



南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



# Practical Malware Analysis

## Chapter 5: IDA Pro

王志

zwang@nankai.edu.cn

updated on Oct. 17<sup>th</sup> 2021

College of Cyber Science  
Nankai University  
2021/2022



允公允能 日新月异

# Agenda

- IDA Pro
- Useful Windows for Analysis
- Navigating IDA Pro
- Using Cross-References
- Analyzing Functions
- Using Graphing Options
- Enhancing Disassembly





允公允能 日新月异

# Practical Malware Analysis

## Ch 5: IDA Pro



南开大学  
Nankai University



允公允能 日新月异

- **Named licenses:**
- **Computer licenses**
- **Floating licenses**



南開大學  
Nankai University



# Named Licenses

- Named licenses are linked to **a specific end user** and may be used on the user's laptop, and two desktop computers. Named licenses are available only to private users. Corporations must use Computer or Floating licenses.





允公允能 日新月异

# Computer Licenses

- Computer licenses are linked to **a specific computer** and may be used by different end-users on that computer provided only one user is active at any time. This license type is suitable for corporations because they are not tied to physical persons and allow for easy license reassignment.





允公允能 日新月异

# Floating Licenses

- Floating licenses (network licenses): can be installed on **unlimited number of computers** (in one organization) but allow only a limited number of simultaneously running copies.



南开大学  
Nankai University



# Two Editions

- **IDA Starter:** supports more than 20 processor families, including the popular **x86** and **ARM** processors
- **IDA Professional:** supports more than 50 processor families and adds support for **64-bit files** (including Intel x86-64 code)







Target OS: Windows		
IDAPROCW	IDA Pro Computer License [Windows]	1879 USD
IDAPROFW	IDA Pro Floating License [Windows]	2819 USD
IDASTACW	IDA Starter Computer License [Windows]	979 USD
IDASTAFW	IDA Starter Floating License [MS Windows]	1469 USD
HEXARM64FW	ARM64 Decompiler Floating License [Windows]	3944 USD
HEXARM64W	ARM64 Decompiler Fixed License [Windows]	2629 USD
HEXARMFW	ARM32 Decompiler Floating License [Windows]	3944 USD
HEXARMW	ARM32 Decompiler Fixed License [Windows]	2629 USD
HEXPPCFW	PPC Decompiler Floating License [Windows]	3944 USD
HEXPPCW	PPC Decompiler Fixed License [Windows]	2629 USD
HEXX64FW	x64 Decompiler Floating License [Windows]	3944 USD
HEXX64W	x64 Decompiler Fixed License [Windows]	2629 USD
HEXX86FW	x86 Decompiler Floating License [Windows]	3944 USD
HEXX86W	x86 Decompiler Fixed License [Windows]	2629 USD





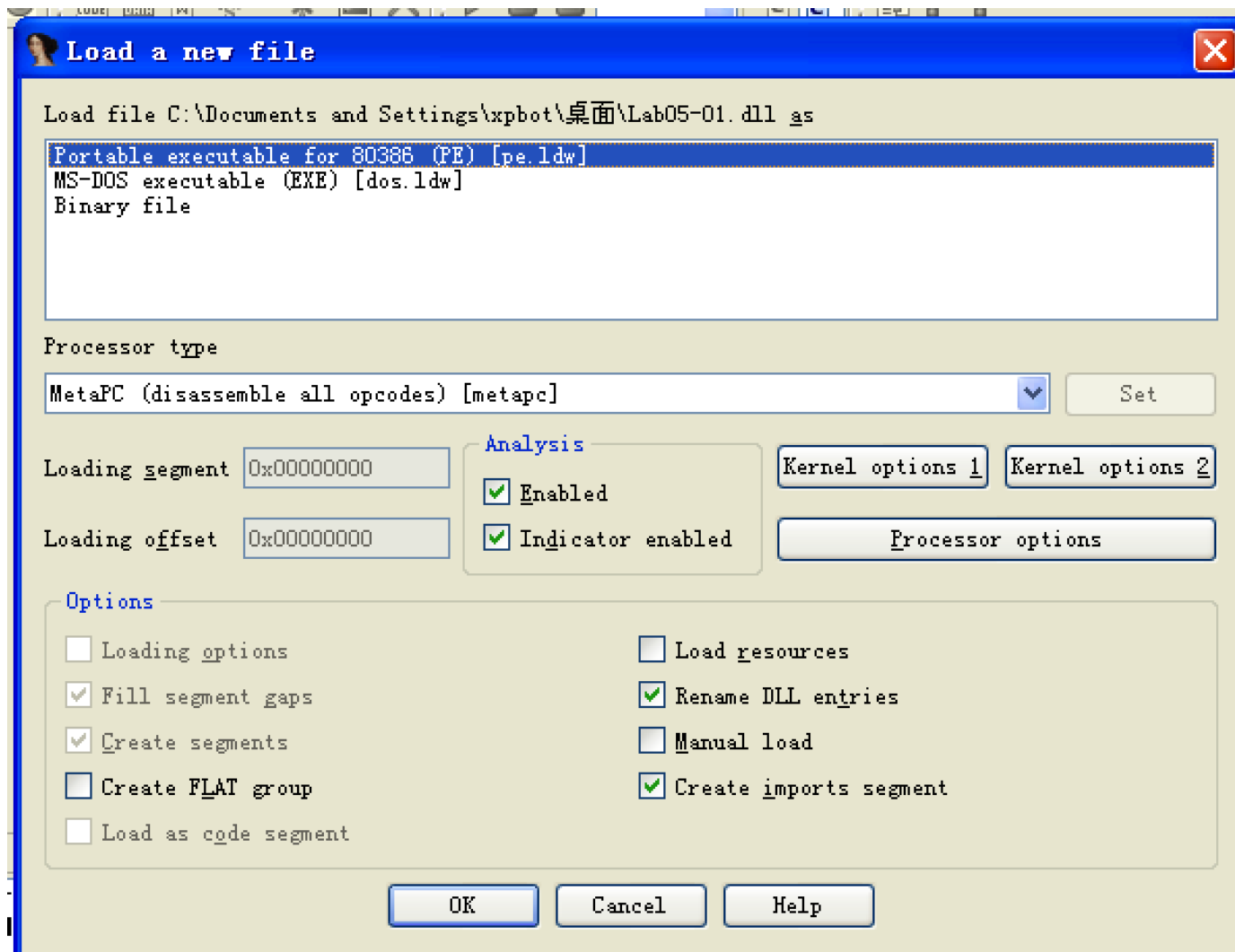
允公允能 日新月异

# IDA Pro

- Function discovery
- Stack analysis
- Local variable identification
- Fast Library Identification and Recognition Technology (FLIRT)
  - Recognize and label library code

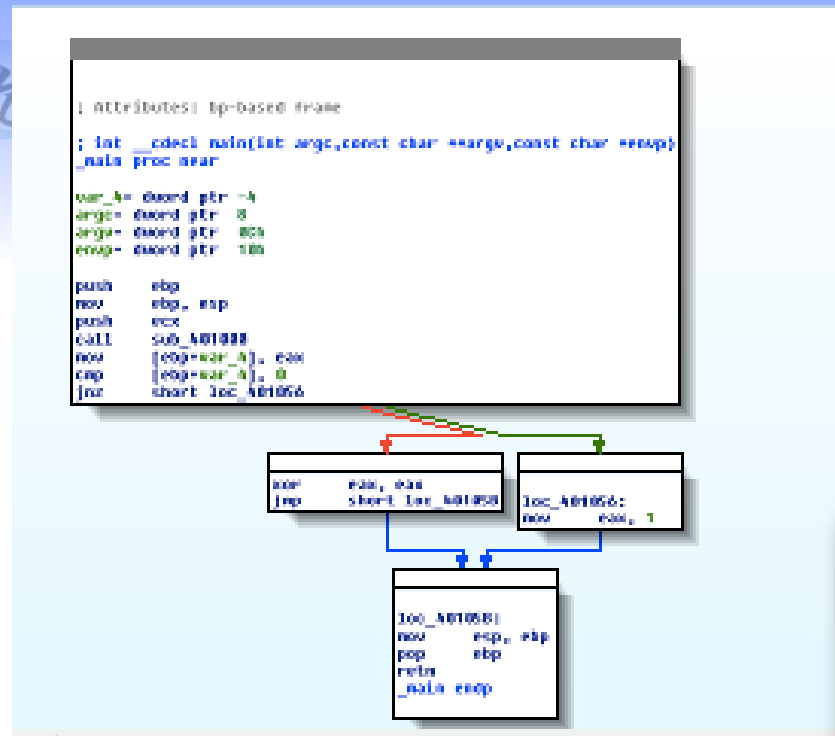


# Loading an Executable



# Graph and Text Mode

- Spacebar switches mode



The screenshot shows the IDA View-A window. The left pane displays a list of memory addresses from `.text:00401040` to `.text:00401043`. The right pane shows the corresponding assembly code. The code is as follows:

```
.text:00401040 ; Attributes: bp-based frame
.text:00401040
.text:00401040 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401040 _main proc near ; CODE XREF: start+AF↓p
.text:00401040
.text:00401040 var_4 = dword ptr -4
.text:00401040 argc = dword ptr 8
.text:00401040 argv = dword ptr 0Ch
.text:00401040 envp = dword ptr 10h
.text:00401040
* .text:00401040 push ebp
* .text:00401041 mov ebp, esp
* .text:00401043 push ecx
```



# Default Graph Mode Display

```

; Attributes: bp-based frame

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

var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
push    ecx
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401056

```





允公允能 日新月异

# Function

- C++ function definition

```
return_type function_name( parameter list ) {  
  
    body of the function  
  
}
```





允公允能 日新月异

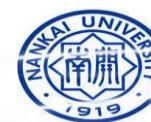
# Calling Convention

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
; CODE XREF: start+AF↓p
_main      proc near

var_4      = dword ptr -4
argc       = dword ptr  8
argv       = dword ptr 0Ch
envp       = dword ptr 10h

        push    ebp
        mov     ebp, esp
```





# Calling Convention

- On x86 platforms, all **arguments** are widened to **32 bits** when they are passed.
- **Return values** are also widened to **32 bits** and returned in the **EAX** register







# Calling Convention

- Specify conventions for **passing arguments** and **return values** between **functions** and **callers**.
- VC compiler supported calling conventions
  - `__cdecl`
  - `__stdcall`





# Calling Convention

- **\_\_cdecl** is the default calling convention for **C and C++** programs

Element	Implementation
Argument-passing order	Right to left.
Stack-maintenance responsibility	Calling function pops the arguments from the stack.





# Calling Convention

- The **\_\_stdcall** calling convention is used to call **Win32 API** functions.

Element	Implementation
Argument-passing order	Right to left.
Stack-maintenance responsibility	Called function pops its own arguments from the stack.



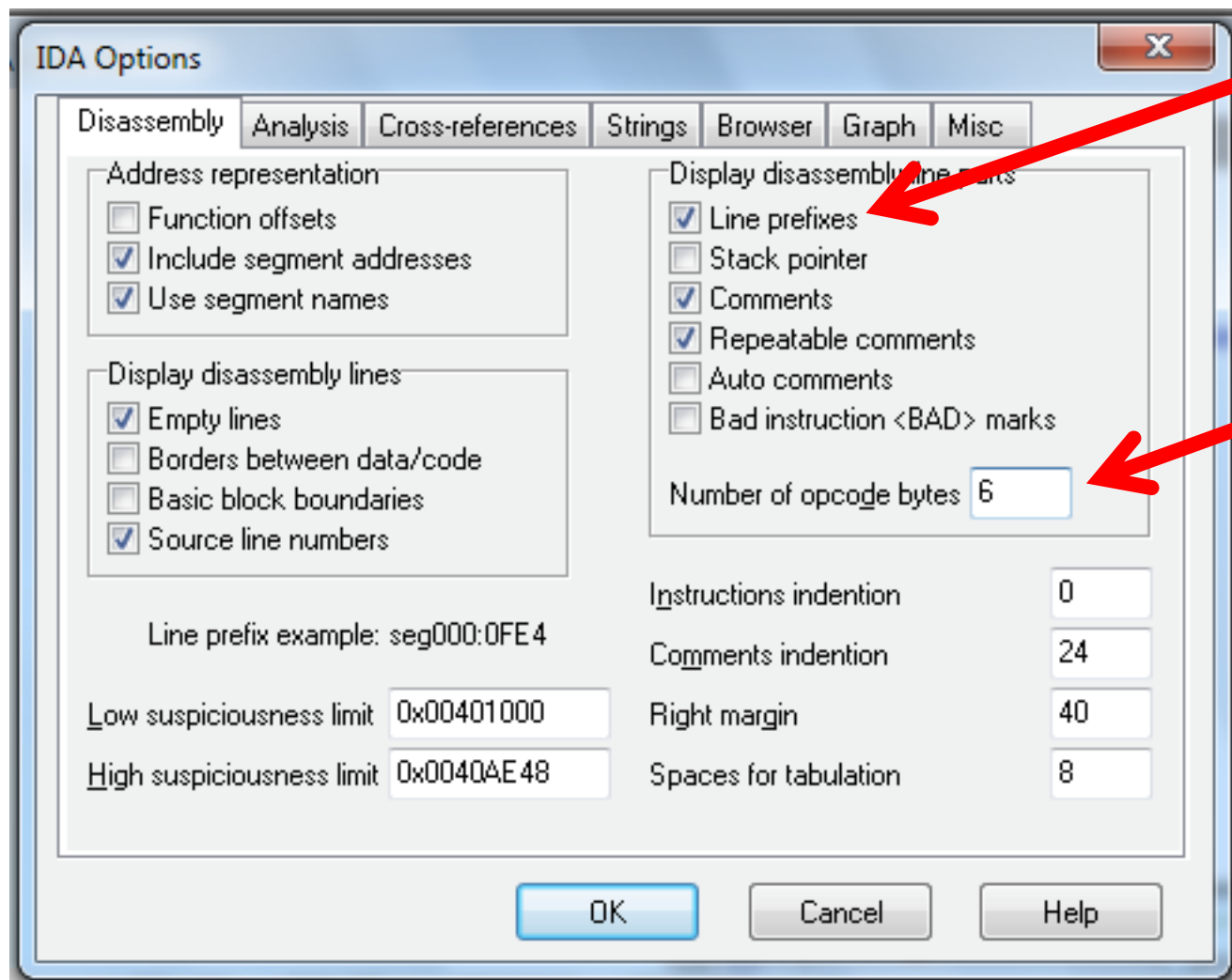


# Calling Convention

```
push    0          ; LPBINDSTATUSCALLBACK
push    0          ; DWORD
push    offset aCEmpdownload_e ; "c:\tempdownload.exe"
mov     eax, [ebp+var_4]
mov     ecx, [eax]
push    ecx        ; LPCSTR
push    0          ; LPUNKNOWN
call    URLDownloadToFileA
mov     esp, ebp
pop     ebp
ret
endp
```



# Config



# Better Graph Mode View

```
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401000
text:00401001
text:00401003
text:00401006
text:00401008
text:0040100D
text:00401010

; int __stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
_WinMain@16 proc near
    var_8 = dword ptr -8
    var_4 = dword ptr -4
    hInstance = dword ptr 8
    hPrevInstance = dword ptr 0Ch
    lpCmdLine = dword ptr 10h
    nShowCmd = dword ptr 14h

    push ebp
    mov ebp, esp
    sub esp, 8
    push 4
    call ???@YAPAXI@Z
    add esp, 4
    mov
```





# Arrows

- Colors

- Red Conditional jump not taken

- Green Conditional jump taken

- Blue Unconditional jump

- Direction

- Up Loop

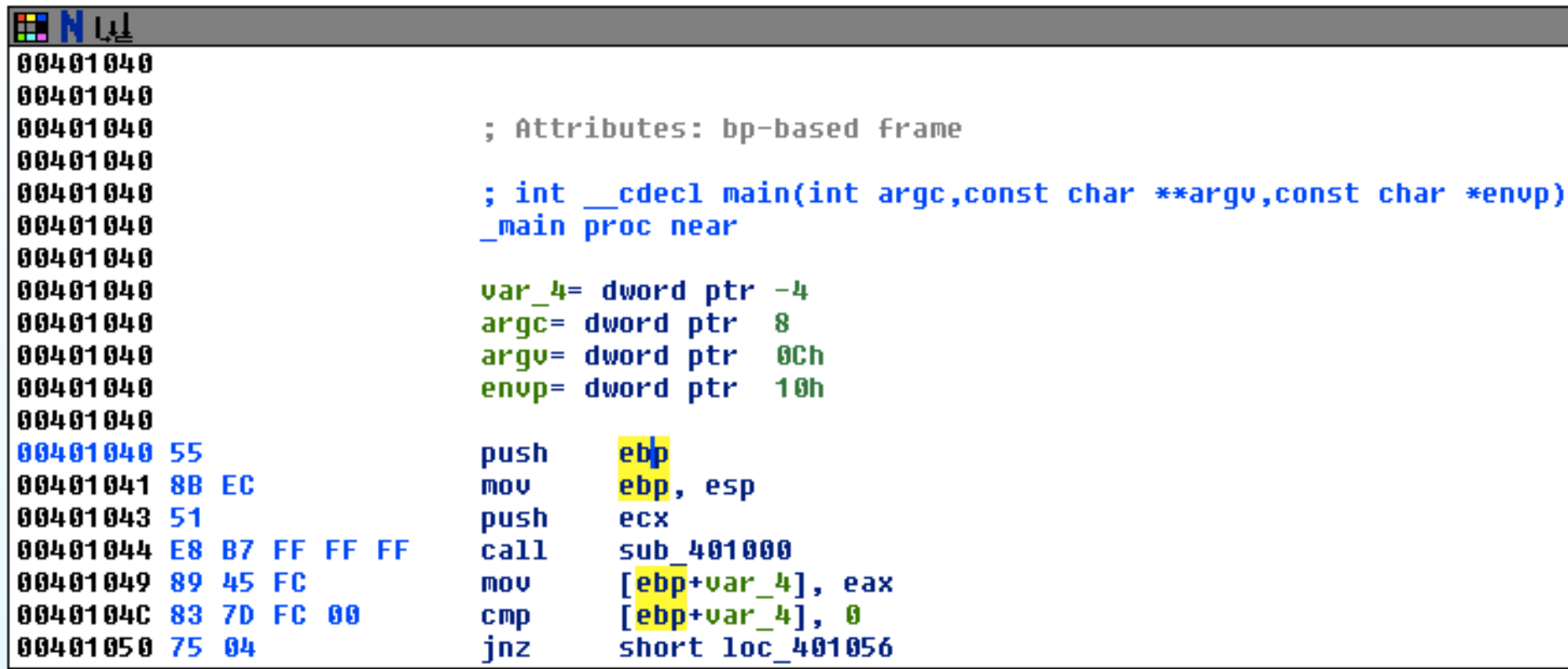






# Highlighting

- Highlighting text in graph mode highlights every instance of that text



```

00401040
00401040
00401040      ; Attributes: bp-based frame
00401040
00401040      ; int __cdecl main(int argc,const char **argv,const char *envp)
00401040      _main proc near
00401040
00401040      var_4= dword ptr -4
00401040      argc= dword ptr  8
00401040      argv= dword ptr  0Ch
00401040      envp= dword ptr  10h
00401040
00401040 55          push     ebp
00401041 8B EC      mov      ebp, esp
00401043 51          push     ecx
00401044 E8 B7 FF FF FF  call     sub_401000
00401049 89 45 FC      mov      [ebp+var_4], eax
0040104C 83 7D FC 00    cmp      [ebp+var_4], 0
00401050 75 04         jnz      short loc_401056
  
```



# Text Mode

Arrows

Solid = Unconditional

Dashed = Conditional

Up = Loop

Section

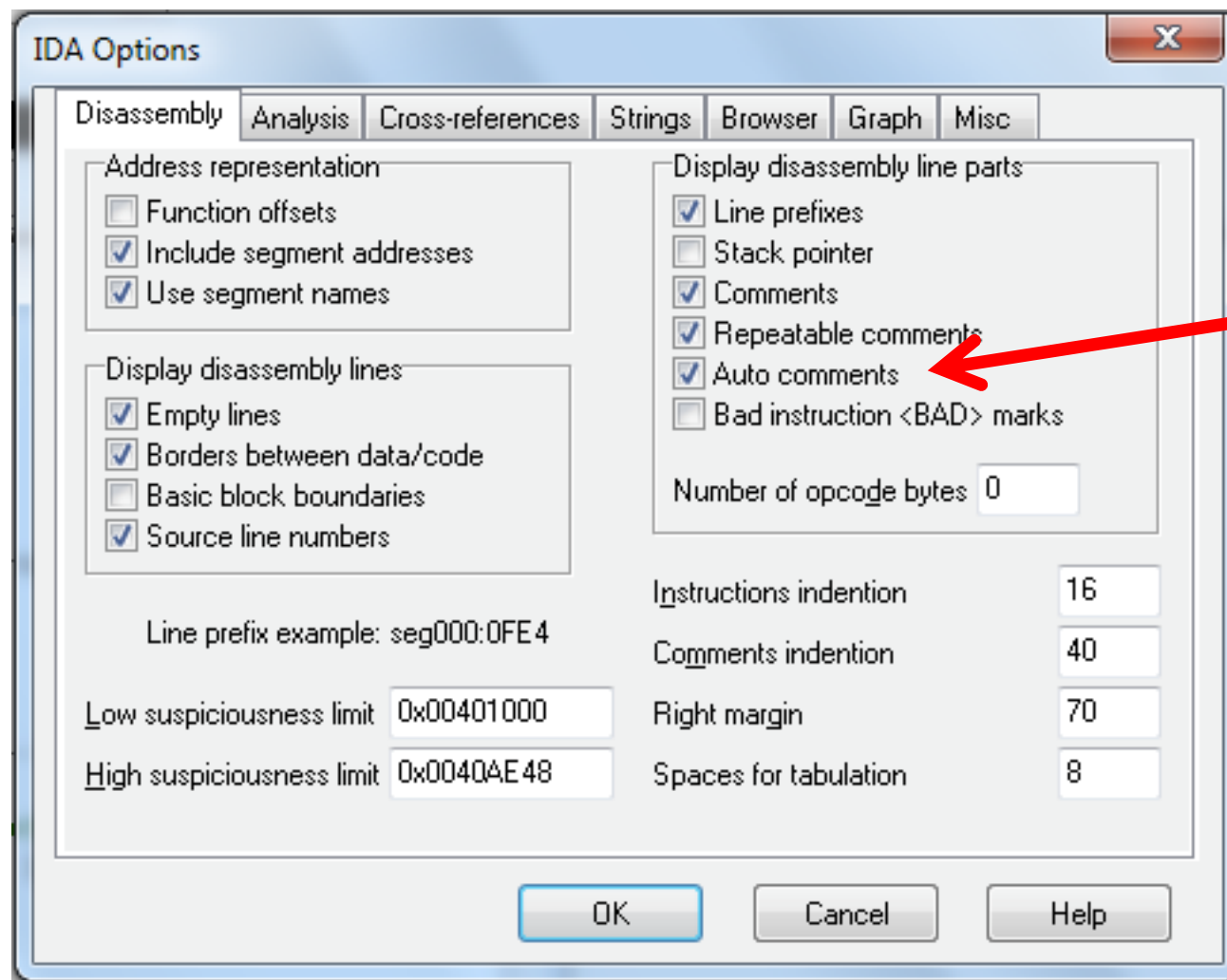
Address

Comment  
Generated by  
IDA Pro

```
.text:00401015      jz      short loc_40102B
.text:00401017      push     offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C      call    sub_40105F
.text:00401021      add     esp, 4
.text:00401024      mov     eax, 1
.text:00401029      jmp     short loc_40103A
.text:0040102B      ; -----
.text:0040102B      loc_40102B: ; CODE XREF: sub_401000+151j
.text:0040102B      push     offset aError1_1NoInte ; "Error 1.1: No Internet\n"
.text:00401030      call    sub_40105F
.text:00401035      add     esp, 4
.text:00401038      xor     eax, eax
.text:0040103A      loc_40103A: ; CODE XREF: sub_401000+291j
.text:0040103A      mov     esp, ebp
.text:0040103C      pop     ebp
```



# Options, General



# Adds Comments to Each Instruction

•	.text:00401015	jz	short loc_40102B ; Jump if Zero (ZF=1)
•	.text:00401017	push	offset aSuccessInterne ; "Success: Internet Connection\n"
•	.text:0040101C	call	sub_40105F ; Call Procedure
•	.text:00401021	add	esp, 4 ; Add
•	.text:00401024	mov	eax, 1
•	.text:00401029	jmp	short loc_40103A ; Jump
	.text:0040102B ; -----		
	.text:0040102B		
	.text:0040102B loc_40102B:		; CODE XREF: sub_401000+15↑j
•	.text:0040102B	push	offset aError1_1NoInte ; "Error 1.1: No Internet\n"
•	.text:00401030	call	sub_40105F ; Call Procedure
•	.text:00401035	add	esp, 4 ; Add
•	.text:00401038	xor	eax, eax ; Logical Exclusive OR
	.text:0040103A		
	.text:0040103A loc_40103A:		; CODE XREF: sub_401000+29↑j
•	.text:0040103A	mov	esp, ebp
•	.text:0040103C	pop	ebp



南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異

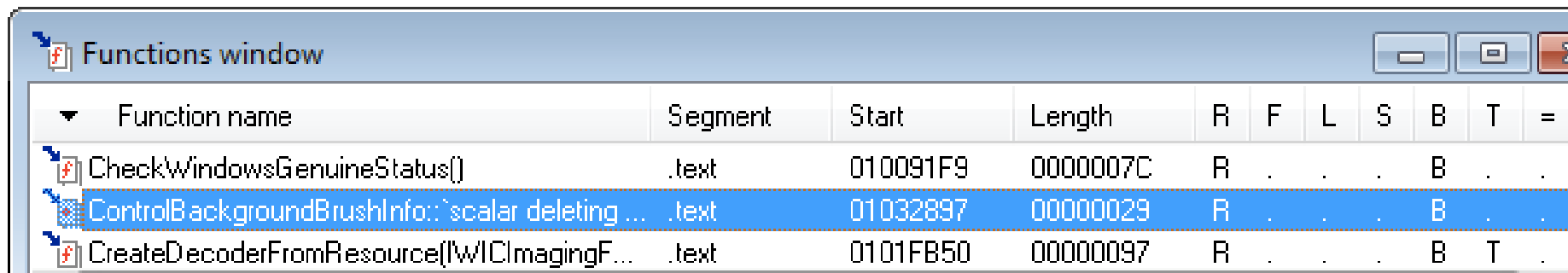


# Useful Windows for Analysis



# Functions

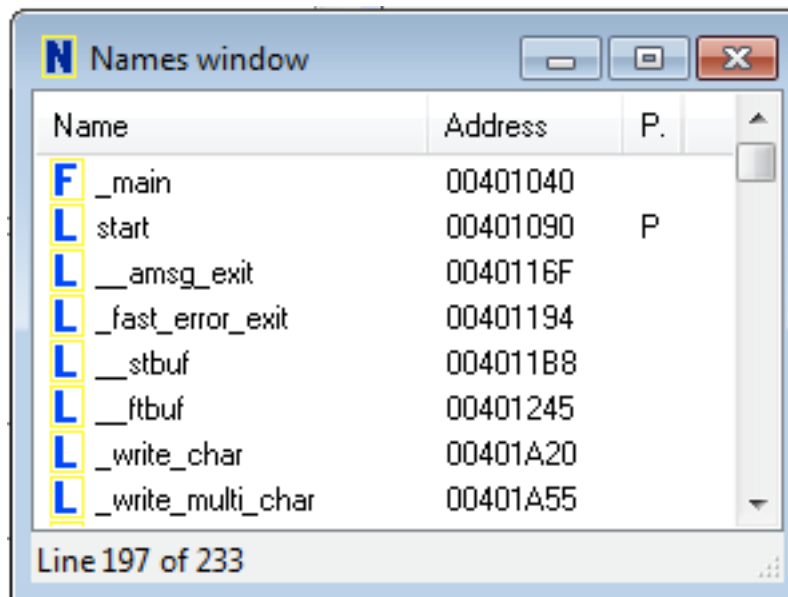
- Shows each function, length, and flags
  - L = Library functions
- Sortable



Function name	Segment	Start	Length	R	F	L	S	B	T	=
CheckWindowsGenuineStatus()	.text	010091F9	0000007C	R	.	.	.	B	.	.
ControlBackgroundBrushInfo::`scalar deleting ...	.text	01032897	00000029	R	.	.	.	B	.	.
CreateDecoderFromResource(IWICImagingF...	.text	0101FB50	00000097	R	.	.	.	B	T	.

# Names Window

- Every address with a name
  - Functions, named code, named data, strings





允公允能 日新月异

# Strings

Strings window

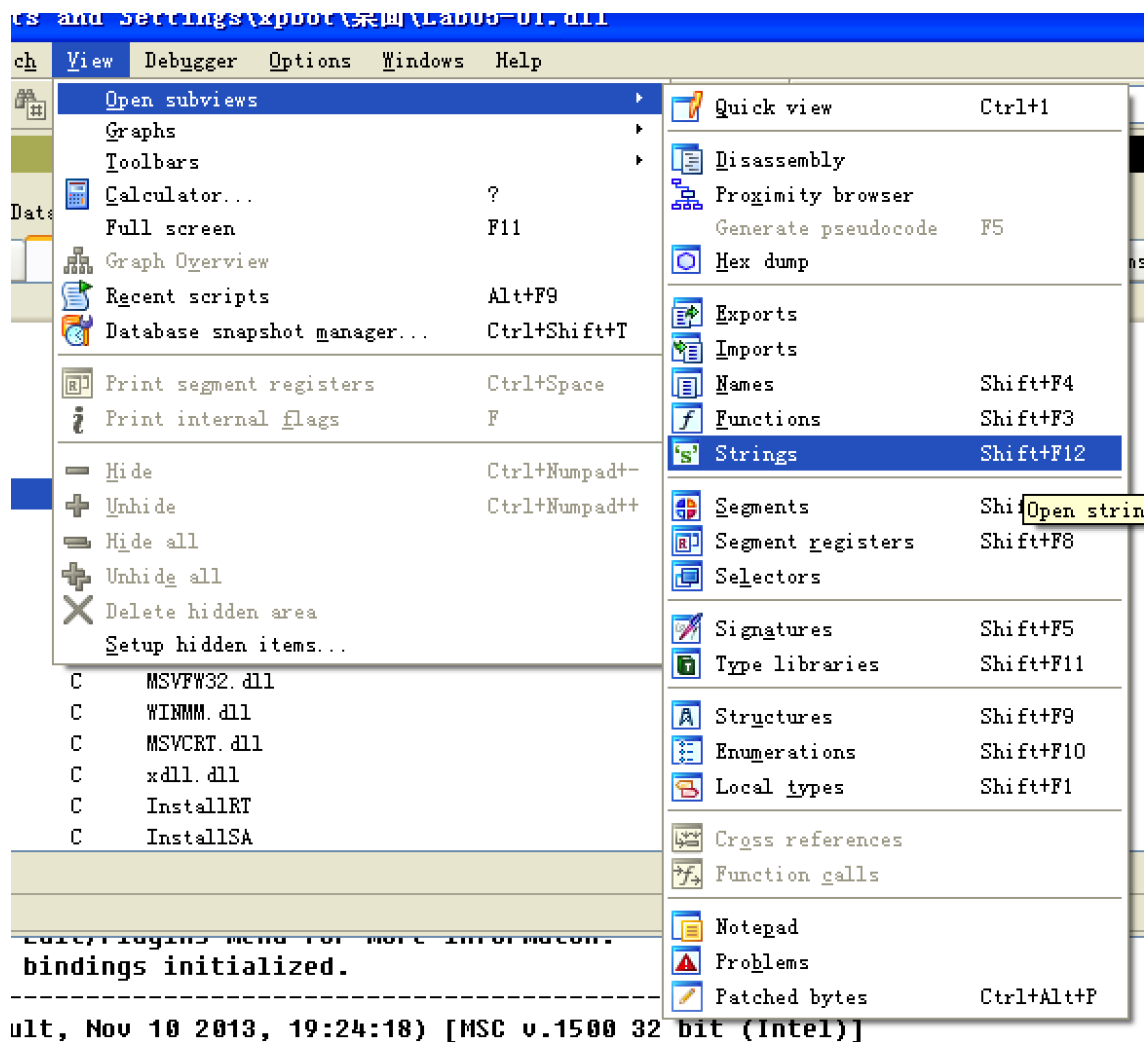
Address	Length	Type	String
["..."] .rdata:0...	0000000F	C	GetStringTypeW/
["..."] .rdata:0...	0000000D	C	SetStdHandle
["..."] .rdata:0...	0000000C	C	CloseHandle
["..."] .rdata:0...	0000000D	C	KERNEL32.dll
["..."] .data:00...	00000018	C	Error 1.1: No Internet\n
["..."] .data:00...	0000001E	C	Success: Internet Connection\n





允允能日新月昇

# IDA Pro version 6

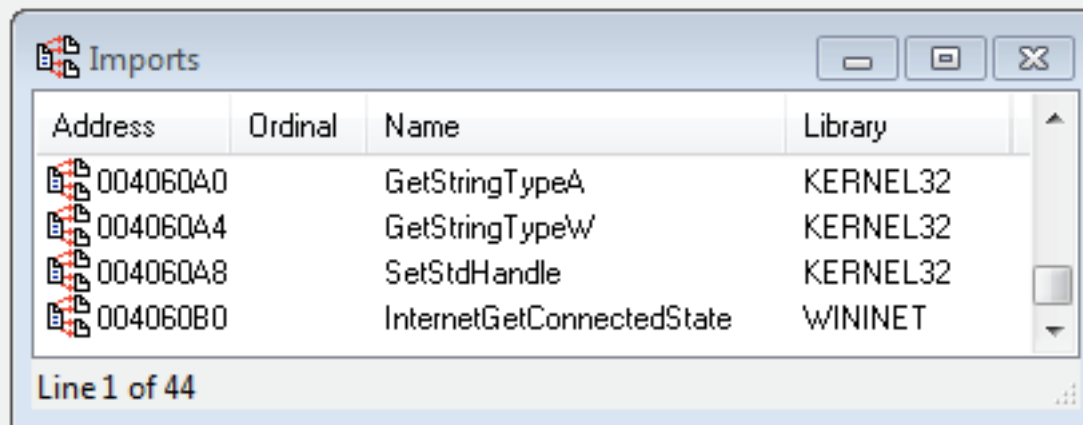


南开大学  
Nankai University



允公允能 日新月异

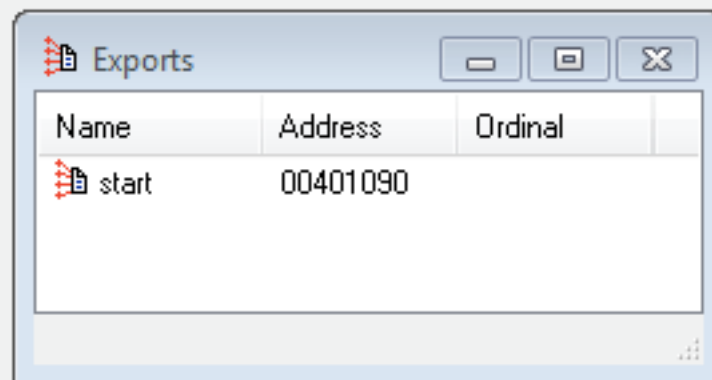
# Imports & Exports



Imports

Address	Ordinal	Name	Library
004060A0		GetStringTypeA	KERNEL32
004060A4		GetStringTypeW	KERNEL32
004060A8		SetStdHandle	KERNEL32
004060B0		InternetGetConnectedState	WININET

Line 1 of 44



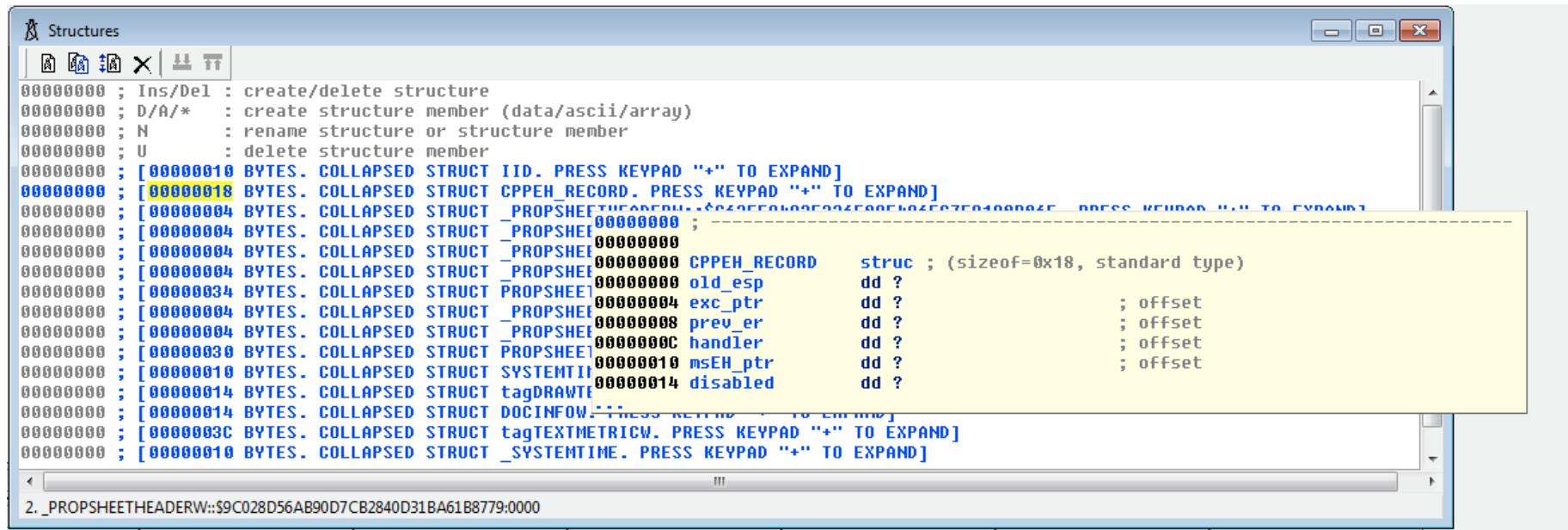
Exports

Name	Address	Ordinal
start	00401090	



允公允能 日新月异

- All active data structures
  - Hover to see yellow pop-up window



# Cross-Reference

- Double-click function
- Jump to code in other views

Functions window

Function name	Segment	Start	Length
__except_handler4	.text	01009D73	00000025
RegWriteString(x,x,x)	.text	01004788	0000002A
__chkstk	.text	010027F0	0000002B
_LegacyFileDialogToHR(x)	.text	010077F3	0000002B
ProcessSetupOption(x)	.text	010025EF	0000002D
CreateFilter(x,x)	.text	010026D5	0000002E
SignalCommDlgError()	.text	01004328	0000002F

Line 50 of 161

Hex View-A

.text:01004740	FF C9 C2 04 00 90 90 90	06 00 00 00 BA 0B 00 00	+ - J . ÉÉÉ - ... ! 8 ..
.text:01004750	07 00 00 00 BB 0B 00 00	50 FF D6 E9 D0 D2 FF FF	• ... + 8 .. P + T --
.text:01004760	50 FF D6 E9 D5 D2 FF FF	0F B7 16 66 85 D2 0F 84	P + T + - ... + T f à - ... ä
.text:01004770	8A 00 00 00 3B FB 0F 83	82 00 00 00 66 89 10 40	è ... ; v ... â é ... f è + @
.text:01004780	40 46 46 47 EB E2 90 90	90 90 90 8B FF 55 8B EC	@ FFGdGÉÉÉÉÉÏ UÏ8
.text:01004790	FF 75 10 FF 15 10 11 00	01 8D 44 00 02 50 FF 75	u + \$ + 4 . i D . 7 P u
.text:010047A0	10 6A 01 6A 00 FF 75 0C	FF 75 08 FF 15 00 10 00	+ j j . u ? u \$ . + .
.text:010047B0	01 5D C2 0C 00 90 90 90	90 90 8B FF 55 8B EC 8B	] - 7 . ÉÉÉÉÉÏ UÏ8Ï

# Function Call

- Parameters pushed onto stack
- CALL to start function

```

0100478B
0100478B
0100478B      ; Attributes: bp-based frame
0100478B
0100478B      ; int __stdcall RegWriteString(HKEY hKey,LPCWSTR lpValueName,BYTE *lpData)
0100478B      _RegWriteString@12 proc near
0100478B
0100478B      hKey= dword ptr  8
0100478B      lpValueName= dword ptr  0Ch
0100478B      lpData= dword ptr  10h
0100478B
0100478B 8B FF      mov     edi, edi
0100478D 55      push    ebp
0100478E 8B EC      mov     ebp, esp
01004790 FF 75 10    push    [ebp+lpData] ; lpString
01004793 FF 15 10 11 00 01 call    ds:__imp__lstrlenW@4 ; lstrlenW(x)
  
```

0.00% (-30,-41) (788,342) 00003B8B 0100478B: RegWriteString(x,x,x)



允公允能 日新月异

# Returning to the Default View

- Windows, Reset Desktop
- Windows, Save Desktop
  - To save a new view





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異

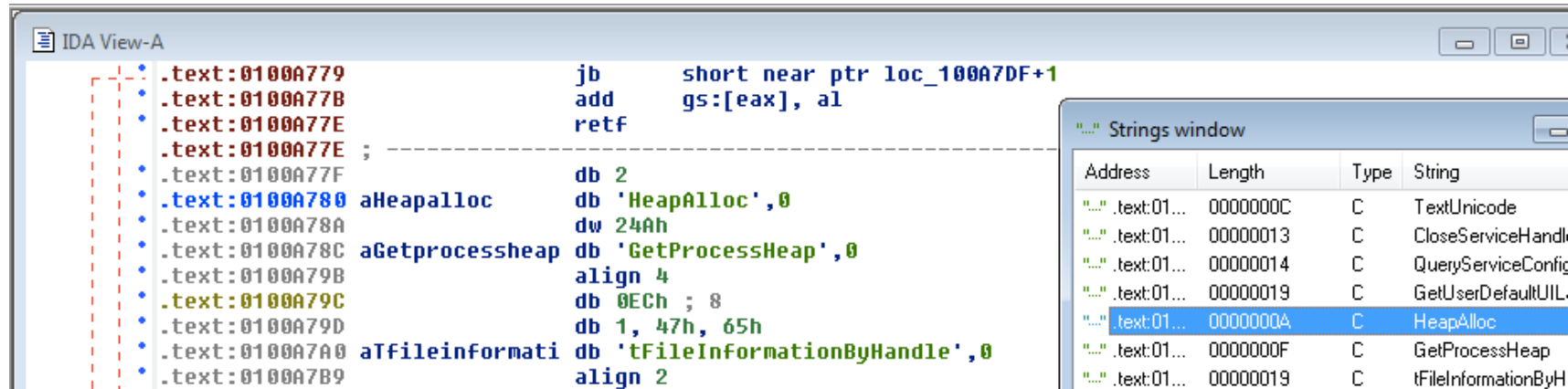


# Navigating IDA Pro



# Imports or Strings

- Double-click any entry to display it in the disassembly window







# Using Links

- Double-click any address in the disassembly window to display that location

```
IDA View-A
• .text:010047A1      push    1           ; dwType
• .text:010047A3      push    0           ; Reserved
• .text:010047A5      push    [ebp+lpValueName] ; lpValueName
• .text:010047A8      push    [ebp+hKey]   ; hKey
• .text:010047AB      call    ds:_imp__RegSetValueExW@24 ; RegSetValueExW(x,x,x,x,x,x)
• .text:010047B1      pop     ebp
• .text:010047B2      retn     0Ch
• .text:010047B2      _RegWriteString@12 endp
• .text:010047B2
• .text:010047B2
```





允公允能 日新月异

# Link Types

- Sub links
  - links to the start of **functions**
  - sub\_4010A0
- Loc links
  - links to jump **destinations**
  - loc\_401097

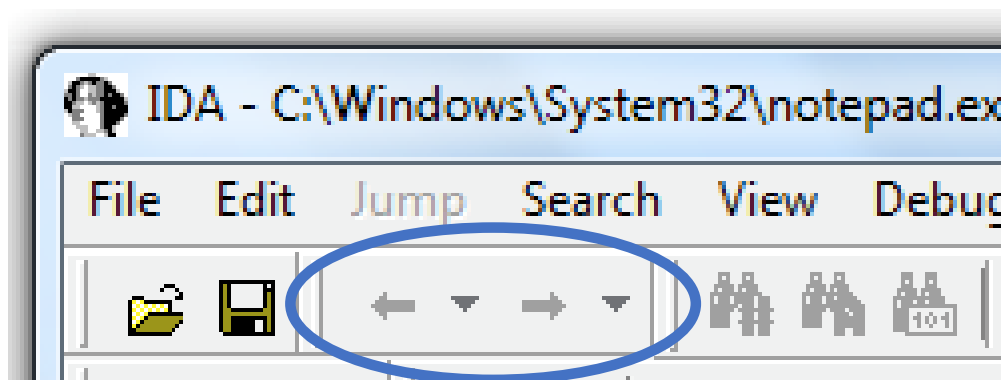




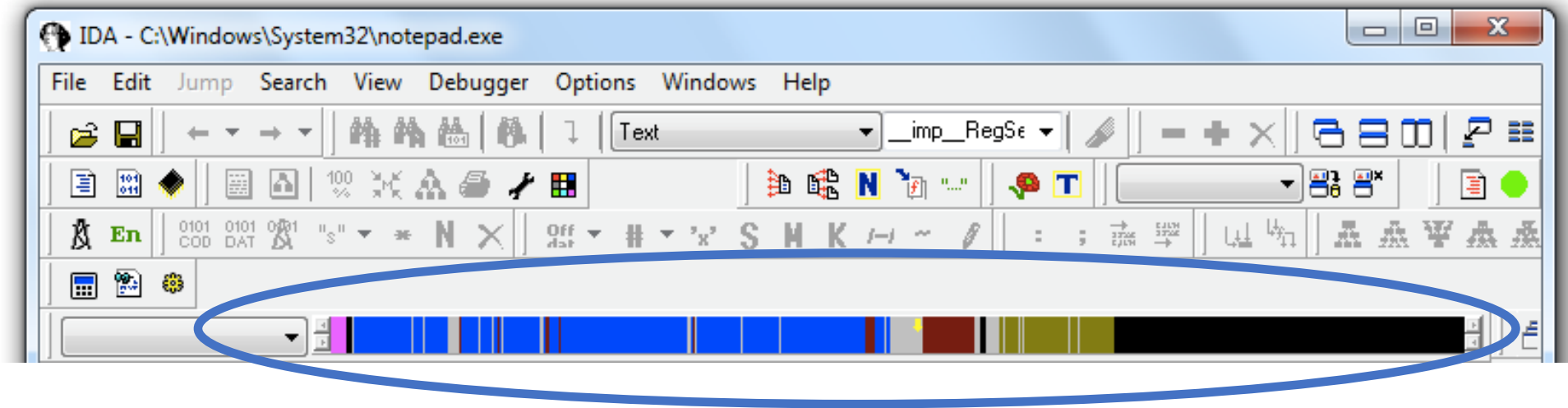
允公允能 日新月异

# History

- Forward and Back buttons work like a Web browser



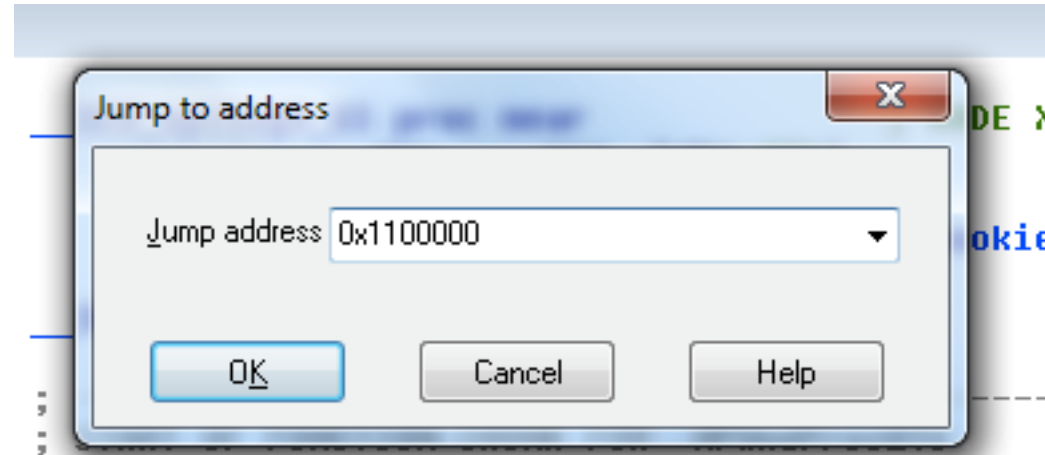
# Navigation Band



- **Light blue:** Library code
- **Red:** Compiler-generated code
- **Dark blue:** User-written code – **Analyze this**

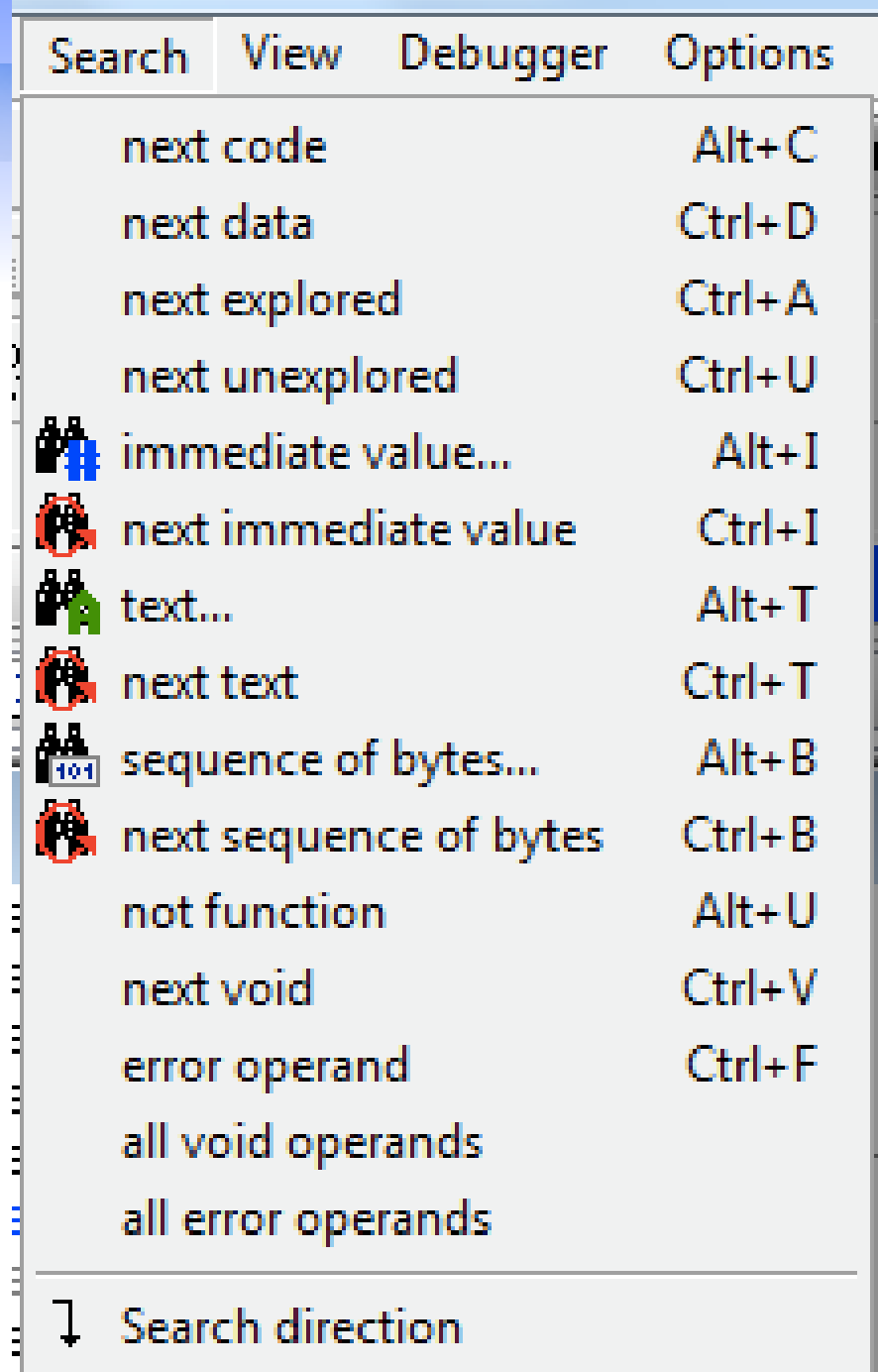
# Jump to Location

- Press g
- Can jump to address or named location



# Searching

- Many options
- Search, Text is handy





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



# Using Cross-References

# Code Cross-References

```
.text:00401440
.text:00401440 ; !!!!!!!!!!!!!!! S U B R O U T I N E !!!!!!!!!!!!!!!
.text:00401440
.text:00401440
.text:00401440 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401440 _main          proc near          ; CODE XREF: start+DE↓
.text:00401440
.text:00401440 var_44          = dword ptr -44h
.text:00401440 var_40          = dword ptr -40h
.text:00401440 var_3C          = dword ptr -3Ch
.text:00401440 var_38          = dword ptr -38h
.text:00401440 var_34          = dword ptr -34h
.text:00401440 var_30          = dword ptr -30h
.text:00401440 var_2C          = dword ptr -2Ch
.text:00401440 var_28          = dword ptr -28h
.text:00401440 var_24          = dword ptr -24h
.text:00401440 var_20          = dword ptr -20h
.text:00401440 var_1C          = dword ptr -1Ch
.text:00401440 var_18          = dword ptr -18h

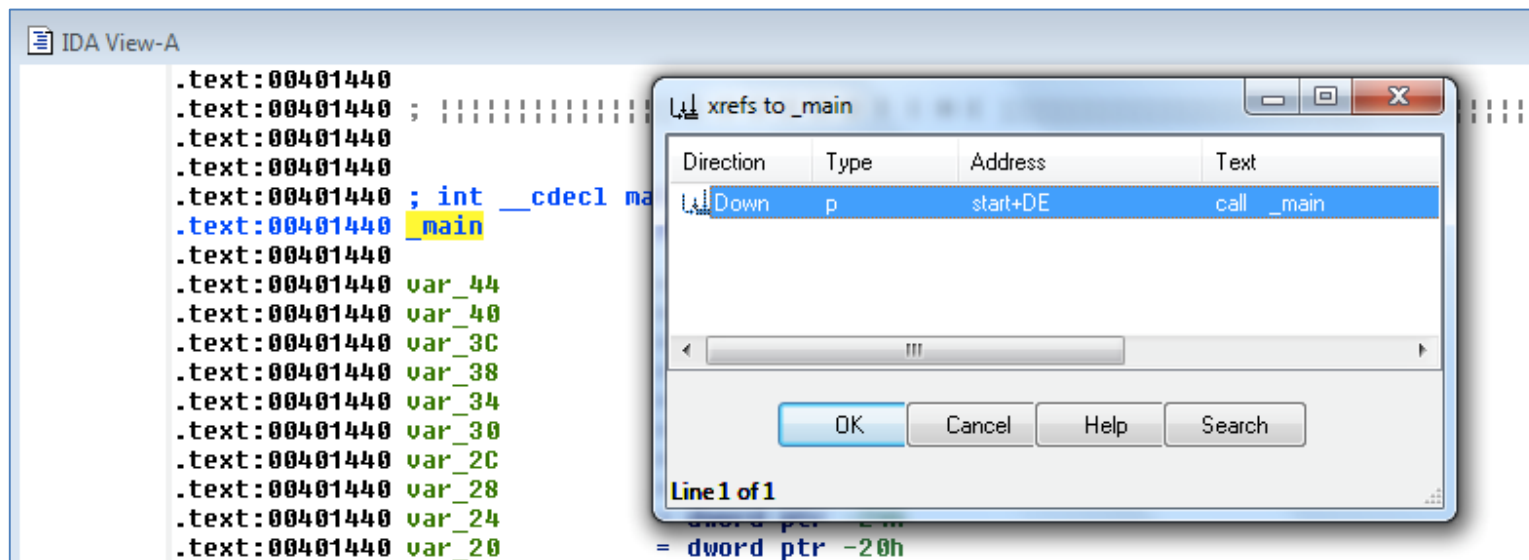
                                push     offset unk_403000
                                call     _initterm
                                call     ds: _p__initenv
                                mov      ecx, [ebp+envp]
                                mov      [eax], ecx
                                push     [ebp+envp]          ; envp
                                push     [ebp+argv]           ; argv
                                push     [ebp+argc]           ; argc
                                call     main
                                add      esp, 30h
```

- XREF comment shows where this function is called
- But it only shows a couple of cross-references by default



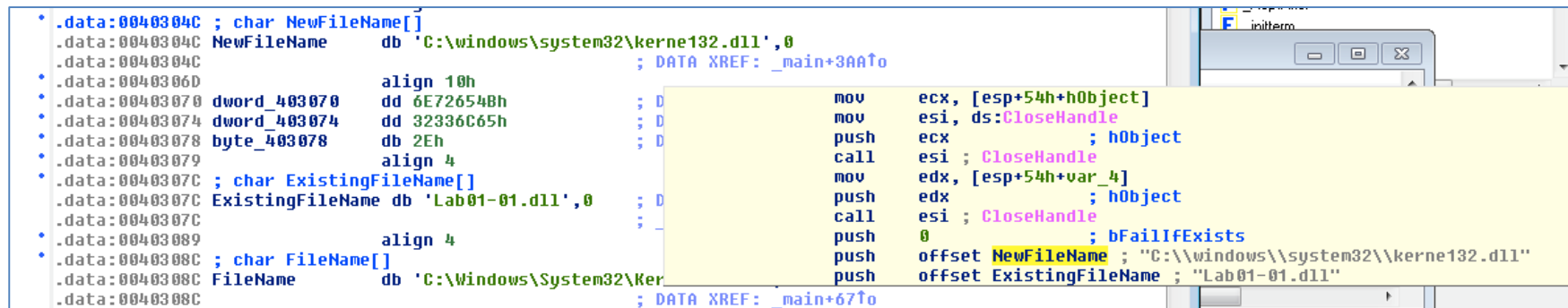
# To See All Cross-References

- Click function name and press **X**



# Data Cross-References

- Demo:
  - Start with strings
  - Double-click an interesting string
  - Hover over DATA XREF to see where that string is used
  - **X** shows all references



```
.data:0040304C ; char NewFileName[]
.data:0040304C NewFileName db 'C:\windows\system32\kerne132.dll',0
.data:0040304C ; DATA XREF: _main+3AAfo
.data:00403060 align 10h
.data:00403070 dword_403070 dd 6E726548h
.data:00403074 dword_403074 dd 32336C65h
.data:00403078 byte_403078 db 2Eh
.data:00403079 align 4
.data:0040307C ; char ExistingFileName[]
.data:0040307C ExistingFileName db 'Lab01-01.dll',0
.data:0040307C ; DATA XREF: _main+3AAfo
.data:00403089 align 4
.data:0040308C ; char FileName[]
.data:0040308C FileName db 'C:\Windows\System32\Ker
.data:0040308C ; DATA XREF: _main+67fo
```

```
mov ecx, [esp+54h+hObject]
mov esi, ds:CloseHandle
push ecx ; hObject
call esi ; CloseHandle
mov edx, [esp+54h+var_4]
push edx ; hObject
call esi ; CloseHandle
push 0 ; bFailIfExists
push offset NewFileName ; "C:\windows\system32\kerne132.dll"
push offset ExistingFileName ; "Lab01-01.dll"
```



南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

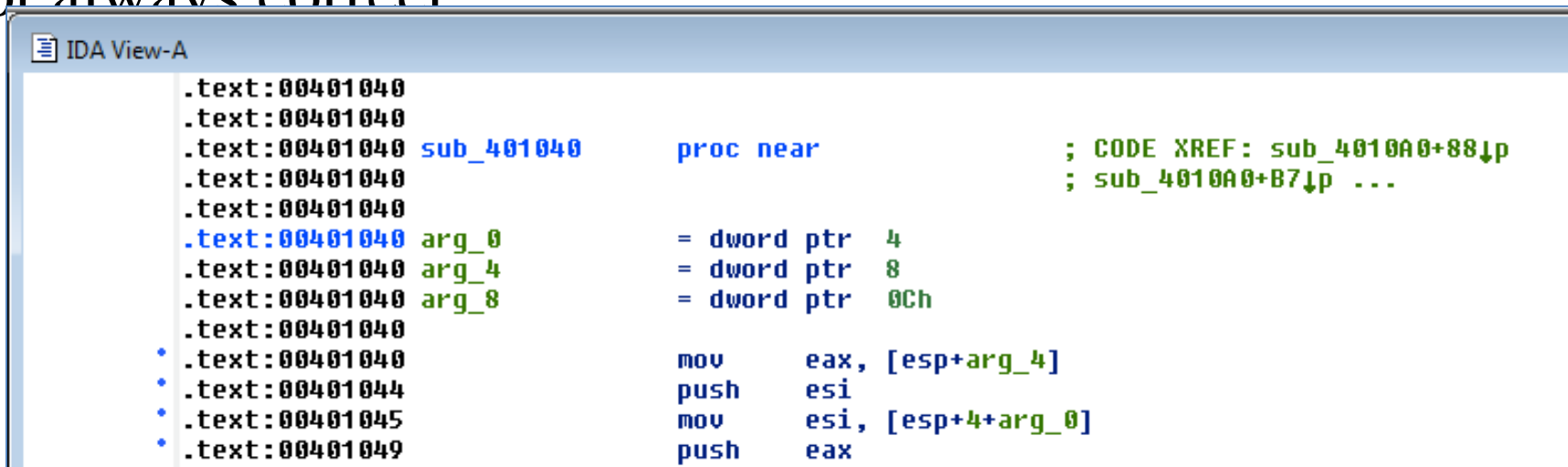
允公允能 日新月異



# Analyzing Functions

# Function and Argument Recognition

- IDA Pro identifies a function, names it, and also names the local variables
- It's not always correct



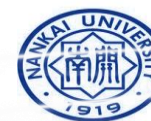
```
IDA View-A
.text:00401040
.text:00401040
.text:00401040 sub_401040      proc near          ; CODE XREF: sub_4010A0+88↓p
.text:00401040                                     ; sub_4010A0+B7↓p ...
.text:00401040
.text:00401040 arg_0          = dword ptr 4
.text:00401040 arg_4          = dword ptr 8
.text:00401040 arg_8          = dword ptr 0Ch
.text:00401040
* .text:00401040      mov     eax, [esp+arg_4]
* .text:00401044      push    esi
* .text:00401045      mov     esi, [esp+4+arg_0]
* .text:00401049      push    eax
```





# Naming Convention

- local variables
  - prefix: var\_
  - suffix: offset relative to EBP
  - negative offset
- parameters
  - prefix arg\_
  - suffix: offset relative to EBP
  - positive offset





# 允公允能 日新月异

```
var_14 = dword ptr -14h
var_10 = dword ptr -10h
var_C = dword ptr -0Ch
var_8 = dword ptr -8
var_4 = dword ptr -4
arg_0 = dword ptr 8
arg_4 = dword ptr 0Ch
```

9

```
push    ebp
mov     ebp, esp
sub     esp, 30h
push    esi
push    edi
mov     [ebp+var_C], 0
mov     word ptr [ebp+var_8], 0
mov     word ptr [ebp+var_4], 0
mov     byte ptr [ebp+var_10], 0
mov     eax, [ebp+arg_4]
```





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



# Using Graphing Options



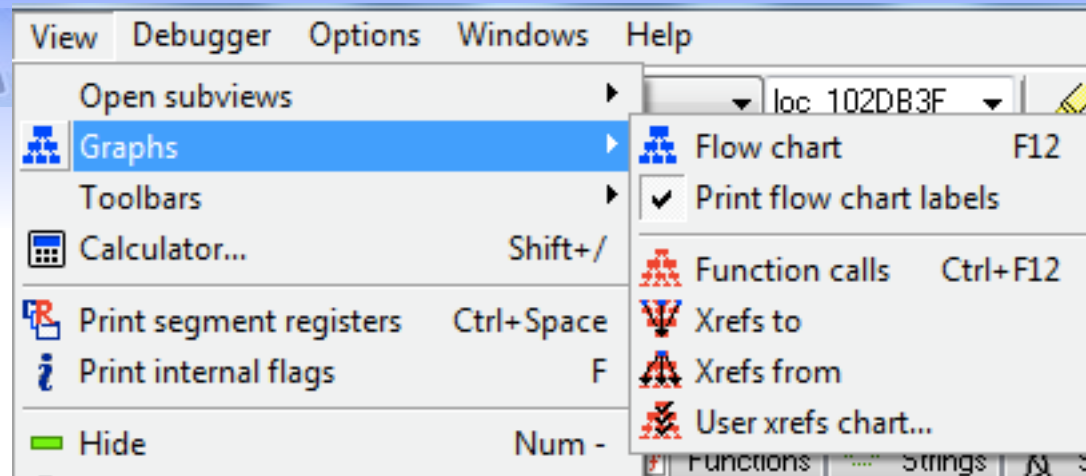


# Graphing Options





# Graphing Options

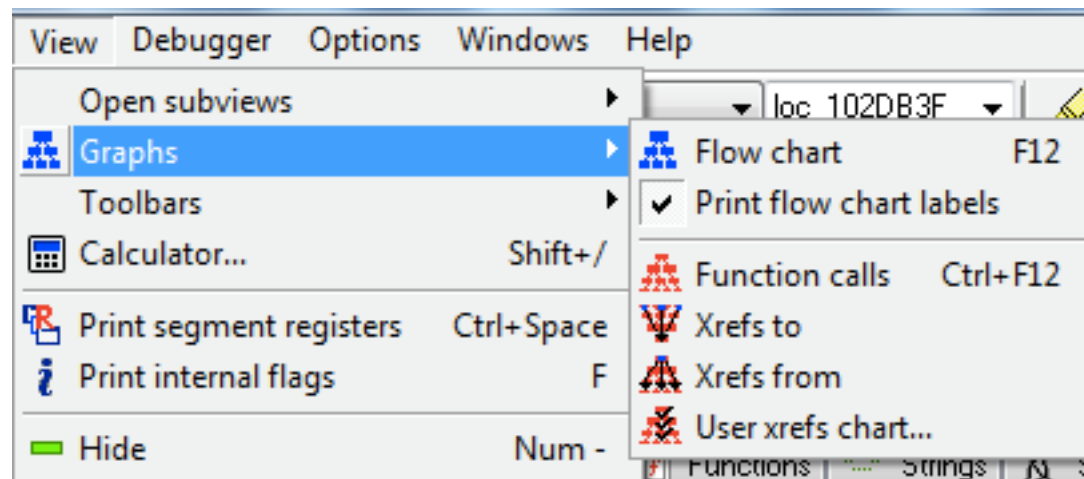


- These are "Legacy Graphs" and cannot be manipulated with IDA
- The first two seem obsolete
  - **Flow chart**
    - Create flow chart of current function
  - **Function calls**
    - Graph function calls for entire program





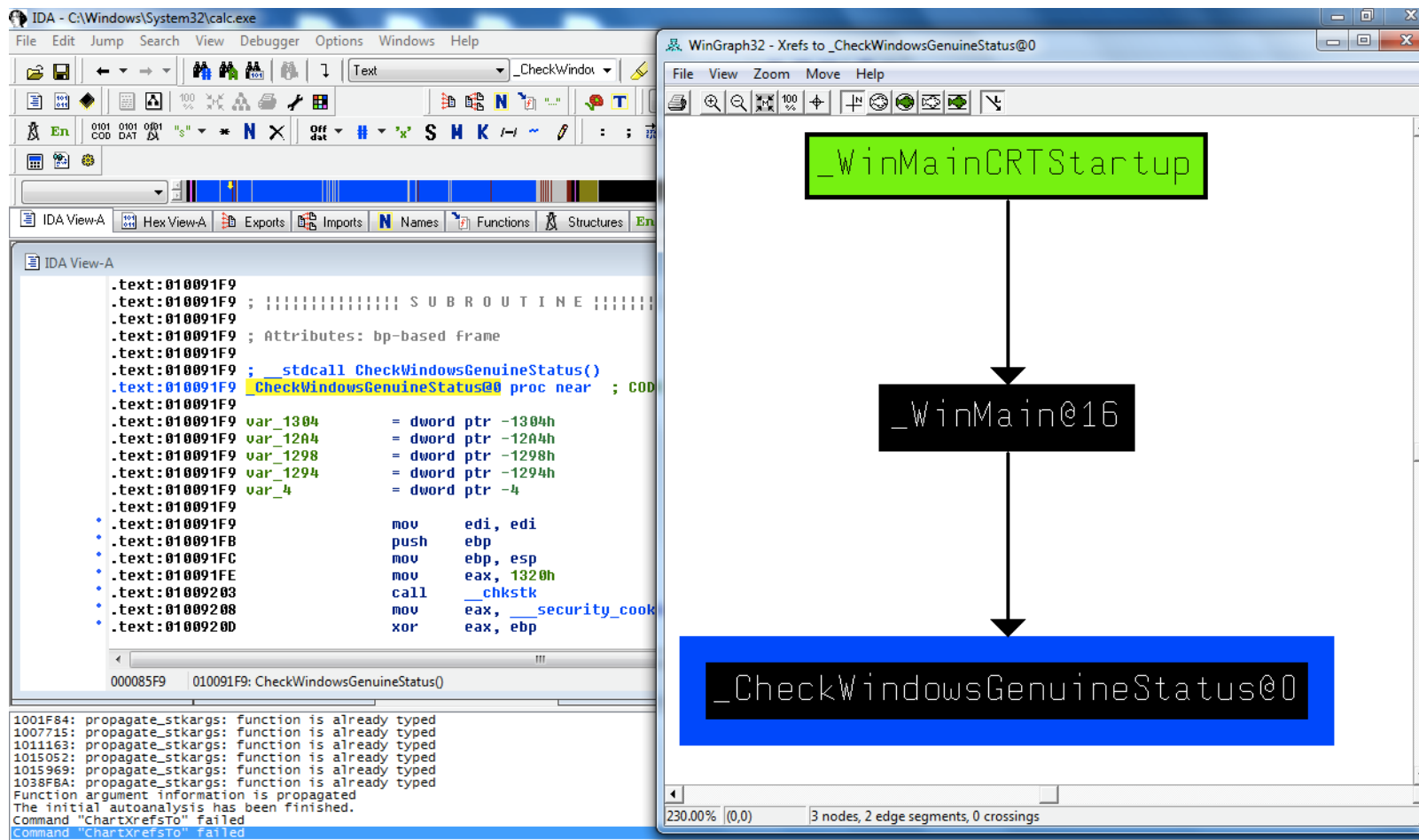
# Graphing Options



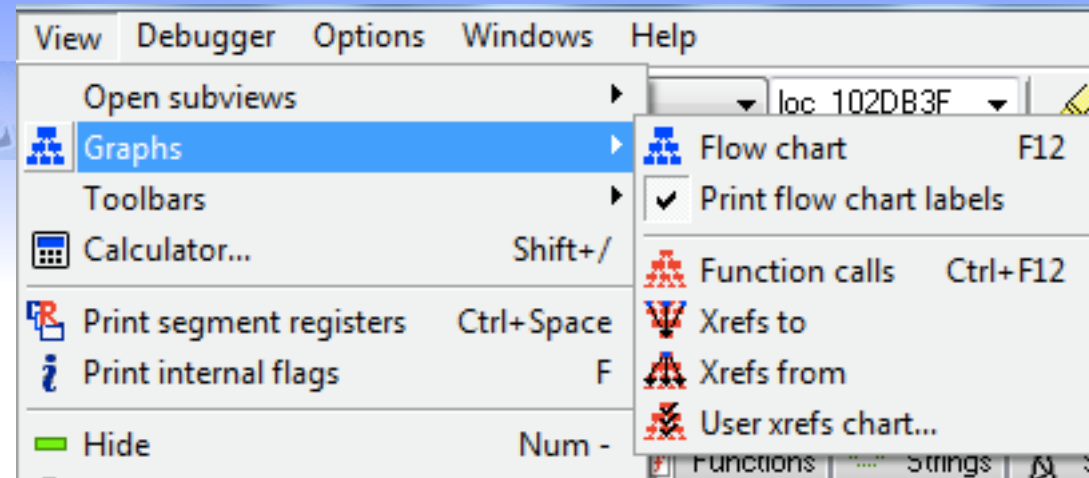
- **Xrefs to**
  - Graphs XREFs to get to selected XREF
  - Can show all the paths that get to a function



# Windows Genuine Status in Calc.exe

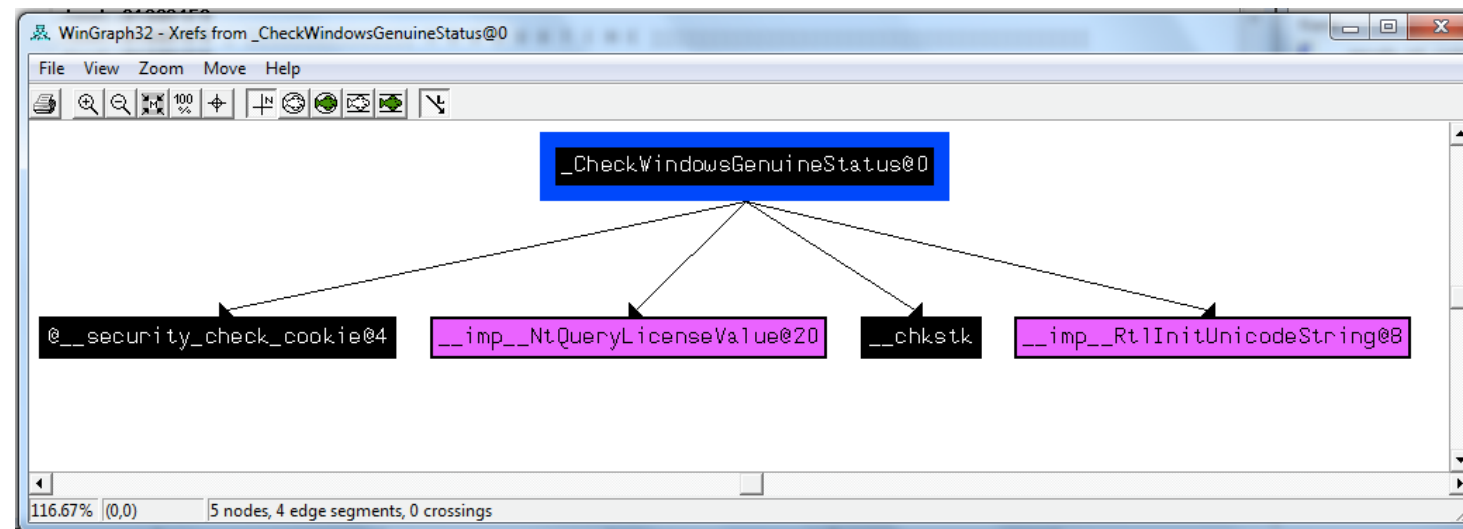


# Graphing Options

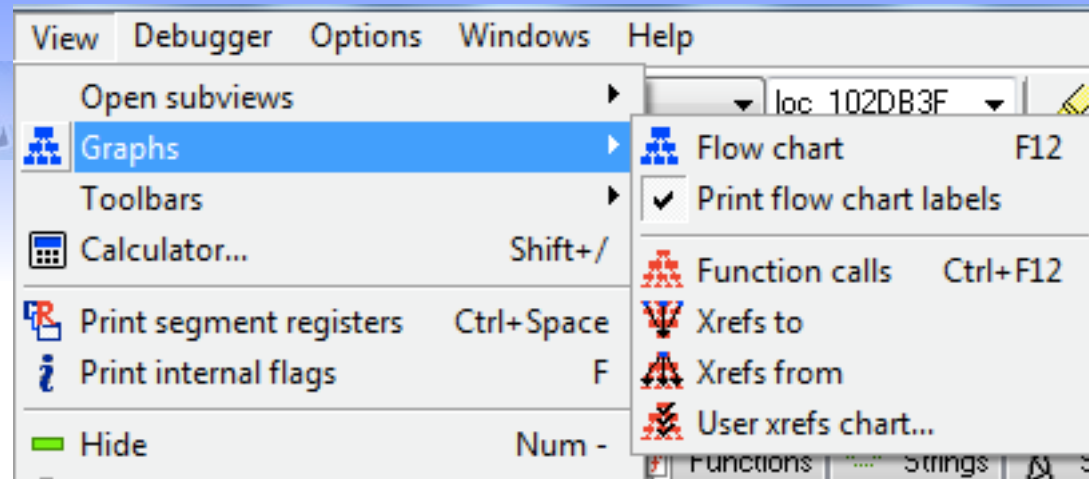


- **Xrefs from**

- Graphs XREFs from selected XREF
- Can show all the paths that exit from a function



# Graphing Options



- **User xrefs chart...**
  - Customize graph's recursive depth, symbols used, to or from symbol, etc.
  - The only way to modify legacy graphs





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



Enhancing Disassembly



允公允能 日新月异

# Warning

- There's no Undo, so if you make changes and mess them up, you may be sorry







允公允能 日新月异

# Renaming Locations

- You can change a name like **sub\_401000** to **ReverseBackdoorThread**
- Change it in one place, IDA will change it everywhere else







Table 6-2. Function Operand Manipulation

Without renamed arguments

```
004013C8 mov    eax, [ebp+arg_4]
004013CB push   eax
004013CC call  _atoi
004013D1 add    esp, 4
004013D4 mov    [ebp+var_598], ax
004013DB movzx  ecx, [ebp+var_598]
004013E2 test   ecx, ecx
004013E4 jnz     short loc_4013F8
004013E6 push   offset aError
004013EB call  printf
004013F0 add    esp, 4
004013F3 jmp    loc_4016FB
004013F8 ; -----
004013F8
004013F8 loc_4013F8:
004013F8 movzx  edx, [ebp+var_598]
004013FF push   edx
00401400 call  ds:htons
```

With renamed arguments

```
004013C8 mov    eax, [ebp+port_str]
004013CB push   eax
004013CC call  _atoi
004013D1 add    esp, 4
004013D4 mov    [ebp+port], ax
004013DB movzx  ecx, [ebp+port]
004013E2 test   ecx, ecx
004013E4 jnz     short loc_4013F8
004013E6 push   offset aError
004013EB call  printf
004013F0 add    esp, 4
004013F3 jmp    loc_4016FB
004013F8 ; -----
004013F8
004013F8 loc_4013F8:
004013F8 movzx  edx, [ebp+port]
004013FF push   edx
00401400 call  ds:htons
```





允公允能 日新月异

# Comments

- Press colon (:) to add a single comment
- Press semicolon (;) to echo this comment to all Xrefs





# Formatting Operands

- Hexadecimal by default
- Right click to use other formats

```
mov     edi, edi
push    ebp
mov     ebp, esp
mov     eax, 1320h
call    __chkstk
mov     eax, ___se
xor     eax, ebp
mov     [ebp+var_4], eax
push    offset aSe
```

**M** Use standard symbolic constant

**#10** 4896

**H**

**#8** 11440o

**#2** 1001100100000b

**B**





# Using Named Constants

- Makes Windows API arguments clearer

Before symbolic constants	After symbolic constants
<pre>mov     esi, [esp+1Ch+argv] mov     edx, [esi+4] mov     edi, ds:CreateFileA push    0      ; hTemplateFile push    80h    ; dwFlagsAndAttributes push    3      ; dwCreationDisposition push    0      ; lpSecurityAttributes push    1      ; dwShareMode</pre>	<pre>mov     esi, [esp+1Ch+argv] mov     edx, [esi+4] mov     edi, ds:CreateFileA push    NULL   ; hTemplateFile push    FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes push    OPEN_EXISTING ; dwCreationDisposition push    NULL   ; lpSecurityAttributes push    FILE_SHARE_READ ; dwShareMode</pre>





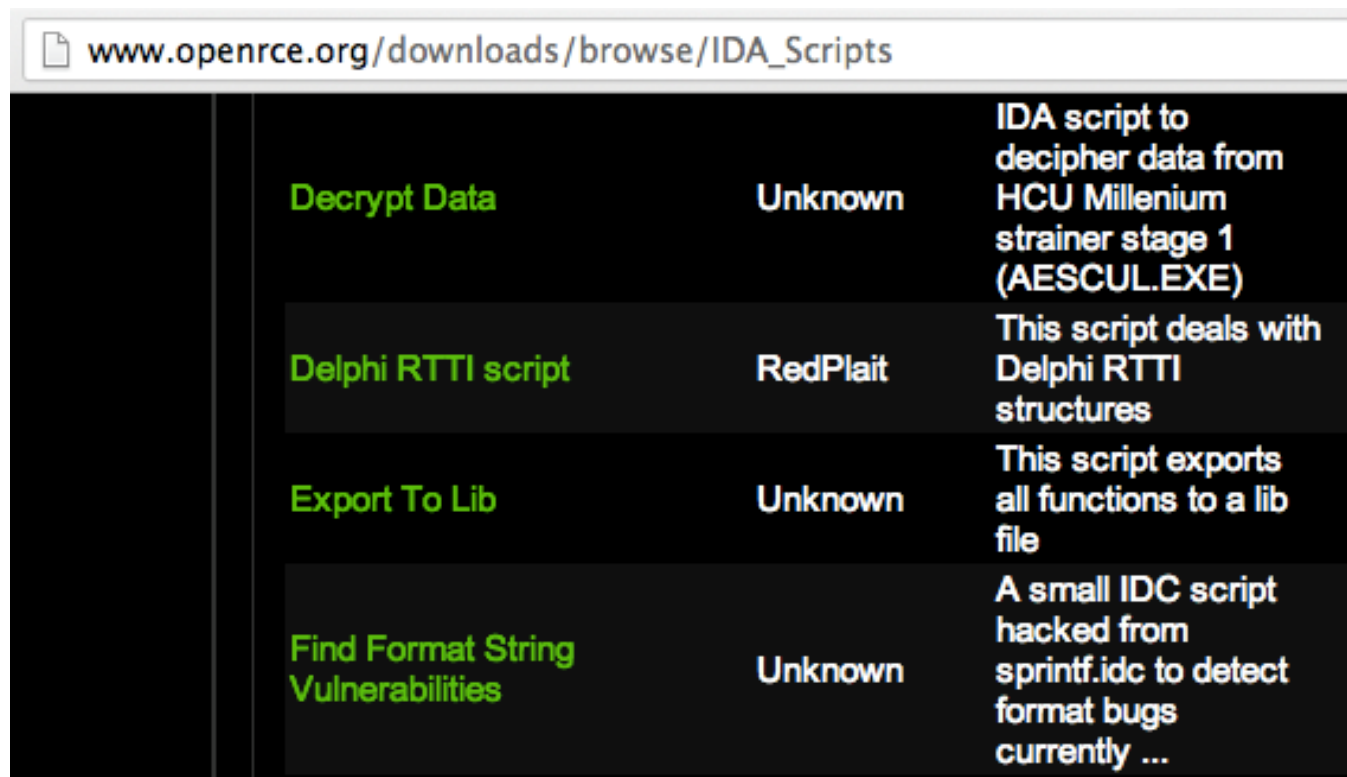
# Redefining Code and Data

- IDA Pro is not always correct.
- **U**: undefine functions, code or data
- **C**: define the raw data as code
- **D**: define the raw data as data, such as BYTE, WORD, DWORD
- **A**: define the raw data as ASCII strings



# Extending IDA with Plug-ins

- IDC (IDA's scripting language) and Python scripts available (link Ch 6a)



A screenshot of a web browser window displaying a list of IDA scripts. The browser's address bar shows the URL [www.openrce.org/downloads/browse/IDA\\_Scripts](http://www.openrce.org/downloads/browse/IDA_Scripts). The page content is a table with four rows, each representing a different script. The first column contains the script name, the second column contains the author, and the third column contains a brief description of the script's functionality.

Decrypt Data	Unknown	IDA script to decipher data from HCU Millenium strainer stage 1 (AESCUL.EXE)
Delphi RTTI script	RedPlait	This script deals with Delphi RTTI structures
Export To Lib	Unknown	This script exports all functions to a lib file
Find Format String Vulnerabilities	Unknown	A small IDC script hacked from sprintf.idc to detect format bugs currently ...





允公允能 日新月异

# Labs

- Analyze the malware found in the file Lab05-01.dll using only IDA Pro.
  - 1. What is the address of DllMain?
  - 2. Use the Imports window to browse to gethostbyname. Where is the import located?





允公允能 日新月异

# Labs

- 3. How many functions call gethostbyname?
- 4. Focusing on the call to gethostbyname located at 0x10001757, can you figure out which DNS request will be made?
- 5. How many local variables has IDA Pro recognized for the subroutine at 0x10001656?
- 6. How many parameters has IDA Pro recognized for the subroutine at 0x10001656?







允公允能 日新月异

# Labs

- 7. Use the Strings window to locate the string `\cmd.exe /c` in the disassembly. Where is it located?
- 8. What is happening in the area of code that references `\cmd.exe /c`?
- 9. In the same area, at `0x100101C8`, it looks like `dword_1008E5C4` is a global variable that helps decide which path to take. How does the malware set `dword_1008E5C4`? (Hint: Use `dword_1008E5C4`'s cross-references.)
- 10. A few hundred lines into the subroutine at `0x1000FF58`, a series of comparisons use `memcmp` to compare strings. What happens if the string comparison to `robotwork` is successful (when `memcmp` returns 0)?





允公允能 日新月异

# Labs

- 11. What does the export PSLIST do?
- 12. Use the graph mode to graph the cross-references from `sub_10004E79`. Which API functions could be called by entering this function? Based on the API functions alone, what could you rename this function?
- 13. How many Windows API functions does `DllMain` call directly? How many at a depth of 2?
- 14. At `0x10001358`, there is a call to `Sleep` (an API function that takes one parameter containing the number of milliseconds to sleep). Looking backward through the code, how long will the program sleep if this code executes?



南开大学  
Nankai University



# Labs

- 15. At 0x10001701 is a call to `socket`. What are the three parameters?
- 16. Using the MSDN page for `socket` and the named symbolic constants functionality in IDA Pro, can you make the parameters more meaningful? What are the parameters after you apply changes?
- 17. Search for usage of the `in` instruction (opcode 0xED). This instruction is used with a magic string VMXh to perform VMware detection. Is that in use in this malware? Using the cross-references to the function that executes the `in` instruction, is there further evidence of VMware detection?





允公允能 日新月异

# Labs

- 18. Jump your cursor to 0x1001D988. What do you find?
- 19. If you have the IDA Python plug-in installed (included with the commercial version of IDA Pro), run Lab05-01.py, an IDA Pro Python script provided with the malware for this book. (Make sure the cursor is at 0x1001D988.) What happens after you run the script?
- 20. With the cursor in the same location, how do you turn this data into a single ASCII string?
- 21. Open the script with a text editor. How does it work?





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



# Practical Malware Analysis

## Chapter 5: IDA Pro

王志

zwang@nankai.edu.cn

updated on Oct. 17<sup>th</sup> 2021

College of Cyber Science  
Nankai University  
2021/2022