functions()` 返回IDA识别出的所有函数入口点列表
- `idautils.Functions(start_addr, end_addr)`，可以指定显示某个范围的函数列表
- `idc.get_func_name(ea)` 返回函数名，ea可以是函数内存范围内的任意一个地址





# get\_func()函数

```
.text:0072C243      sub_72C243      proc near      ; DATA XREF: .rdata:off_87F958↓o
.text:0072C243 B8 2C F9 87+      mov     eax, offset off_87F92C ; "CPtrList"
.text:0072C248 C3               ret     retn
.text:0072C248      sub_72C243      endp
.text:0072C248
.text:0072C249
.text:0072C249      ; ===== S U B R O U T I N E =====
.text:0072C249
.text:0072C249      sub_72C249      proc near      ; DATA XREF: .rdata:off_87F990↓o
.text:0072C249 B8 64 F9 87+      mov     eax, offset off_87F964 ; "CPtrArray"
.text:0072C24E C3               ret     retn
.text:0072C24E      sub_72C249      endp
```

```
Python>ea = here()
Python>func = idaapi.get_func(ea)
Python>type(func)
<class 'ida_funcs.func_t'>
Python>print("Start: 0x%x, End: 0x%x" % (func.start_ea, func.end_ea))
Start: 0x72c243, End: 0x72c249
```



允公允能 日新月异

# 函数的属性

```
Python>dir(func)
```

```
['__class__', '__del__', '__delattr__', '__dict__', '__doc__', '__eq__', '__format__', '__get_points__', '__get_regvars__',  
 '__get_tails__', '__getattr__', '__gt__', '__hash__', '__init__', '__lt__', '__module__', '__ne__', '__new__',  
 '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__',  
 '__swig_destroy__', '__weakref__', '_print', 'analyzed_sp', 'argsize', 'clear', 'color', 'compare', 'contains',  
 'does_return', 'empty', 'endEA', 'end_ea', 'extend', 'flags', 'fpd', 'frame', 'frregs', 'frsize', 'intersect', 'is_far',  
 'llabelqty', 'llabels', 'need_prolog_analysis', 'overlaps', 'owner', 'pntqty', 'points', 'referers', 'refqty', 'regargqty',  
 'regargs', 'regvarqty', 'regvars', 'size', 'startEA', 'start_ea', 'tailqty', 'tails', 'this', 'thisown']
```



南开大学  
Nankai University



# get\_next\_func()、get\_prev\_func()

```
.text:0072C243      sub_72C243      proc near          ; DATA XREF: .rdata:off_87F958↓o
.text:0072C243 B8 2C F9 87+      mov     eax, offset off_87F92C ; "CPtrList"
.text:0072C248 C3                retn
.text:0072C248      sub_72C243      endp
.text:0072C248
.text:0072C249
.text:0072C249      ; ===== S U B R O U T I N E =====
.text:0072C249
.text:0072C249      sub_72C249      proc near          ; DATA XREF: .rdata:off_87F990↓o
.text:0072C249 B8 64 F9 87+      mov     eax, offset off_87F964 ; "CPtrArray"
.text:0072C24E C3                retn
.text:0072C24E      sub_72C249      endp
```

```
Python> ea = idc.get_next_func(ea)
```

```
Python> print("0x%x, %s" % (ea, idc.get_func_name(ea)))
```

```
0x72c249, sub_72C249
```

Python





# next\_head() 函数

- 遍历函数内部的汇编指令

Please enter script body

```
1 ea = here()
2 start = idc.get_func_attr(ea, FUNCATTR_START)
3 end = idc.get_func_attr(ea, FUNCATTR_END)
4 cur_addr = start
5 while cur_addr <= end:
6     print("0x%x %s" % (cur_addr,
7         idc.generate_disasm_line(cur_addr, 0)))
8     cur_addr = idc.next_head(cur_addr, end)
```

Line:3 Column:42

Tab size

Run

Export

Import

```
sub_72C243      proc near                ; DATA XREF: .rdata:off_87F958↓o
+              mov     eax, offset off_87F92C ; "CPtrList"
               ret     eax
sub_72C243      endp
```

```
0x72c243 mov     eax, offset off_87F92C; "CPtrList"
0x72c248 ret     eax
```

Python





# 函数属性

- IDA提供了9个函数属性标签

- FUNC\_NORET
- FUNC\_FAR
- FUNC\_LIB
- FUNC\_STATIC
- FUNC\_FRAME
- FUNC\_USERFAR
- FUNC\_HIDDEN
- FUNC\_THUNK
- FUNC\_BOTTOMBP

```
import idutils
for func in idutils.Functions():
    flags = idc.get_func_attr(func,FUNCATTR_FLAGS)
    if flags & FUNC_NORET: print("0x%x FUNC_NORET" %
func)
    if flags & FUNC_FAR: print("0x%x FUNC_FAR" % func)
    if flags & FUNC_LIB: print("0x%x FUNC_LIB" % func)
    if flags & FUNC_STATIC: print("0x%x FUNC_STATIC" %
func)
    if flags & FUNC_FRAME: print("0x%x FUNC_FRAME" %
```

```
0x71bf5f FUNC_HIDDEN
0x71bf5f FUNC_BOTTOMBP
0x71bfe9 FUNC_LIB
0x71bfe9 FUNC_FRAME
0x71bfe9 FUNC_HIDDEN
0x71bfe9 FUNC_BOTTOMBP
0x71c08b FUNC_LIB
0x71c08b FUNC_HIDDEN|
0x71c08b FUNC_BOTTOMBP
0x71c0a4 FUNC_LIB
0x71c0a4 FUNC_HIDDEN
0x71c0a4 FUNC_BOTTOMBP
...
```





# FUNC\_NORET

```
; Attributes: library function noreturn bp-based frame

unknown_libname_1554 proc near          ; CODE XREF: CDataExchange:
                                         ; CWinApp::SetCurrentHandles

var_4          = dword ptr -4

        push    ebp
        mov     ebp, esp
        push    ecx
39+      push    offset unk_899400
        lea     eax, [ebp+var_4]
        push    eax
0+      mov     [ebp+var_4], offset unk_9C5F9C0
E+      call    __CxxThrowException@8 ; _CxxThrowException(x
unknown_libname_1554 endp
```





允公允能 日新月异

# FUNC\_FAR和FUNC\_USERFAR

- 函数的长调用
  - 需要用到段寄存器
  - 很少见到





允公允能 日新月异

# FUNC\_LIB

- 库函数
- 一般不对库函数进行  
逆向分析

0x72c863 FUNC\_LIB ??1CThreadLocalObject@@QAE@XZ  
0x72c881 FUNC\_LIB ?AfxTermLocalData@@YGXPAUHINSTANCE\_@@@H@Z  
0x72c893 FUNC\_LIB ??1CThreadSlotData@@QAE@XZ  
0x72c8ea FUNC\_LIB ?AfxTlsRelease@@YGXXZ  
0x72c91e FUNC\_LIB ?AfxCriticalInit@@YGHXZ  
0x72c942 FUNC\_LIB ?AfxCriticalTerm@@YGXXZ  
0x72c987 FUNC\_LIB ?AfxLockGlobals@@YGXH@Z  
0x72c9ea FUNC\_LIB ?AfxUnlockGlobals@@YGXH@Z  
0x72ca37 FUNC\_LIB ?\_AfxInitDBCS@@YGHXZ  
0x72ca7b FUNC\_LIB ?AfxGetFileName@@YGIPBDPADI@Z  
0x72caaa FUNC\_LIB ?SetCurrentHandles@CWinApp@@QAEXXZ  
0x72cbf8 FUNC\_LIB ?AfxWinInit@@YGHPAUHINSTANCE\_@@@0PADH@Z  
0x72ccd1 FUNC\_LIB ?GetErrorMessage@COleException@@UAEHPADIPAI@.  
0x72cd27 FUNC\_LIB ?AfxThrowOleException@@YGXJ@Z  
0x72cd69 FUNC\_LIB ?InternalRelease@CCmdTarget@@QAEKXZ



南开大学  
Nankai University

什么是静态函数(Static Function)?

正常使用主观题需2.0以上版本雨课堂

作答



南开大学  
Nankai University



允公允能 日新月异

# 静态函数 (Static Function)

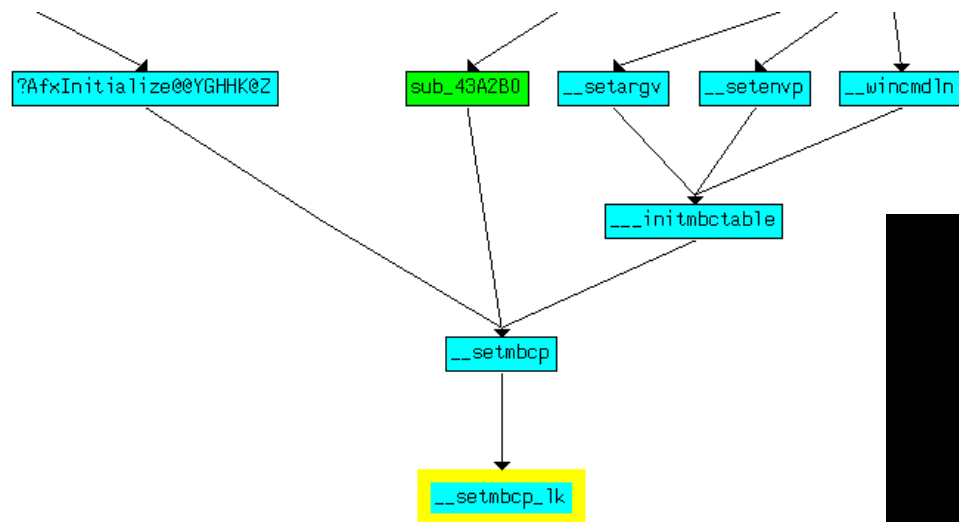
- C/C++中，作用域仅在一个文件中。
  - 不同的文件，可以定义相同名字的静态函数





# FUNC\_STATIC静态函数

- 函数的调用关系
- Xrefs to



```

; ===== SUBROUTINE =====
; Attributes: library function static bp-based frame
; int __cdecl __setmbcp_1k(UINT CodePage)
__setmbcp_1k    proc near                ; CODE XREF: __setmbcp+9F↓p
  
```

```

; ===== SUBROUTINE =====
; Attributes: library function bp-based frame
; int __cdecl __setmbcp(int)
__setmbcp      proc near                ; CODE XREF: sub_43A2B0+88↑p
                                          ; ___initmbctable+B↓p ...
  
```



Stack pointer和Frame pointer有什么区别?

正常使用主观题需2.0以上版本雨课堂

作答

# FUNC\_FRAME

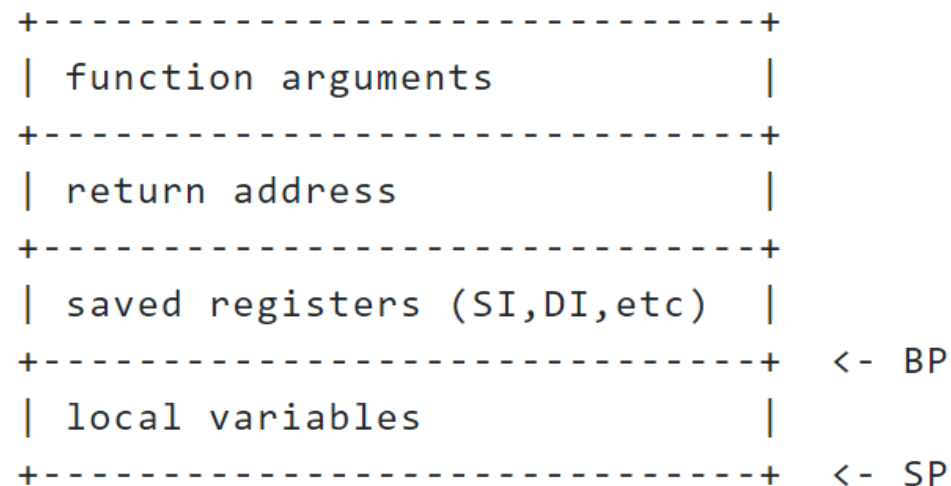
- 函数里面是否使用了帧指针（Frame Pointer）
- EBP

```
; ===== SUBROUTINE =====  
; Attributes: library function bp-based frame  
  
; int __cdecl _setmbcp(int)  
__setmbcp      proc near                ; CODE XREF: sub_43A2B0+88↑p  
                                           ; ___initmbctable+B↓p ...  
  
var_24          = dword ptr -24h  
var_20          = dword ptr -20h  
var_1C          = dword ptr -1Ch  
ms_exc          = CPPEH_RECORD ptr -18h  
CodePage        = dword ptr 8  
  
; __unwind { // __SEH_prolog  
    push 14h  
    push offset stru_87FED0  
    call __SEH_prolog  
    or [ebp+var_20], 0FFFFFFFFh  
    push 0Dh  
    call __lock  
    pop ecx  
    xor edi, edi
```



# FUNC\_BOTTOMBP

- 函数的帧指针（Frame Pointer）等于栈指针（Stack Pointer）
- “BP equal to SP” means that the frame pointer points to the **bottom of the stack**. It is usually used for the processors who set up the stack frame with EBP and ESP both pointing to the bottom of the frame (**currently MC6816, M32R**).”





允公允能 日新月异

# FUNC\_HIDDEN

- 隐藏函数
- Hidden and needs to be expanded to **view**.
- If we were to go to an address of a function that is marked as hidden it would **automatically** be expanded.

0x72cf99 FUNC\_HIDDEN  
0x7374e8 FUNC\_HIDDEN  
0x737708 FUNC\_HIDDEN  
0x740075 FUNC\_HIDDEN  
0x74d700 FUNC\_HIDDEN  
0x74d710 FUNC\_HIDDEN  
0x74d740 FUNC\_HIDDEN

Python



南开大学  
Nankai University



允公允能 日新月异

# FUNC\_THUNK

- 跳转到其它函数的函数
- “They are simple functions that  
jump to another function.”

```
; ===== SUBROUTINE =====  
; Attributes: thunk  
  
; void __stdcall RtlUnwind(PVOID TargetFrame, PVOID TargetIp,  
RtlUnwind      proc near      ; CODE XREF: _UnwindN  
; __global_unwind2+13  
  
TargetFrame     = dword ptr  4  
TargetIp        = dword ptr  8  
ExceptionRecord = dword ptr 0Ch  
ReturnValue     = dword ptr 10h  
  
                jmp     ds:__imp_RtlUnwind  
RtlUnwind      endp
```





允公允能 日新月异

# 添加函数

```
.text:00407DC1  
.text:00407DC1      mov     ebp, esp  
.text:00407DC3      sub     esp, 48h  
.text:00407DC6      push    ebx|
```

- `idc.add_func(0x00407DC1, 0x00407E90)`





允公允能 日新月异

# 获得函数的参数

- `idaapi.get_arg_addrs(ea)`

```
.text:000000014001B5FF      js      loc_14001B72B
.text:000000014001B605      mov     rcx, cs:qword_14002D368 ; hWnd
.text:000000014001B60C      xor     r9d, r9d          ; lParam
.text:000000014001B60F      xor     r8d, r8d          ; wParam
.text:000000014001B612      mov     edx, 0BDh ; '½' ; Msg
.text:000000014001B617      call    cs:SendMessageW
.text:000000014001B61D      xor     esi, esi
```

```
Python>ea = 0x00014001B617
```

```
Python>idaapi.get_arg_addrs(ea)
```

```
[0x14001b605, 0x14001b612, 0x14001b60f, 0x14001b60c]
```

# 获得函数的汇编指令

- `idautils.FuncItems(ea)` 返回函数中指令的地址

```
dism_addr = list(idautils.FuncItems(here()))
for line in dism_addr:
    print("0x%x %s" % (line, idc.generate_disasm_line(line,
0)))
```

```
0x42d995 jnz    short locret_42D9B2|
0x42d997 mov    [esp+arg_0], ecx
0x42d99b jmp    sub_42CFB0
0x42d9a0 mov    [esp+arg_0], ecx; float
0x42d9a4 jmp    sub_42C880
0x42d9a9 mov    [esp+arg_0], ecx
0x42d9ad jmp    sub_42D4C0
0x42d9b2 retn
```

Python







允公允能 日新月异

# 相关函数

- `idautils.Functions()` 获得程序中所有的函数列表（函数入口点地址）
- `idc.get_func_attr(ea, FUNCATTR_FLAGS)`. 获得函数属性
- `idautils.FuncItems(ea)` 获得函数中的所有指令（指令的地址）
- `idc.print_insn_mnem(ea)` 获得指令的助记符（Mnemonic）
- `idc.get_operand_type(ea, n)` 获得操作数的类型
  - `op_t.type`



## 概述下面这段代码的功能？

```
for func in idutils.Functions():
    flags = idc.get_func_attr(func, FUNCATTR_FLAGS)
    if flags & FUNC_LIB or flags & FUNC_THUNK:
        continue
    dism_addr = list(idutils.FuncItems(func))
    for line in dism_addr:
        m = idc.print_insn_mnem(line)
        if m == 'call' or m == 'jmp':
            op = idc.get_operand_type(line, 0)
            if op == o_reg:
                print("0x%x %s" % (line, idc.generate_disasm_line(line, 0)))
```



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异

操作数



允公允能 日新月异

# 操作数

- 操作数的类型
  - `idc.get_operand_type(ea,n)`
  - `ea`是指令地址
  - `n`是操作数的索引
  - `o_void`, `o_reg`, `o_mem`, `o_phrase`, `o_displ`, `o_imm`, `o_far`, `o_near`





允公允能 日新月异

# 操作数类型

- o\_void
  - 指令没有操作数

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0xa09166 retn  
Python>print(idc.get_operand_type(ea,0))  
0
```





允公允能 日新月异

# 操作数类型

- o\_reg
  - 操作数是寄存器

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0xa09163 pop      edi  
Python>print(idc.get_operand_type(ea,0))  
1
```





允公允能 日新月异

# 操作数类型

- o\_mem
  - 操作数是内存地址

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0xa05d86 cmp      ds:dword_A152B8, 0
```

```
Python>print(idc.get_operand_type(ea,0))  
2
```





允公允能 日新月异

# 操作数类型

- o\_phrase
  - 操作数是寄存器的表达式

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0x1000b8c2 mov      [edi+ecx], eax  
Python>print(idc.get_operand_type(ea,0))  
3
```







允公允能 日新月异

# 操作数类型

- o\_displ
  - 操作数是寄存器加位移数值（displacement value）

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0xa05dc1 mov     eax, [edi+18h]  
Python>print(idc.get_operand_type(ea,1))  
4
```





# 操作数类型

- o\_imm
  - 操作数是立即数

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))  
0xa05da1 add     esp, 0Ch  
  
Python>print(idc.get_operand_type(ea,1))  
5
```





# 操作数的分析

seg000:00BC1388	push	0Ch
seg000:00BC138A	push	0BC10B8h
seg000:00BC138F	push	[esp+10h+arg_0]
seg000:00BC1393	call	ds:_strnicmp

seg000:00BC1388	push	0Ch
seg000:00BC138A	push	offset aNtoskrnl_exe ; "ntoskrnl.exe"
seg000:00BC138F	push	[esp+10h+arg_0]
seg000:00BC1393	call	ds:_strnicmp

op\_plain\_offset将操作数改成offset



```
min = idc.get_inf_attr(INF_MIN_EA)
max = idc.get_inf_attr(INF_MAX_EA)
# for each known function
for func in idutils.Functions():
    flags = idc.get_func_attr(func, FUNCATTR_FLAGS)
    # skip library & thunk functions
    if flags & FUNC_LIB or flags & FUNC_THUNK:
        continue
    dism_addr = list(idutils.FuncItems(func))
    for curr_addr in dism_addr:
        if idc.get_operand_type(curr_addr, 0) == 5 and \
            (min < idc.get_operand_value(curr_addr, 0) < max):
            idc.OpOff(curr_addr, 0, 0)
        if idc.get_operand_type(curr_addr, 1) == 5 and \
            (min < idc.get_operand_value(curr_addr, 1) < max):
            idc.op_plain_offset(curr_addr, 1, 0)|
```



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异

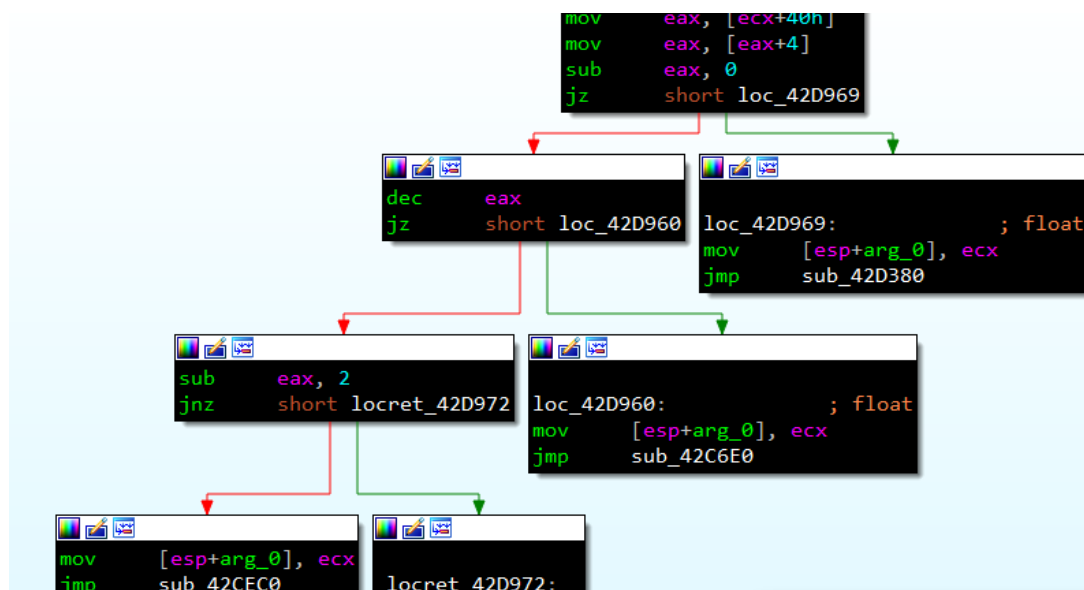
基本块

静态分析中，什么是基本块（Basic Block）？

作答

# 基本块 (Basic Block)

- 基本块是一段代码序列，该代码序列没有分支，也就是只有一个入口点和一个出口点
  - IDA的图形模式
- 控制流分析Control flow analysis
  - 循环识别
  - 控制流混淆



公允能日新月昇

# XOR加密

```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

argc= dword ptr 4
argv= dword ptr 8
envp= dword ptr 0Ch

push esi
push edi
push 0Ah ; Size
call ds:malloc

mov esi, eax
mov edi, offset str_encrypted
xor eax, eax ; eax = 0
sub edi, esi
pop ecx
```

```
loop:
lea edx, [eax+esi]
mov cl, [edi+edx]
xor cl, ds:b_key ; cl = 0
inc eax
mov [edx], cl
cmp eax, 9 ; index
jnb short loop
```

```
push esi ; Memory
call ds:free

add esp, 0Ch
xor eax, eax ; eax = 0
pop edi
pop esi
retn

_main endp
```

```
.text:0040104A loop: ; CODE
.text:0040104A lea edx, [eax+esi]
.text:0040104D mov cl, [edi+edx]
.text:00401050 xor cl, ds:b_key ; cl = 0
.text:00401056 inc eax
.text:00401057 mov [edx], cl
.text:00401059 cmp eax, 9 ; index
.text:0040105C jnb short loop
```







```
ea = 0x0401050 XOR
```

```
f = idaapi.get_func(ea)
```

```
fc = idaapi.FlowChart(f, flags=idaapi.FC_PREDS)
```

```
for block in fc:
```

```
    print("ID: %i Start: 0x%x End: 0x%x" % (block.id, block.start_ea,  
block.end_ea))
```

```
    if block.start_ea <= ea < block.end_ea:
```

```
        print("    Basic Block selected")
```

```
    successor = block.succs()
```

```
    for addr in successor:
```

```
        print("    Successor: 0x%x" % addr.start_ea)
```

```
    pre = block.preds()
```

```
    for addr in pre:
```

```
        print("    Predecessor: 0x%x" % addr.end_ea)
```

```
    if ida_gdl.is_ret_block(block.type):
```

```
        print("    Return Block")
```



ID: 0 Start: 0x401034 End: 0x40104a  
Successor: 0x40104a  
ID: 1 Start: 0x40104a End: 0x40105e  
Basic Block selected  
Successor: 0x40105e  
Successor: 0x40104a|  
Predecessor: 0x40104a  
Predecessor: 0x40105e  
ID: 2 Start: 0x40105e End: 0x40107c  
Predecessor: 0x40105e  
Return Block



在一个基本块的指令序列中，除了最后一条指令，对于中间的指令，以下哪些指令是不会出现的？

☒ A jmp

☒ B je

☒ C call

☒ D ret

提交



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异

交叉引用

# 交叉引用

- 交叉引用Xrefs用于分析数据或代码被引用的信息
- WriteFile函数的交叉引用

```
Python>wf_addr = idc.get_name_ea_simple("WriteFile")
Python>print("0x%x %s" % (wf_addr, idc.generate_disasm_line(wf_addr, 0)))
0x1000e1b8 extrn WriteFile:dword
Python>for addr in idutils.CodeRefsTo(wf_addr, 0):\
    print("0x%x %s" % (addr, idc.generate_disasm_line(addr, 0)))
0x10004932 call     ds:WriteFile
0x10005c38 call     ds:WriteFile
0x10007458 call     ds:WriteFile
```



允公允能 日新月异

# 代码交叉引用

- `idc.get_name_ea_simple(str)` 返回API函数的地址
- `idautils.CodeRefsTo(ea, flow)`.返回代码交叉引用的地址
  - `CodeRefsTo`返回调用该函数的地址
  - `CodeRefsFrom`返回该函数调用的函数地址





允公允能 日新月异

# 数据交叉引用

- `idautils.DataRefsTo(ea)`.返回引用指定位置数据的地址

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))
```

```
0x1000e3ec db 'vnc32',0
```

```
Python>for addr in idautils.DataRefsTo(ea):\
```

```
    print("0x%x %s" % (addr, idc.generate_disasm_line(addr, 0)))
```

```
0x100038ac push    offset aVnc32          ; "vnc32"
```







允公允能 日新月异

# 数据交叉引用

- `idautils.DataRefsFrom(ea)`.返回被引用数据的地址

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))
0x100038ac push      offset aVnc32          ; "vnc32"
Python>for addr in idautils.DataRefsFrom(ea):\
    print("0x%x %s" % (addr, idc.generate_disasm_line(addr, 0)))
0x1000e3ec db  'vnc32',0
```







# 交叉引用

- `idautils.XrefsFrom(ea, flags=0)`
- `idautils.XrefsTo(ea, flags=0)`

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))
0x1000eee0 unicode 0, <Path>,0

Python>for xref in idautils.XrefsTo(ea, 1):
    print("%i %s 0x%x 0x%x %i" % (xref.type, idautils.XrefTypeName(xref.type),
xref.frm, xref.to, xref.iscode))

Python>
1 Data_Offset 0x1000ac0d 0x1000eee0 0

Python>>print("0x%x %s" % (xref.frm, idc.generate_disasm_line(xref.frm, 0)))
0x1000ac0d push      offset KeyName          ; "Path"
```



允公允能 日新月异

# 交叉引用的类型

```
0   = 'Data_Unknown'
1   = 'Data_Offset'
2   = 'Data_Write'
3   = 'Data_Read'
4   = 'Data_Text'
5   = 'Data_Informational'
16  = 'Code_Far_Call'
17  = 'Code_Near_Call'
```

```
18  = 'Code_Far_Jump'
19  = 'Code_Near_Jump'
20  = 'Code_User'
21  = 'Ordinary_Flow'
```





允公允能 日新月异

# 多次引用

```
Python>print("0x%x %s" % (ea, idc.generate_disasm_line(ea, 0)))
0xa21138 extrn GetProcessHeap:dword

Python> for xref in idutils.XrefsTo(ea, 1):
    print("%i %s 0x%x 0x%x %i" % (xref.type, idutils.XrefTypeName(xref.type),
xref.frm, xref.to, xref.iscode))

Python>
17 Code_Near_Call 0xa143b0 0xa21138 1
17 Code_Near_Call 0xa1bb1b 0xa21138 1
3 Data_Read 0xa143b0 0xa21138 0
3 Data_Read 0xa1bb1b 0xa21138 0

Python>print(idc.generate_disasm_line(0xa143b0, 0))
call    ds:GetProcessHeap
```





允公允能 日新月异

# 多次引用

```
def get_to_xrefs(ea):  
    xref_set = set([])  
    for xref in idutils.XrefsTo(ea, 1):  
        xref_set.add(xref.frm)  
    return xref_set  
  
def get_frm_xrefs(ea):  
    xref_set = set([])  
    for xref in idutils.XrefsFrom(ea, 1):  
        xref_set.add(xref.to)  
    return xref_set
```





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异

搜索



允公允能 日新月异

# 搜索Searching

- `ida_search.find_binary(start, end, searchstr, radiux, sflag)`
  - `start`和`end`确定了搜索的范围
  - `searchstr`是搜索的内容
  - `radiux`处理器模式（processor modules）（Chapter 19 of Chris Eagle's The IDA Pro Book. ）
  - `sflag`设置搜索的方向或者条件



南开大学  
Nankai University



允公允能 日新月异

# sflag

```
SEARCH_UP = 0
SEARCH_DOWN = 1
SEARCH_NEXT = 2
SEARCH_CASE = 4
SEARCH_REGEX = 8
SEARCH_NOBRK = 16
SEARCH_NOSHOW = 32
SEARCH_IDENT = 128
SEARCH_BRK = 256
```







允公允能 日新月异

# sflag

- SEARCH\_UP 和 SEARCH\_DOWN 设置搜索的方向
- SEARCH\_NEXT 搜索下一个匹配对象
- SEARCH\_CASE 是否区分大小写
- SEARCH\_NOSHOW 不显示搜索过程
- IDAPython默认同时搜索Unicode和ASCII两种字符集





```
pattern = '55 8B EC'
addr = idc.get_inf_attr(INF_MIN_EA)
for x in range(0, 5):
    addr = ida_search.find_binary(addr, idc.BADADDR, pattern,
16,ida_search.SEARCH_DOWN)
    if addr != idc.BADADDR:
        print("0x%x %s" % (addr, idc.generate_disasm_line(addr, 0)))
```

Python>

0x401000 push ebp

0x401000 push ebp

0x401000 push ebp

0x401000 push ebp

0x401000 push ebp

总是出现同一个地址

```
pattern = '55 8B EC'
addr = idc.get_inf_attr(INF_MIN_EA)
for x in range(0, 5):
    addr = ida_search.find_binary(addr, idc.BADADDR, pattern, 16,
    ida_search.SEARCH_NEXT|ida_search.SEARCH_DOWN)
    if addr != idc.BADADDR:
        print("0x%x %s" % (addr, idc.generate_disasm_line(addr, 0)))
```

Python>

0x401000 push ebp

0x401040 push ebp

0x401070 push ebp

0x4010e0 push ebp

0x401150 push ebp



允公允能 日新月异

# 搜索文本find\_text

`ida_search.find_text(ea, y, x, searchstr, sflag)`  
y和x设置为0

```
Python>cur_addr = idc.get_inf_attr(INF_MIN_EA)
for x in range(0, 5):
    cur_addr = ida_search.find_text(cur_addr, 0, 0, "Accept",
ida_search.SEARCH_DOWN)
    if addr == idc.BADADDR:
        break
    print("0x%x %s" % (cur_addr, idc.generate_disasm_line(cur_addr, 0)))
    cur_addr = idc.next_head(cur_addr)
Python>
0x40da72 push    offset aAcceptEncoding; "Accept-Encoding:\n"
0x40face push    offset aHttp1_1Accept; " HTTP/1.1\r\nAccept: /* \r\n "
0x40fadf push    offset aAcceptLanguage; "Accept-Language: ru \r\n"
...
```



允公允能 日新月异

# 搜索立即数

- **ida\_search.find\_imm(ea, flag, value)**

```
Python>addr = ida_search.find_imm(get_inf_attr(INF_MIN_EA), SEARCH_DOWN, 0x343FD)
Python>addr
[268453092, 0]
Python>print("0x%x %s %x" % (addr[0], idc.generate_disasm_line(addr[0], 0),
addr[1]))
0x100044e4 imul    eax, 343FDh 0
```





允公允能 日新月异

# 实验

- 完成实验 Lab5





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



恶意代码分析与防治技术

## 第6章 IDA Python

王志

zwang@nankai.edu.cn

南开大学 网络空间安全学院

2023-2024学年