

Practical Malware Analysis

Ch 5: IDA Pro



Last modified
9-28-21

IDA Pro Versions

- Full-featured pay version
- Old free version
 - Both support x86
 - Pay version supports x64 and other processors, such as cell phone processors
- Both have code signatures for common library code in FLIRT (Fast Library identification and Recognition Technology)

Graph and Text Mode

- Spacebar switches mode



The screenshot shows the IDA View-A window with the following assembly code:

```
.text:00401040
.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:00401041
* .text:00401043
    push    ebp
    mov     ebp, esp
    push    ecx
```

Default Graph Mode Display

```

; Attributes: bp-based frame

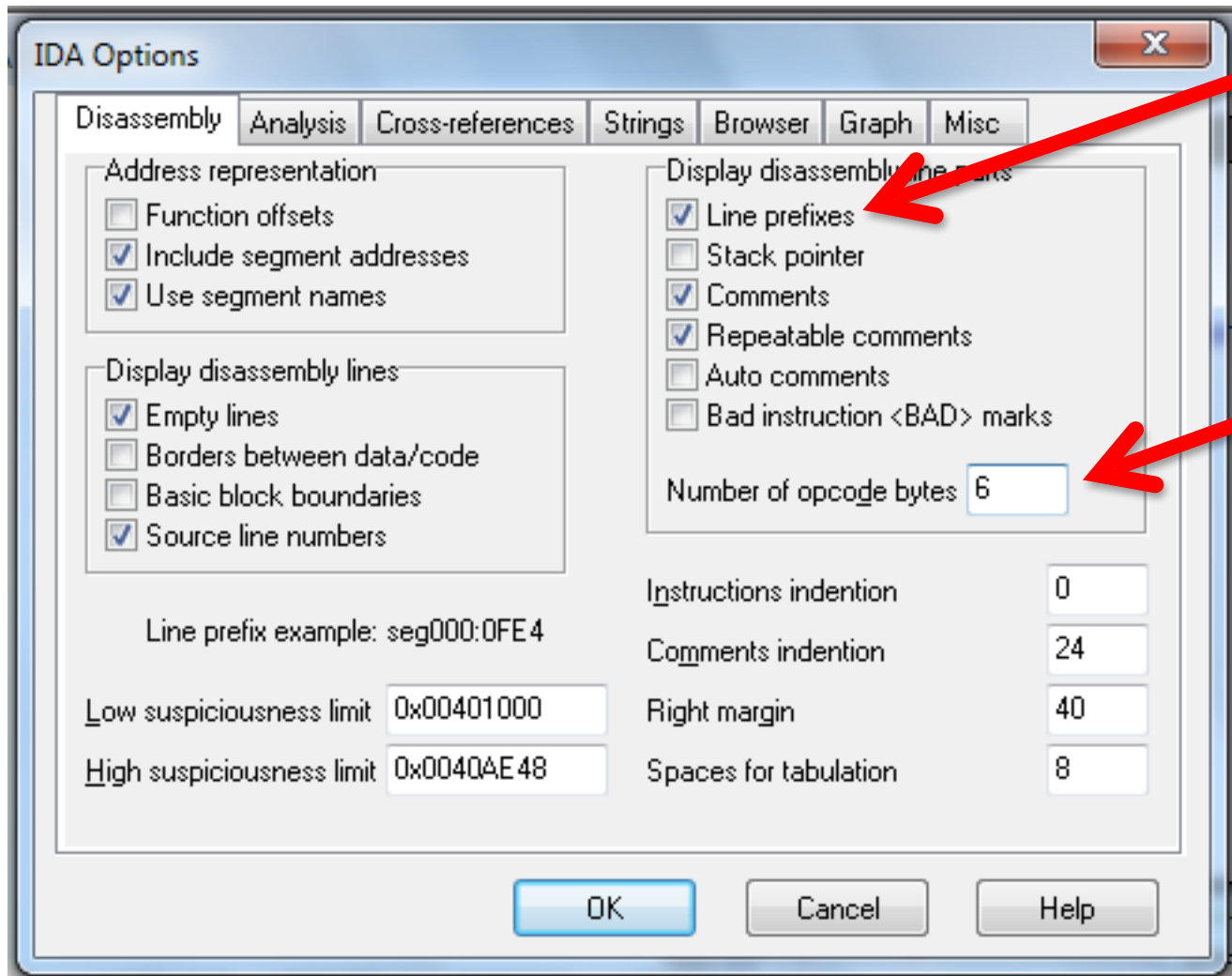
; int __cdecl main(int argc,const char **argv,const char *envp)
_main proc near

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

```

Options, General



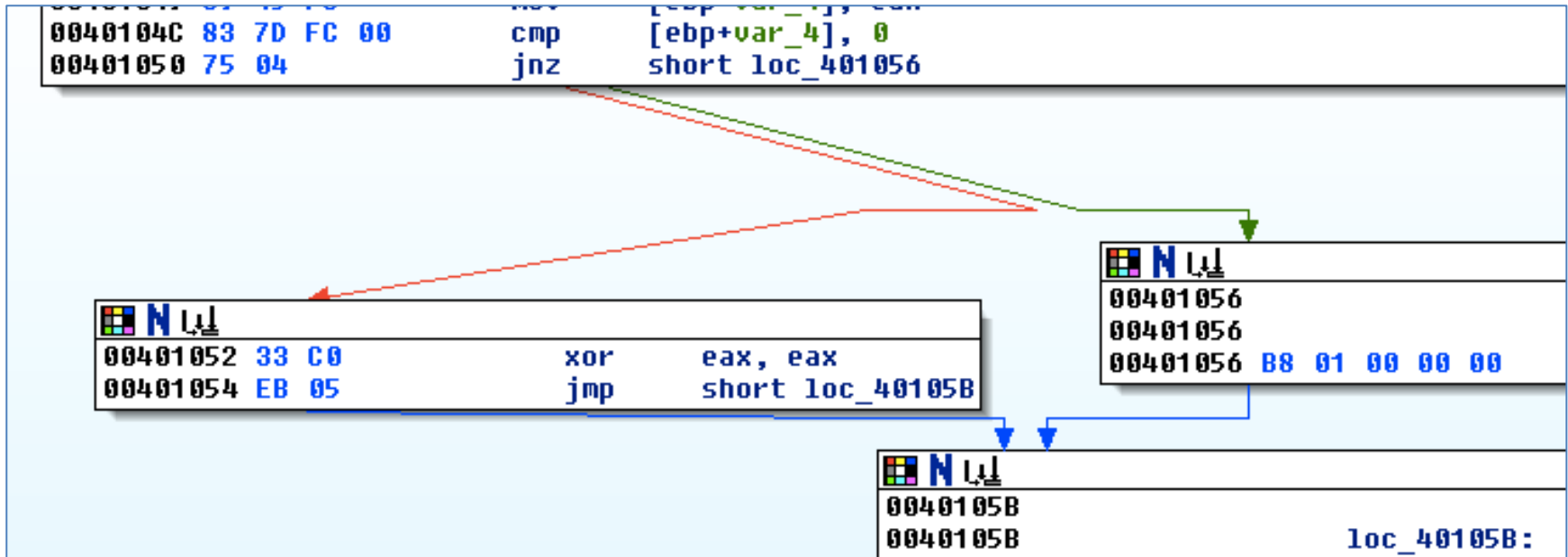
Better Graph Mode View

```
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
```

Arrows

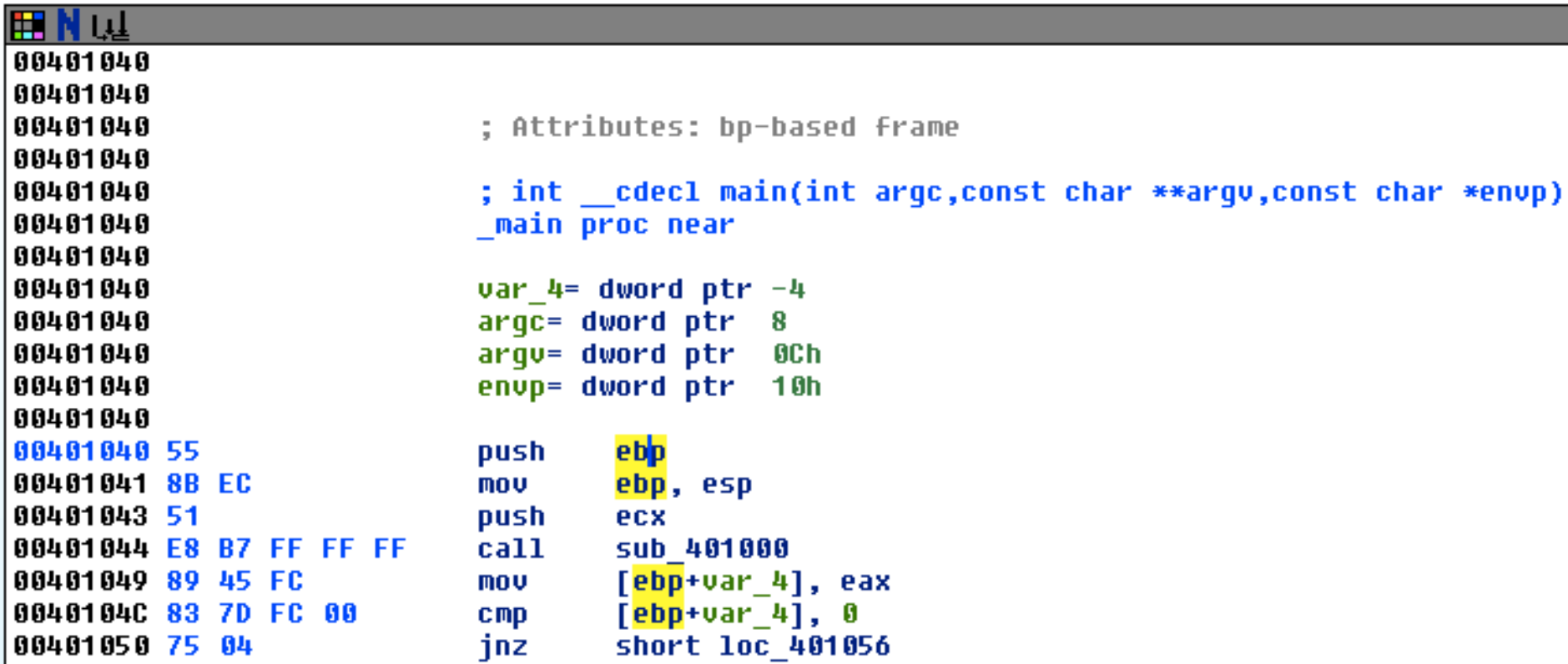
- Colors
 - Red Conditional jump not taken
 - Green Conditional jump taken
 - Blue Unconditional jump
- Direction
 - Up Loop

Arrow Color Example



Highlighting

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



The screenshot shows a debugger window with a title bar containing a logo and the text 'N'. The main area displays assembly code with several instances of the text 'ebp' highlighted in yellow. The code includes comments and instructions such as 'push ebp', 'mov ebp, esp', 'push ecx', 'call sub_401000', 'mov [ebp+var_4], eax', 'cmp [ebp+var_4], 0', and 'jnz short loc_401056'. The assembly code is color-coded: blue for addresses and instructions, green for variables, and black for other 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

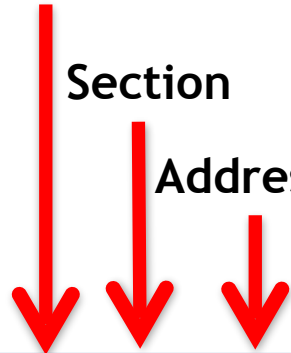
Dashed = Conditional

Up = Loop

Comment
Generated by
IDA Pro

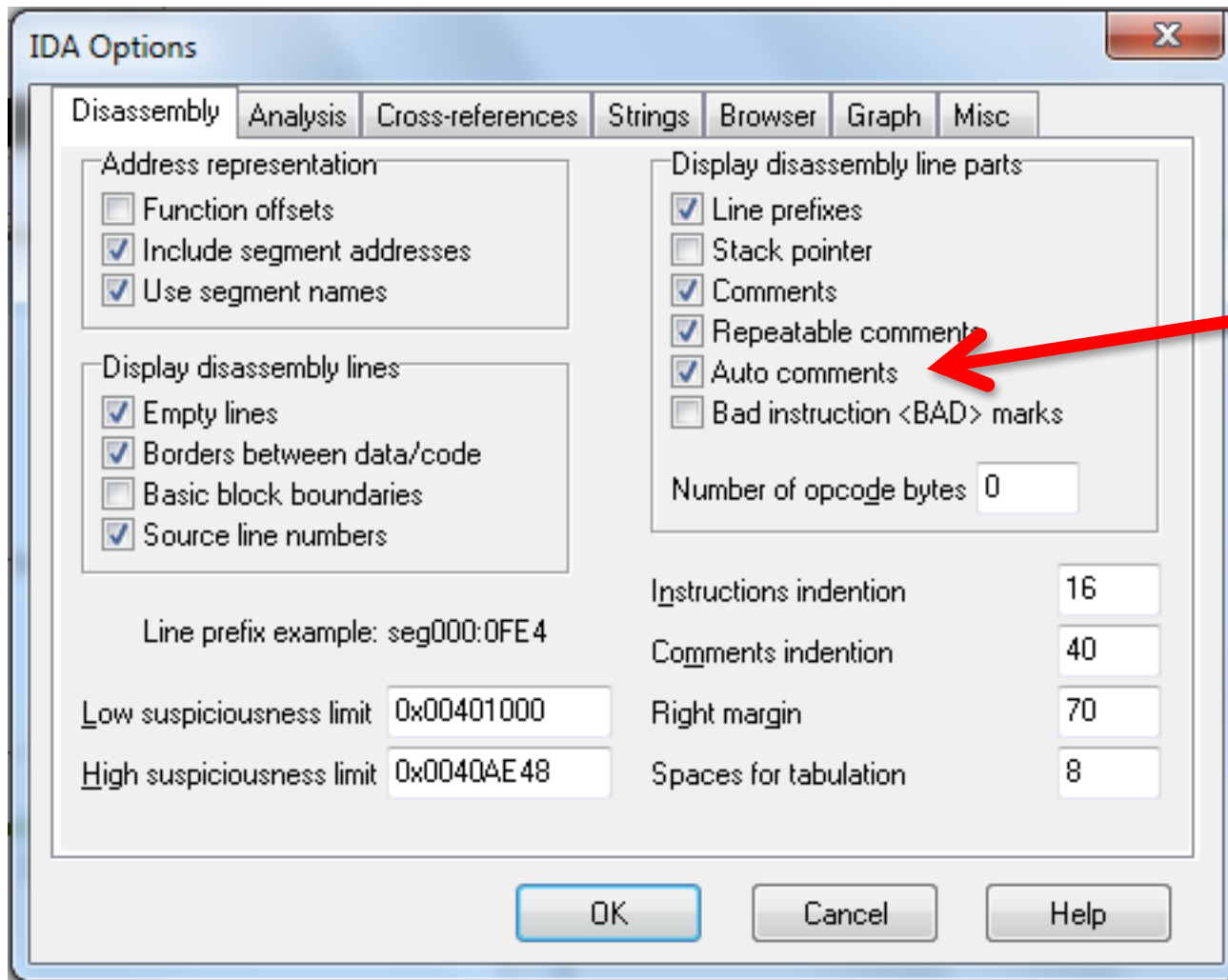
Section

Address



```
.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



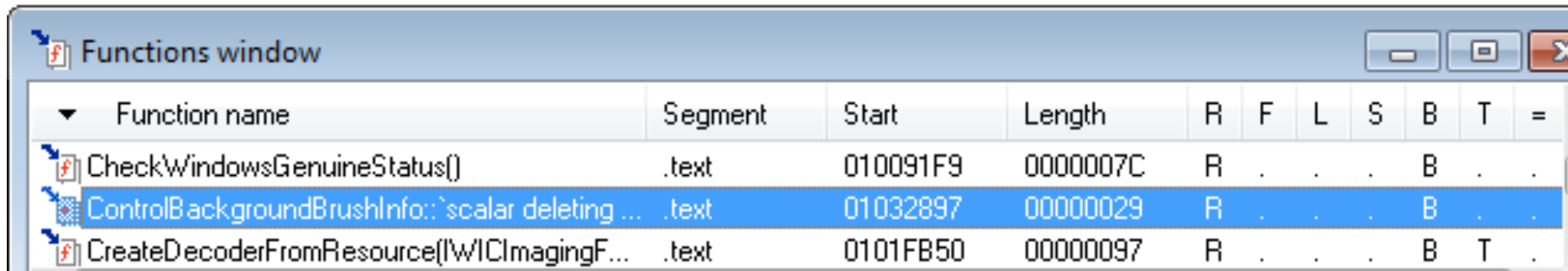
Adds Comments to Each Instruction

```
.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      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      loc_40103A:                ; CODE XREF: sub_401000+29↑j
.text:0040103A      mov    esp, ebp
.text:0040103C      pop    ebp
```

Useful Windows for Analysis

Functions

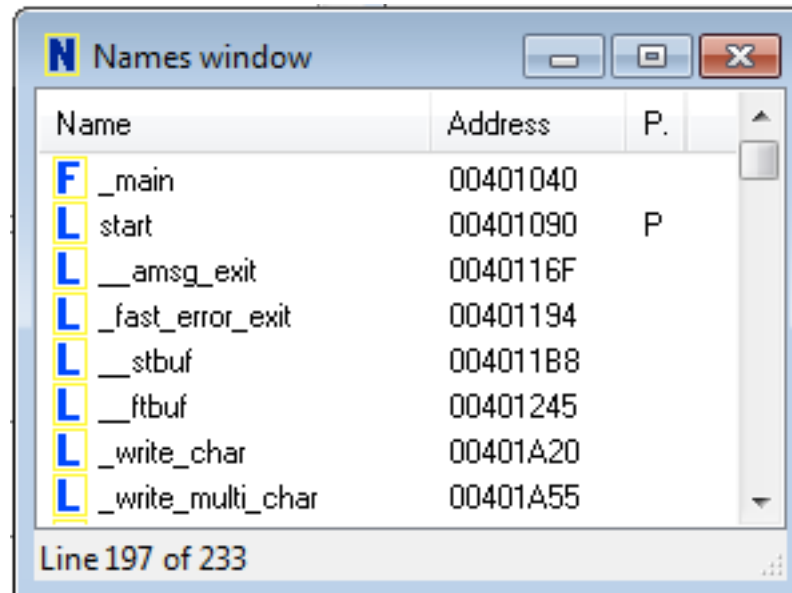
- Shows each function, length, and flags
 - L = Library functions
- Sortable
 - Large functions usually more important



Function name	Segment	Start	Length	R	F	L	S	B	T	=
CheckWindowsGenuineStatus()	.text	010091F9	0000007C	R	.	.	.	B	.	.
ControlBackgroundBrushInfo::`scalar deletingtext	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



The screenshot shows a window titled "Names window" with a table of symbols. The table has three columns: "Name", "Address", and "P.". The symbols listed are:

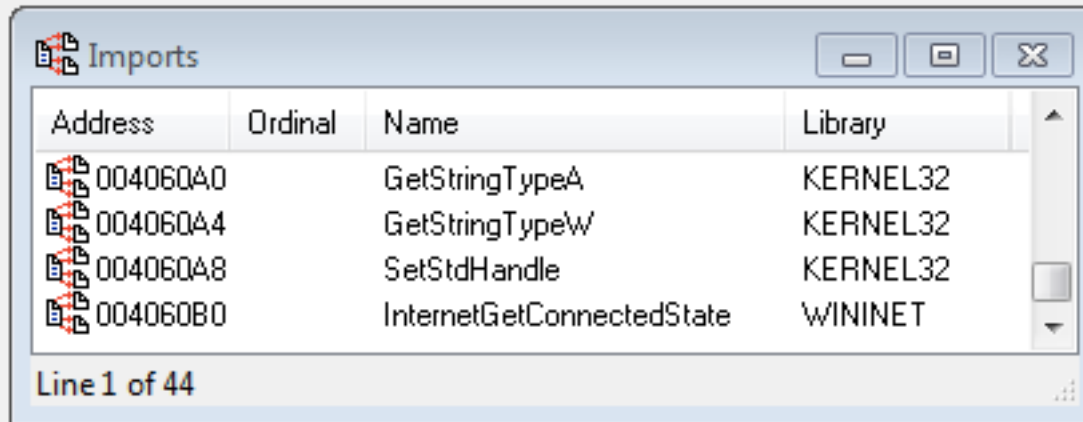
Name	Address	P.
F _main	00401040	
L start	00401090	P
L __amsg_exit	0040116F	
L _fast_error_exit	00401194	
L __stbuf	004011B8	
L __ftbuf	00401245	
L _write_char	00401A20	
L _write_multi_char	00401A55	

At the bottom of the window, it says "Line 197 of 233".

Strings

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

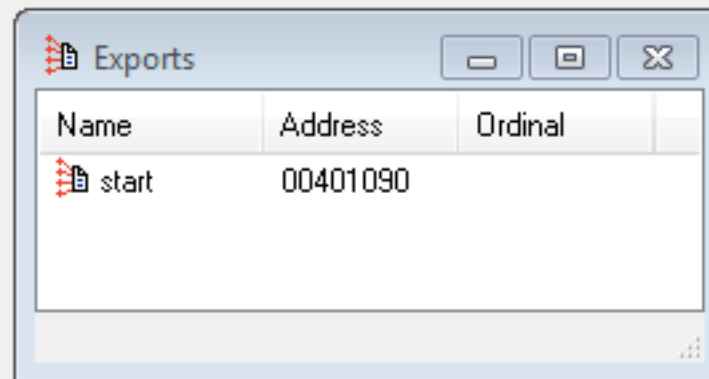
Imports & Exports



The Imports dialog box shows a list of imported functions. Each entry includes a small icon of a document with a red arrow, a hexadecimal address, the function name, and the library name. The status bar at the bottom indicates 'Line 1 of 44'.

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

Line 1 of 44

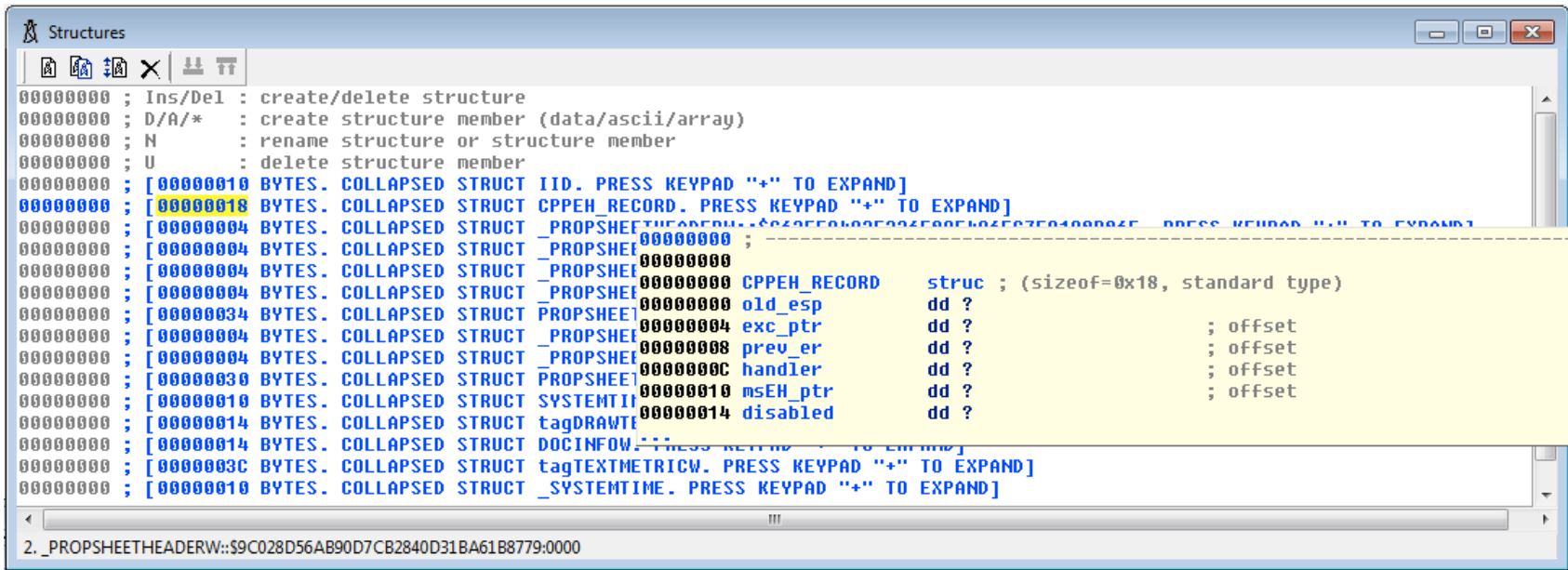


The Exports dialog box shows a list of exported functions. Each entry includes a small icon of a document with a red arrow, the function name, the address, and the ordinal. The status bar at the bottom right has a small icon.

Name	Address	Ordinal
start	00401090	

Structures

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



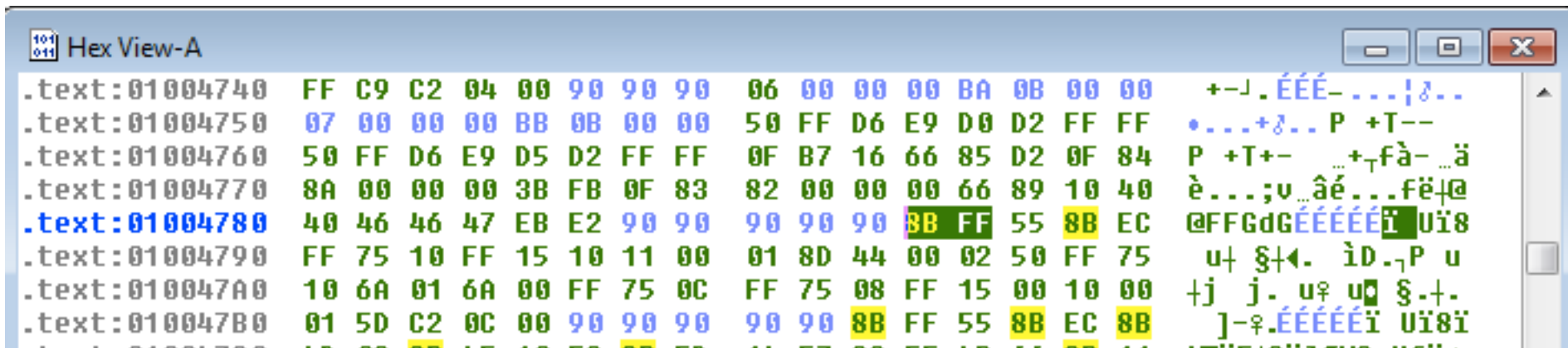
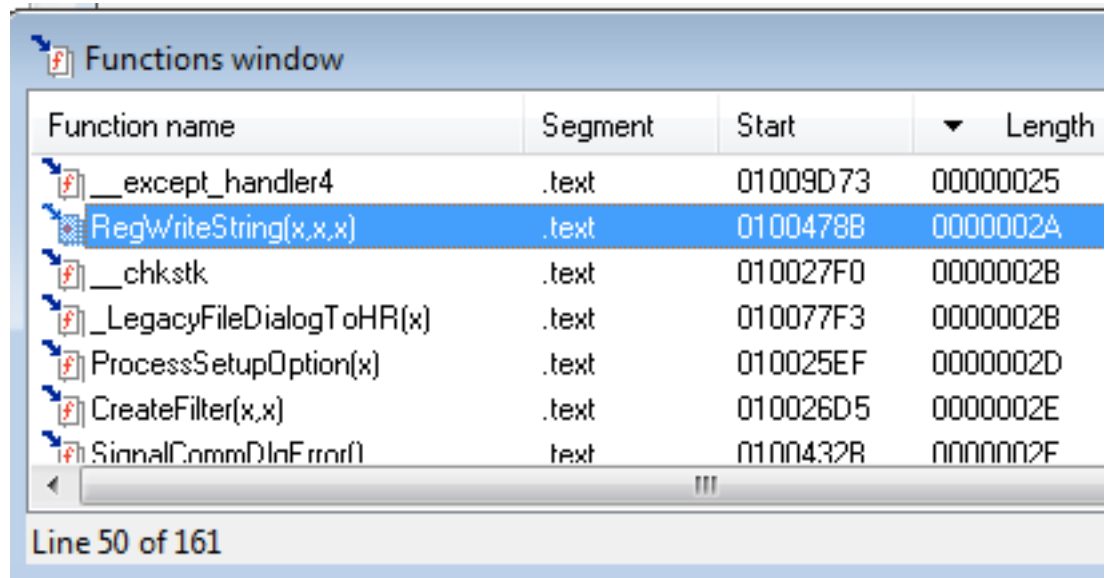
```
00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/* : create structure member (data/ascii/array)
00000000 ; N : rename structure or structure member
00000000 ; U : delete structure member
00000000 ; [00000010 BYTES. COLLAPSED STRUCT IID. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT CPPEH_RECORD. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000034 BYTES. COLLAPSED STRUCT PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000030 BYTES. COLLAPSED STRUCT PROPSHEET_HEADER. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT SYSTEMTIME. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000014 BYTES. COLLAPSED STRUCT tagDRAWITEMEXTRA. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000014 BYTES. COLLAPSED STRUCT DOCINFOW. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [0000003C BYTES. COLLAPSED STRUCT tagTEXTMETRICW. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _SYSTEMTIME. PRESS KEYPAD "+" TO EXPAND]
```

```
00000000 CPPEH_RECORD struct ; (sizeof=0x18, standard type)
00000000 old_esp dd ?
00000004 exc_ptr dd ? ; offset
00000008 prev_er dd ? ; offset
0000000C handler dd ? ; offset
00000010 msEH_ptr dd ? ; offset
00000014 disabled dd ?
```

2. _PROPSHEETHEADERW::9C028D56AB90D7CB2840D31BA61B8779:0000

Cross-Reference

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



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)

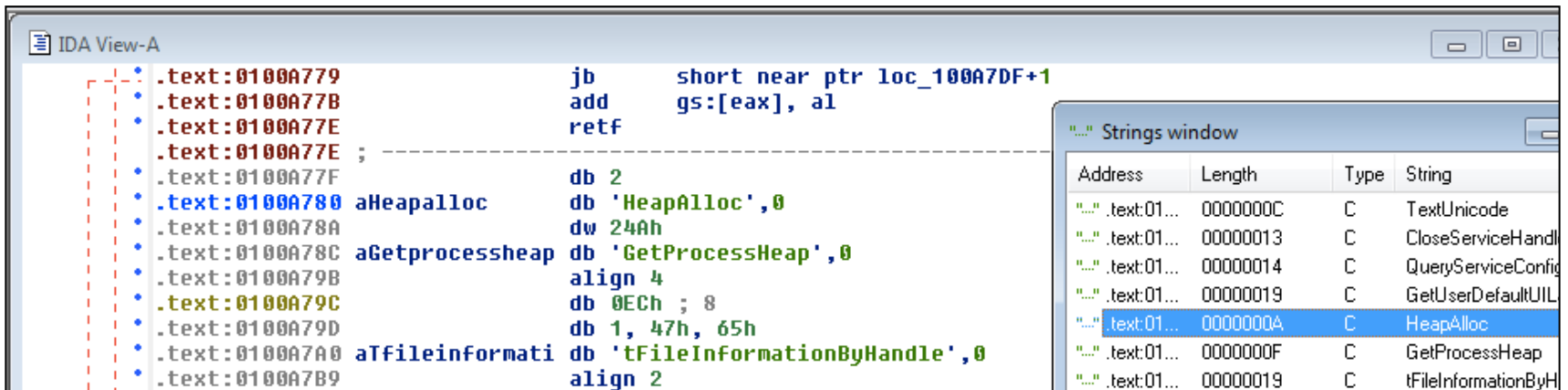
Returning to the Default View

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

Navigating IDA Pro

Imports or Strings

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



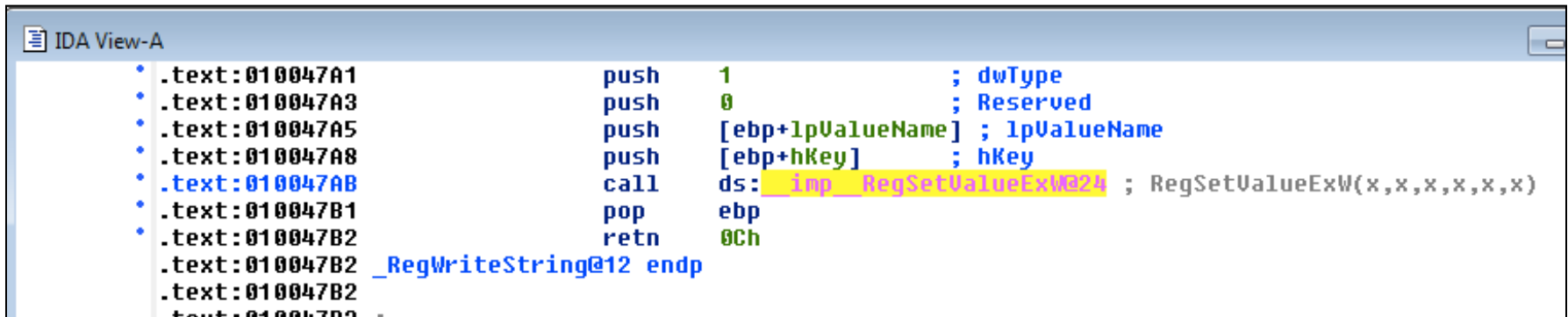
The screenshot shows the IDA View-A window with assembly code on the left and a Strings window on the right. The assembly code includes instructions like `jb short near ptr loc_100A7DF+1`, `add gs:[eax], al`, `retf`, `db 2`, `db 'HeapAlloc',0`, `dw 24Ah`, `db 'GetProcessHeap',0`, `align 4`, `db 0ECh ; 8`, `db 1, 47h, 65h`, `db 'tFileInformationByHandle',0`, and `align 2`. The Strings window on the right lists various strings with their addresses, lengths, and types. The string 'HeapAlloc' is highlighted in blue.

```
.text:0100A779      jb      short near ptr loc_100A7DF+1
.text:0100A77B      add     gs:[eax], al
.text:0100A77E      retf
.text:0100A77E      ; -----
.text:0100A77F      db     2
.text:0100A780  aHeapAlloc      db     'HeapAlloc',0
.text:0100A78A      dw     24Ah
.text:0100A78C  aGetprocessheap  db     'GetProcessHeap',0
.text:0100A79B      align  4
.text:0100A79C      db     0ECh ; 8
.text:0100A79D      db     1, 47h, 65h
.text:0100A7A0  aTfileinformati  db     'tFileInformationByHandle',0
.text:0100A7B9      align  2
```

Address	Length	Type	String
\".text:01...	0000000C	C	TextUnicode
\".text:01...	00000013	C	CloseServiceHandl
\".text:01...	00000014	C	QueryServiceConfig
\".text:01...	00000019	C	GetUserDefaultUIL
\".text:01...	0000000A	C	HeapAlloc
\".text:01...	0000000F	C	GetProcessHeap
\".text:01...	00000019	C	tFileInformationByH

Using Links

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

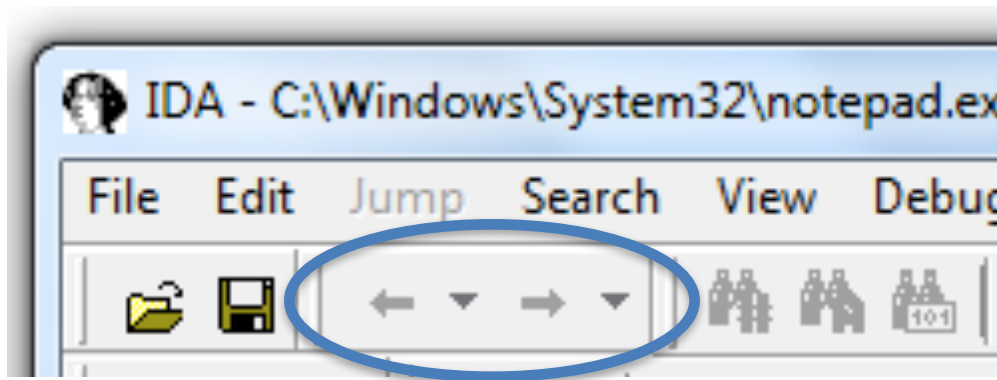


The screenshot shows the IDA View-A window with a list of assembly instructions. The instruction at address 010047AB is highlighted in yellow. The instruction is a call to the Windows API function RegSetValueExW, with the address 010047A4 highlighted in the instruction itself.

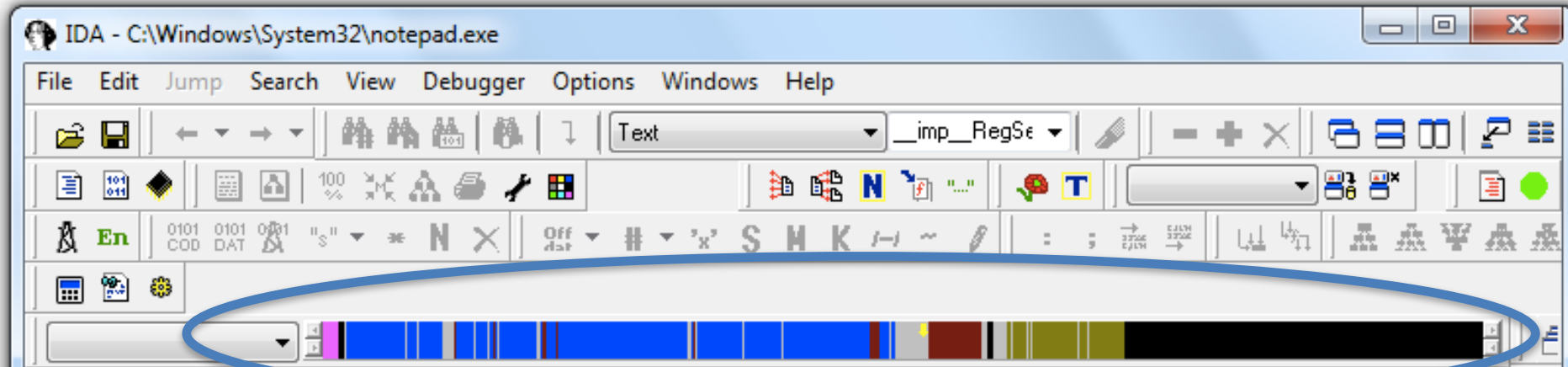
```
IDA View-A
• .text:010047A1      push    1           ; dwType
• .text:010047A3      push    0           ; Reserved
• .text:010047A5      push    [ebp+lpValueName] ; lpValueName
• .text:010047A8      push    [ebp+hKey]   ; hKey
• .text:010047AB      call   ds:010047A4 ; RegSetValueExW(x,x,x,x,x,x)
• .text:010047B1      pop     ebp
• .text:010047B2      retn   0Ch
.text:010047B2      _RegWriteString@12 endp
.text:010047B2
.text:010047B2
```


History

- Forward and Back buttons work like a Web browser



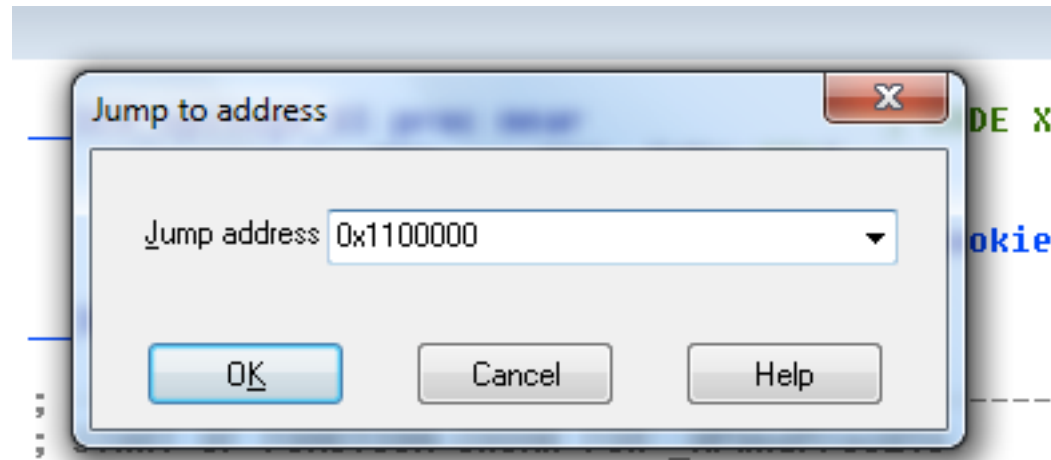
Navigation Band



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

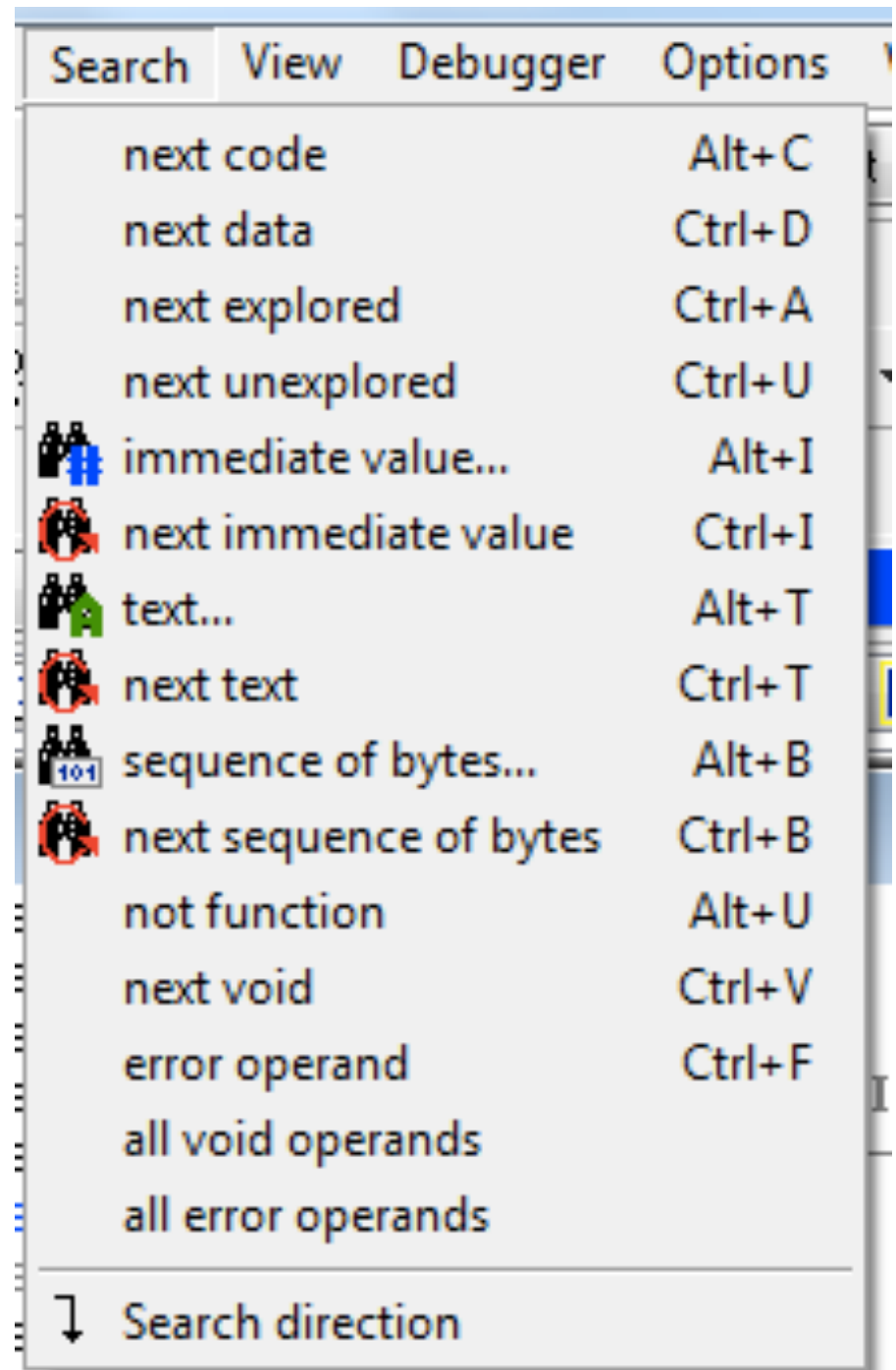
Jump to Location

- Press **G**
- Can jump to address or named location



Searching

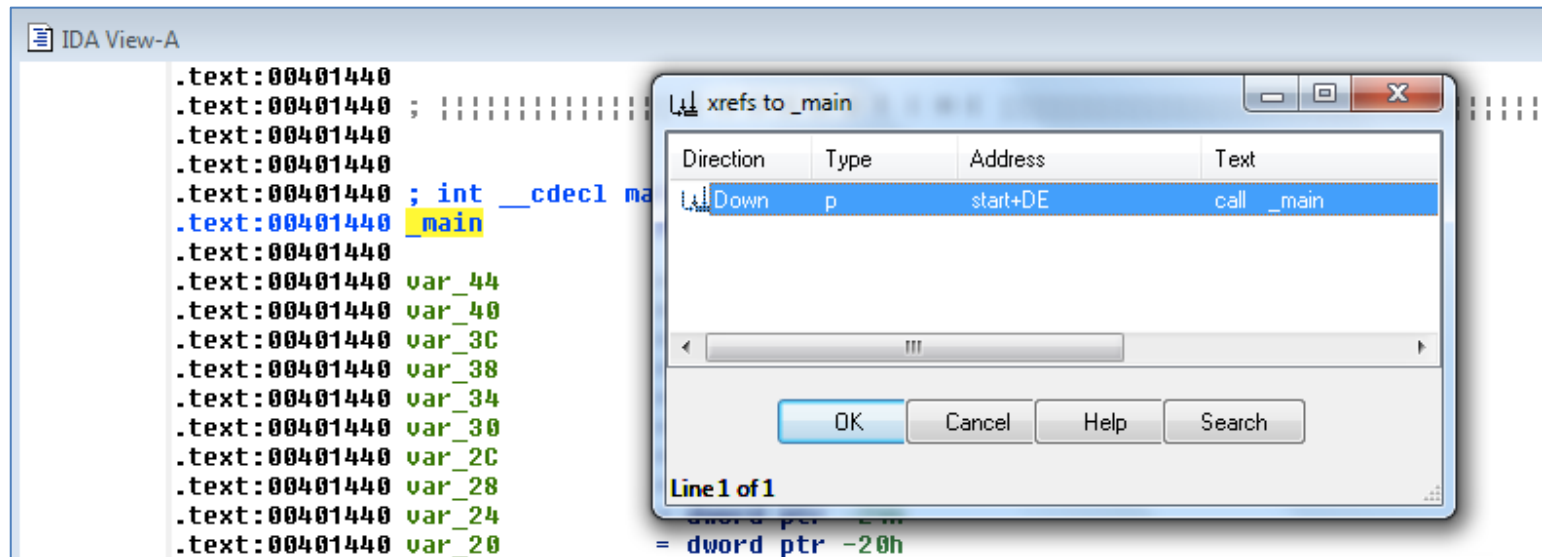
- Many options
- Search, Text is handy



Using Cross-References

To See All Cross-References

- Click function name and press X



Analyzing Functions

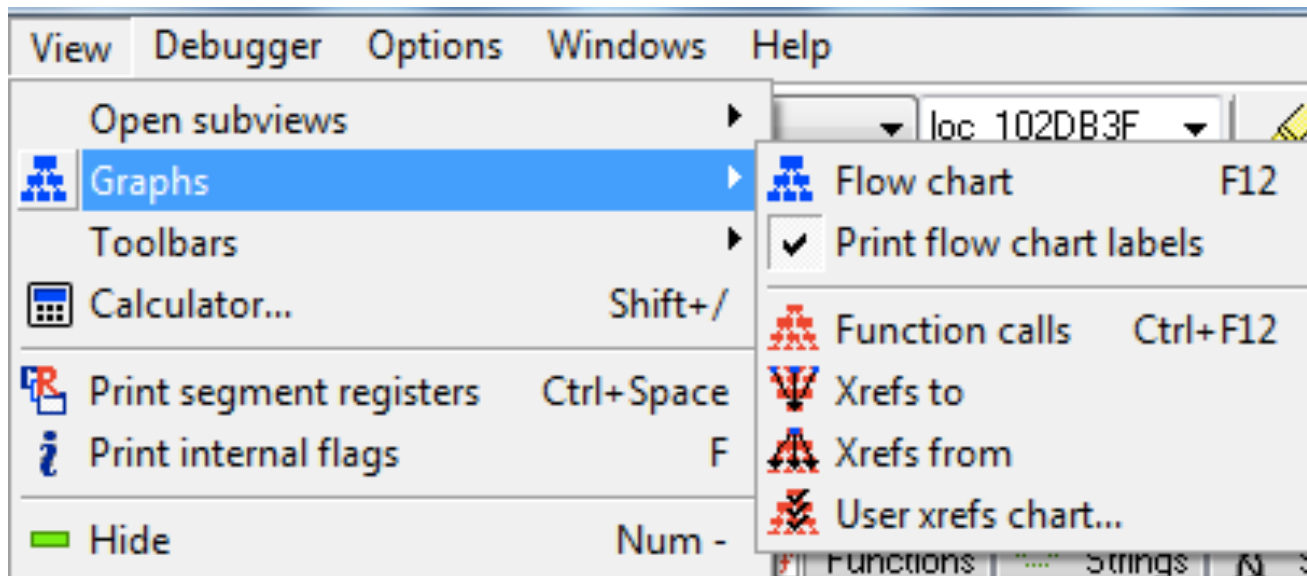
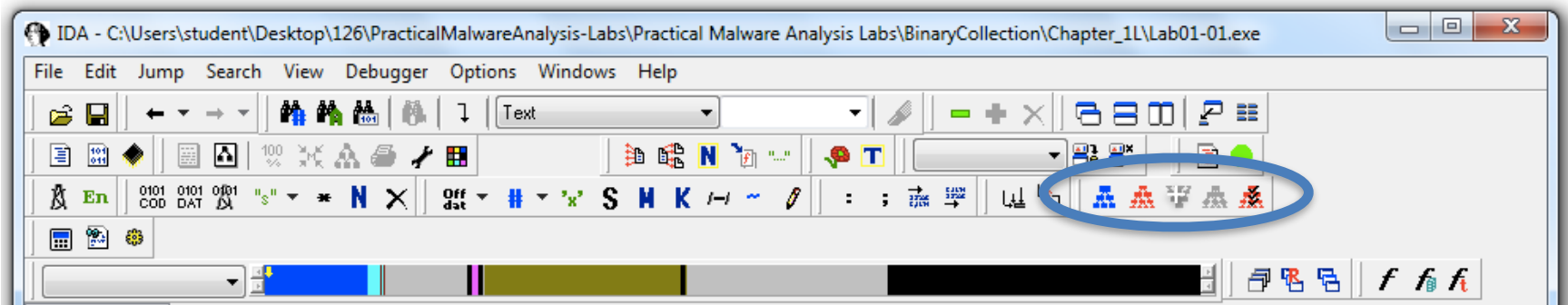
Function and Argument Recognition

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

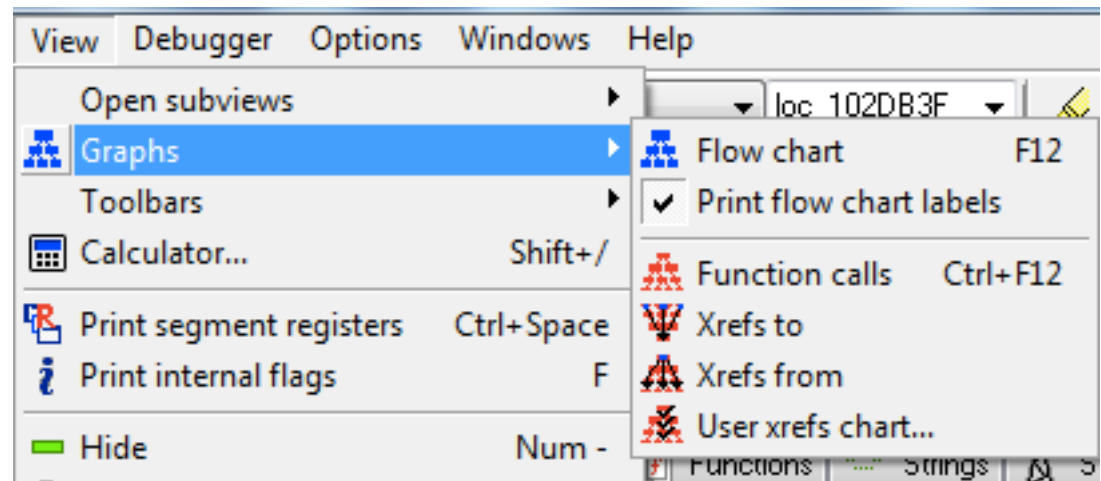
```
IDA View-A
.text:00401040
.text:00401040
.text:00401040 sub_401040      proc near          ; CODE XREF: sub_4010A0+88↓p
.text:00401040                                     ; sub_4010A0+B7↓p ...
.text:00401040
.text:00401040 arg_0          = dword ptr  4
.text:00401040 arg_4          = dword ptr  8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

Using Graphing Options

Graphing Options

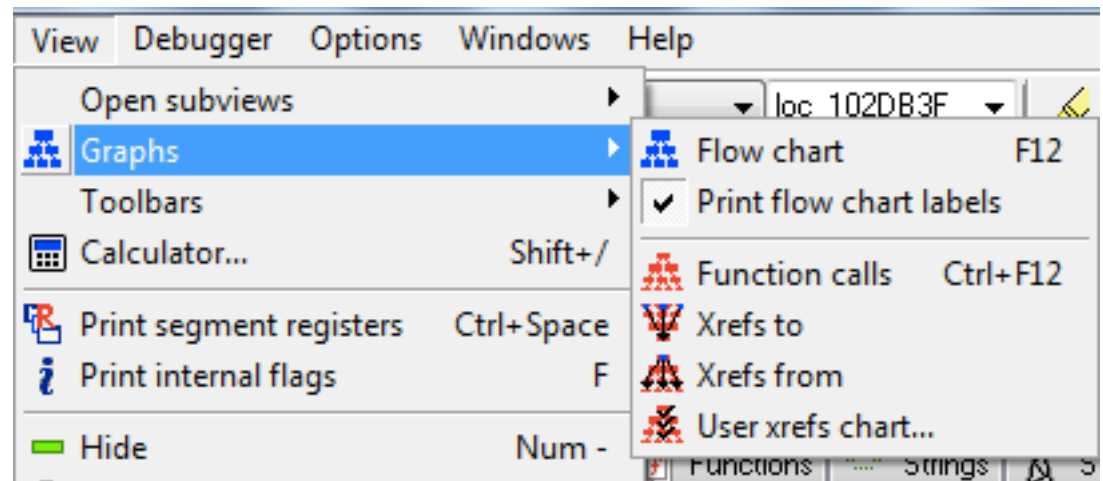


Graphing Options



- 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

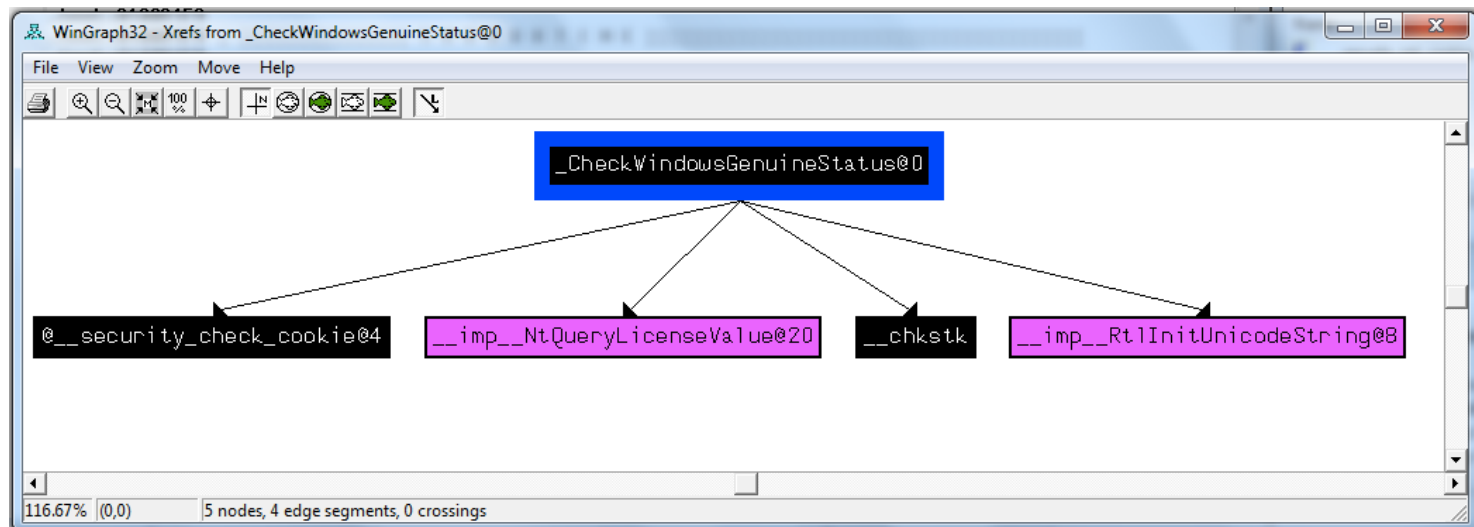
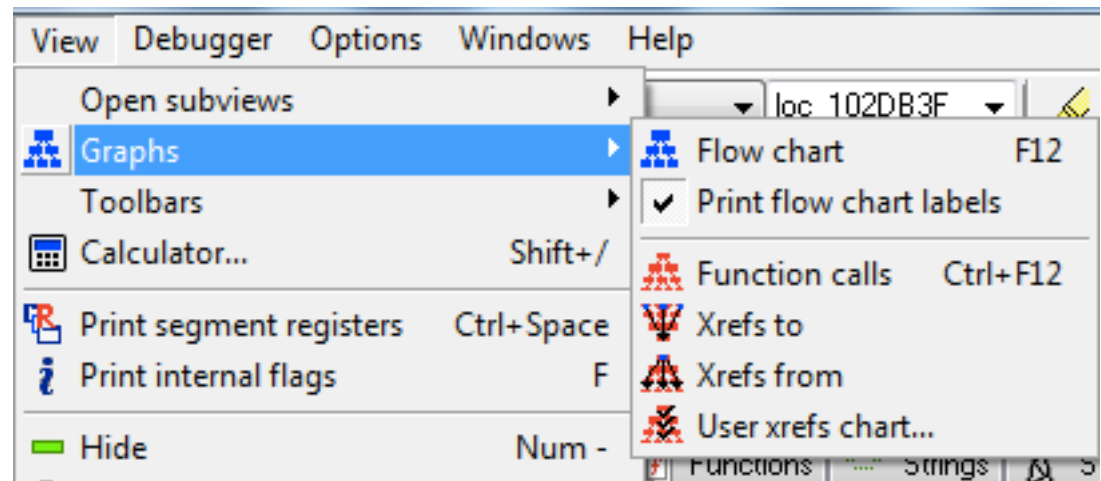


- **Xrefs to**
 - Graphs XREFs to get to selected XREF
 - Can show all the paths that get to a function

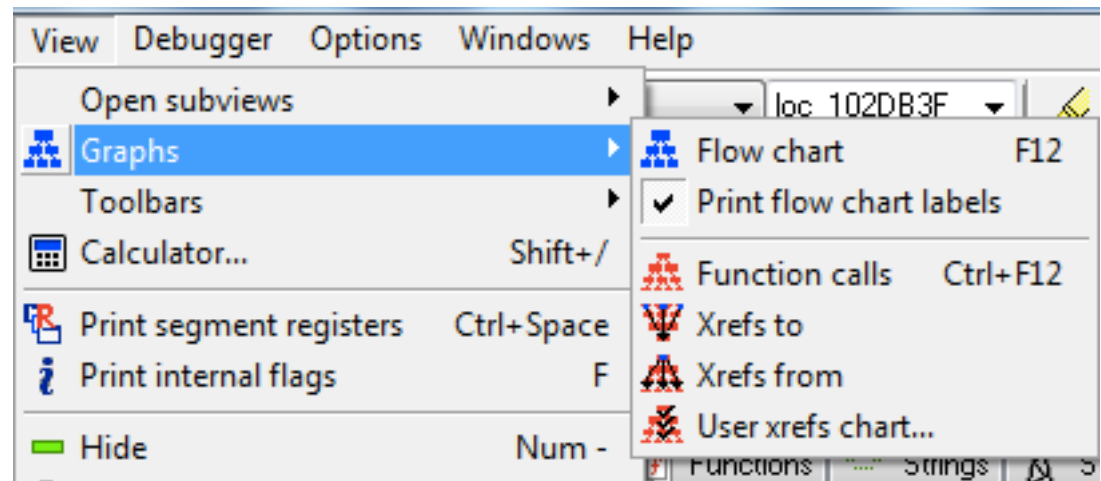
Graphing Options

- **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

Enhancing Disassembly

Warning

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

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

With renamed arguments

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

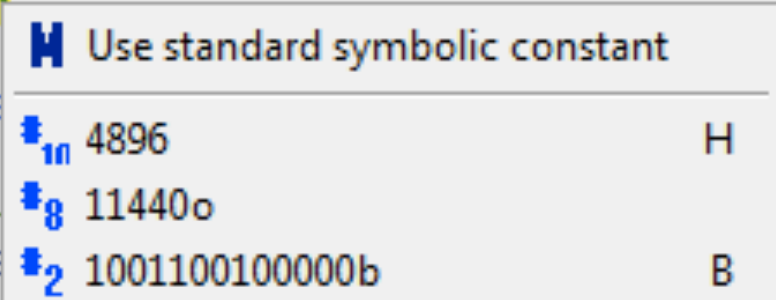
Comments

- 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
```



Format	Value	Symbolic Constant
Use standard symbolic constant		
Decimal	4896	H
Octal	11440o	
Binary	1001100100000b	B

Using Named Constants

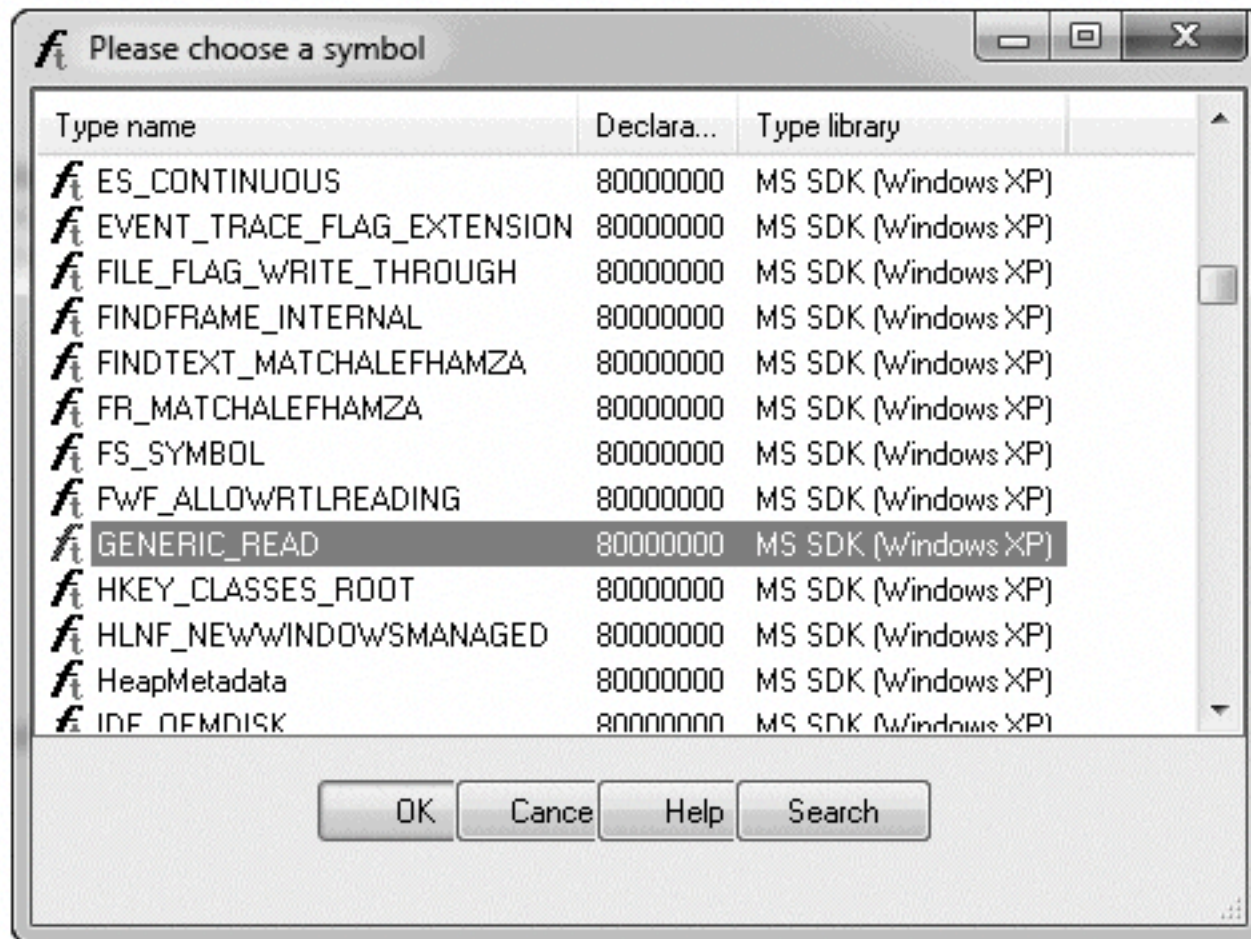


Figure 6-11. Standard symbolic constant window

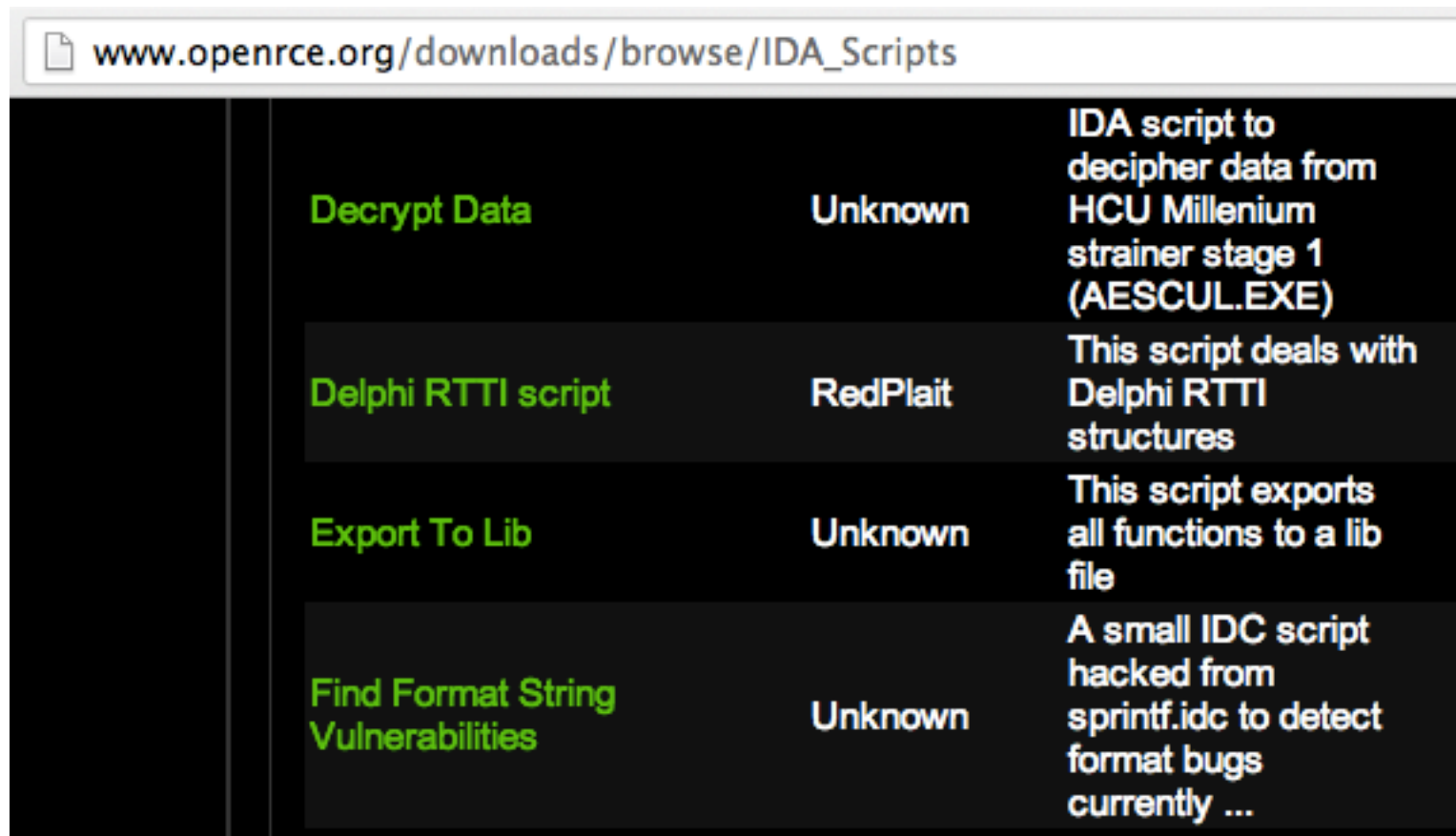
Using Named Constants

- 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>

Extending IDA with Plug-ins

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



The screenshot shows a web browser window with the address bar containing the URL www.openrce.org/downloads/browse/IDA_Scripts. Below the address bar is a table listing four IDC scripts. The table has three columns: the script name, the author, and a description of the script's functionality.

Script Name	Author	Description
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 ...

Kahoot!