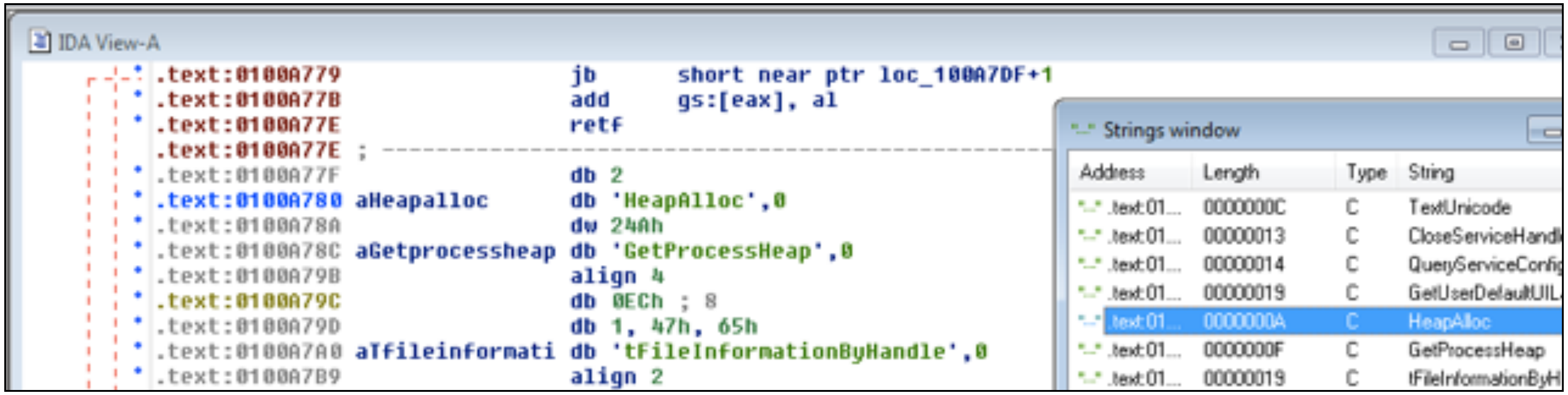
Navigating IDA Pro

Imports or Strings

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



IDA View-A

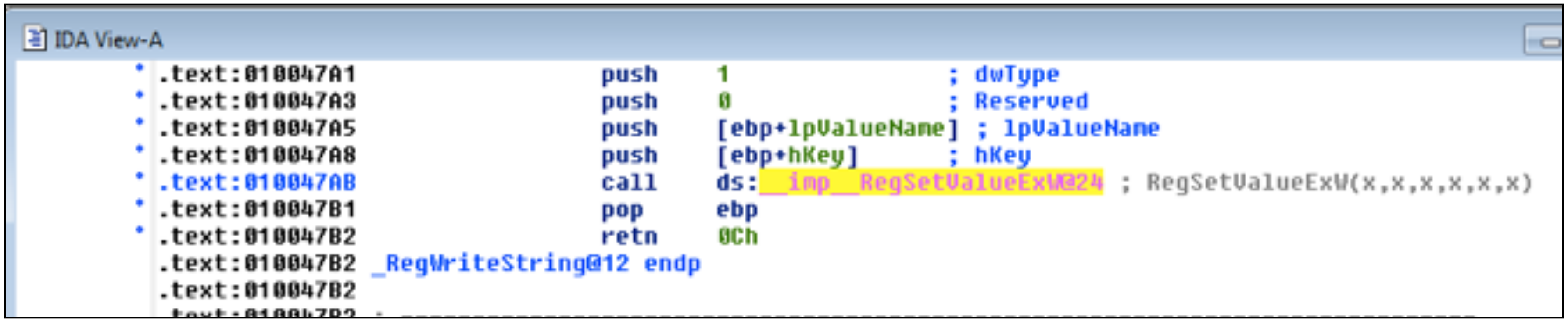
```
.text:0100A779                 jb      short near ptr loc_100A7DF+1
.text:0100A77B                 add     gs:[eax], al
.text:0100A77E                 retf
.text:0100A77E ; ---------------------------------------
.text:0100A77F                 db 2
.text:0100A780 aHeapalloc      db 'HeapAlloc',0
.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
```

Strings window

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

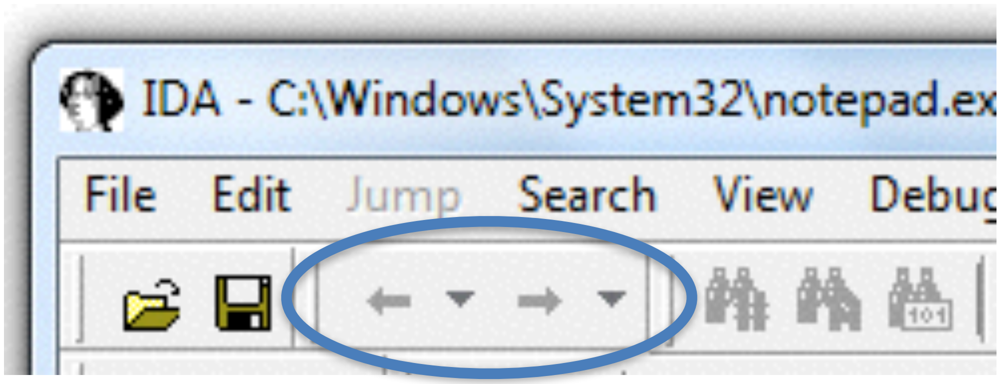


IDA View-A

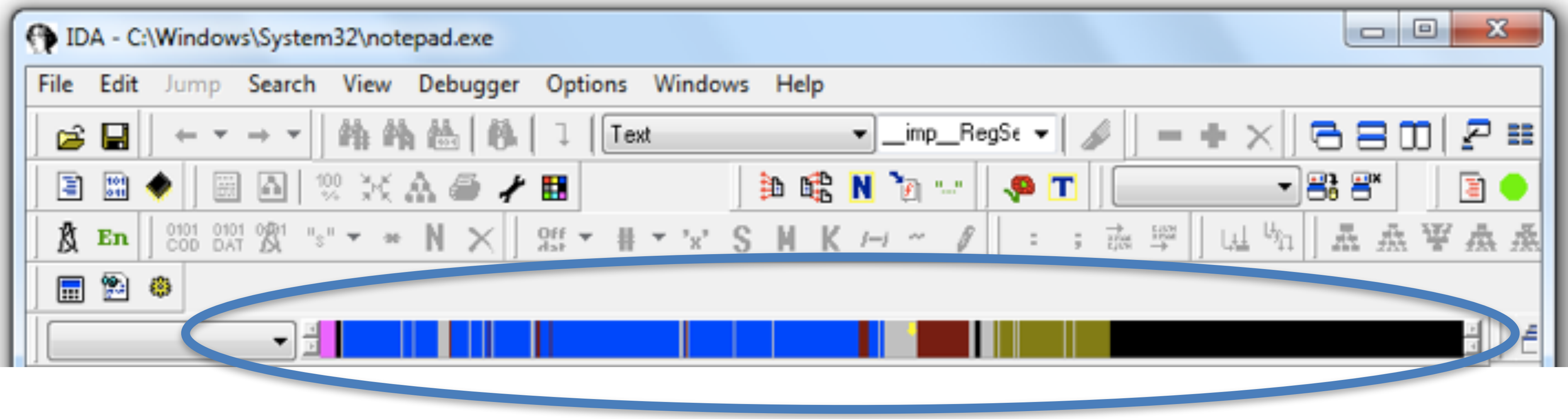
```
.text:010047A1                 push    1                ; dwType
.text:010047A3                 push    0                ; Reserved
.text:010047A5                 push    [ebp+lpValueName] ; lpValueName
.text:010047A8                 push    [ebp+hKey]       ; hKey
.text:010047AB                 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
```

History

- Forward and Back buttons work like a Web browser



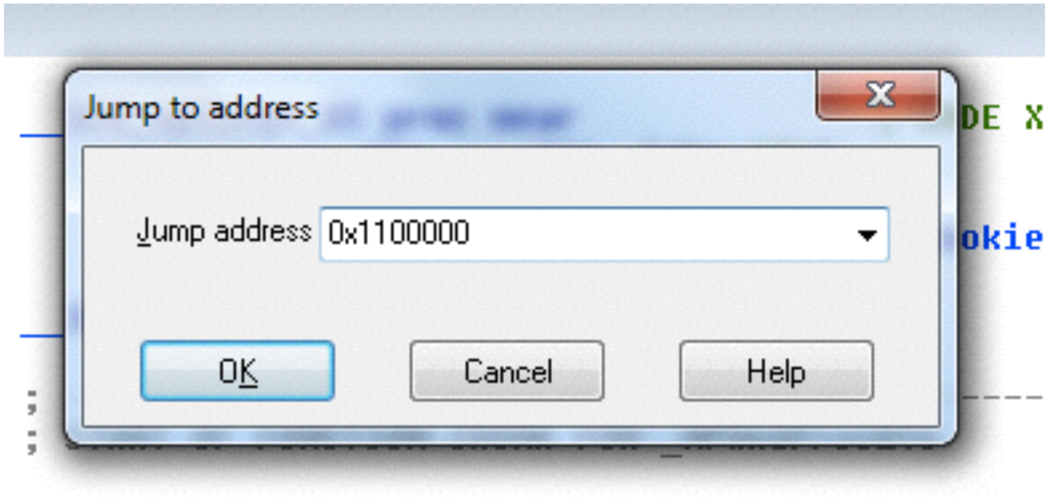
Navigation Band



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

Jump to Location

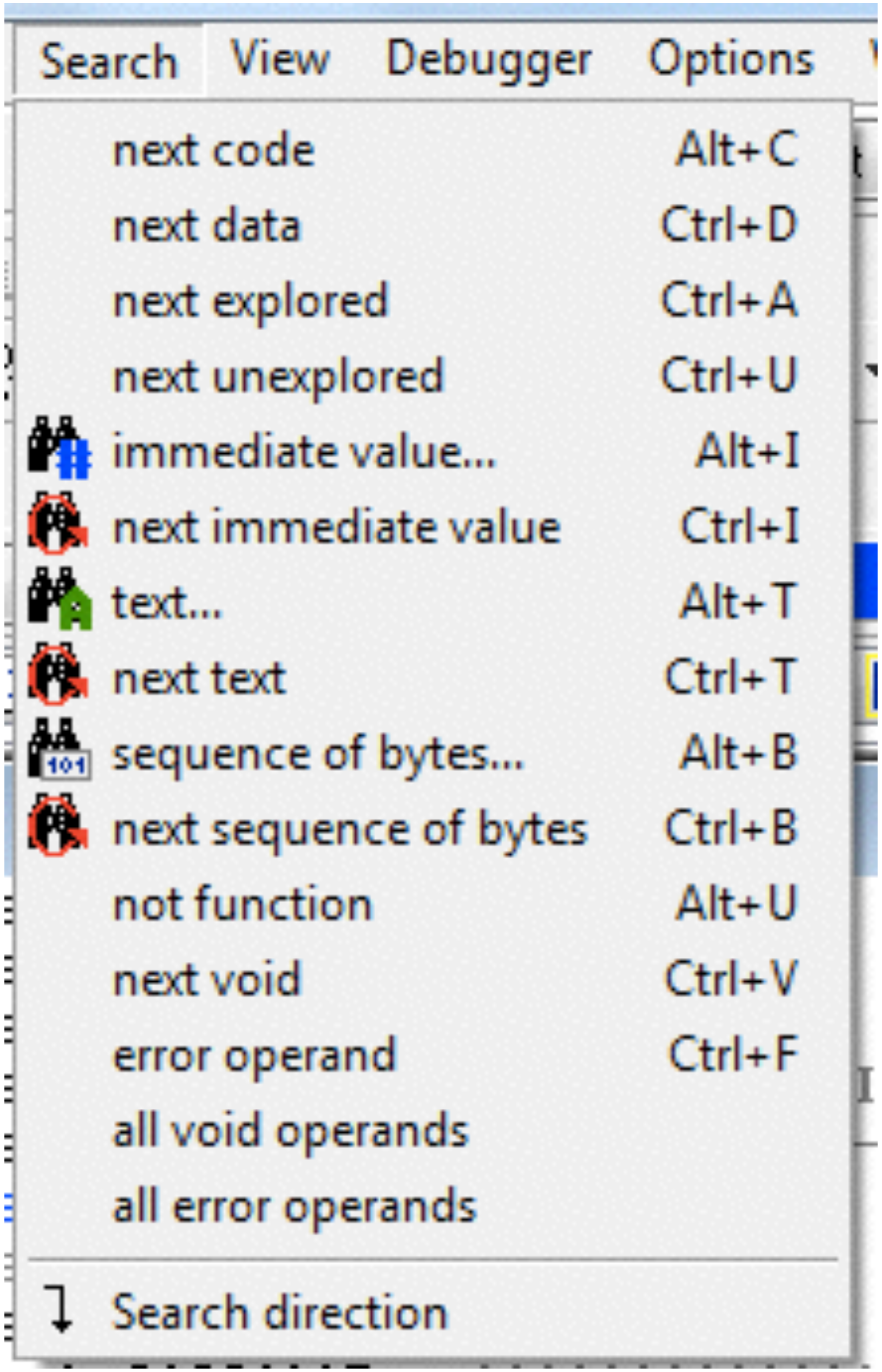
- Press **G**
- Can jump to address or named location



Jump address 0x1100000

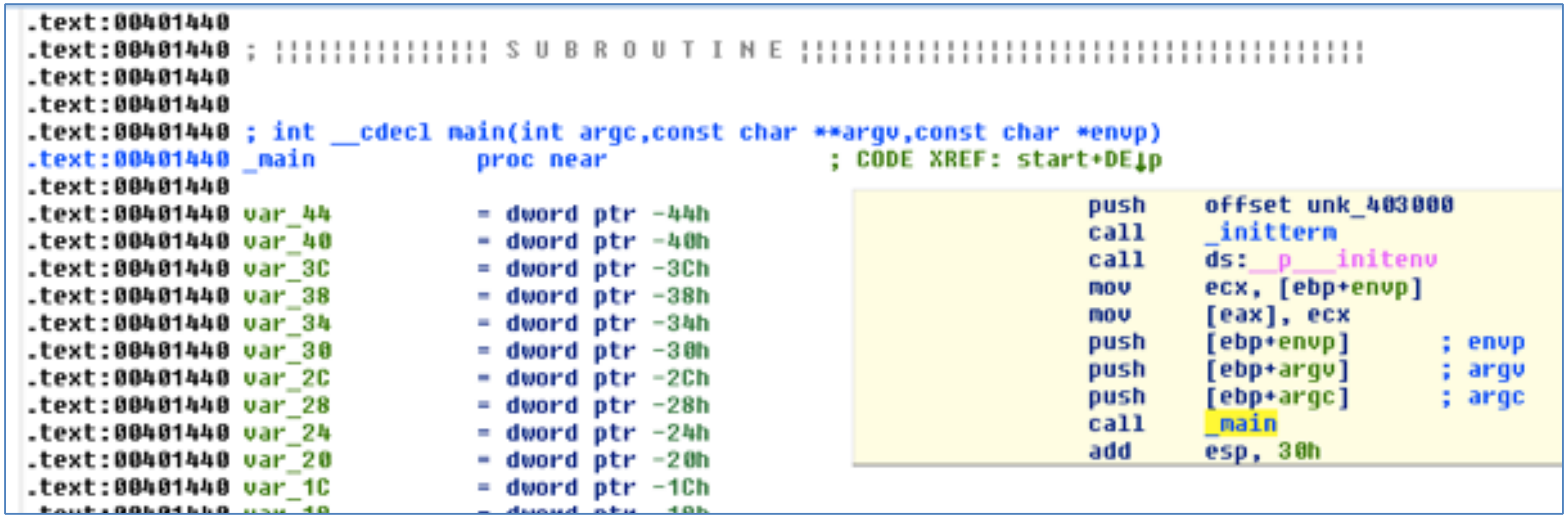
Searching

- Many options
- Search, Text is handy



Using Cross-References

Code Cross-References



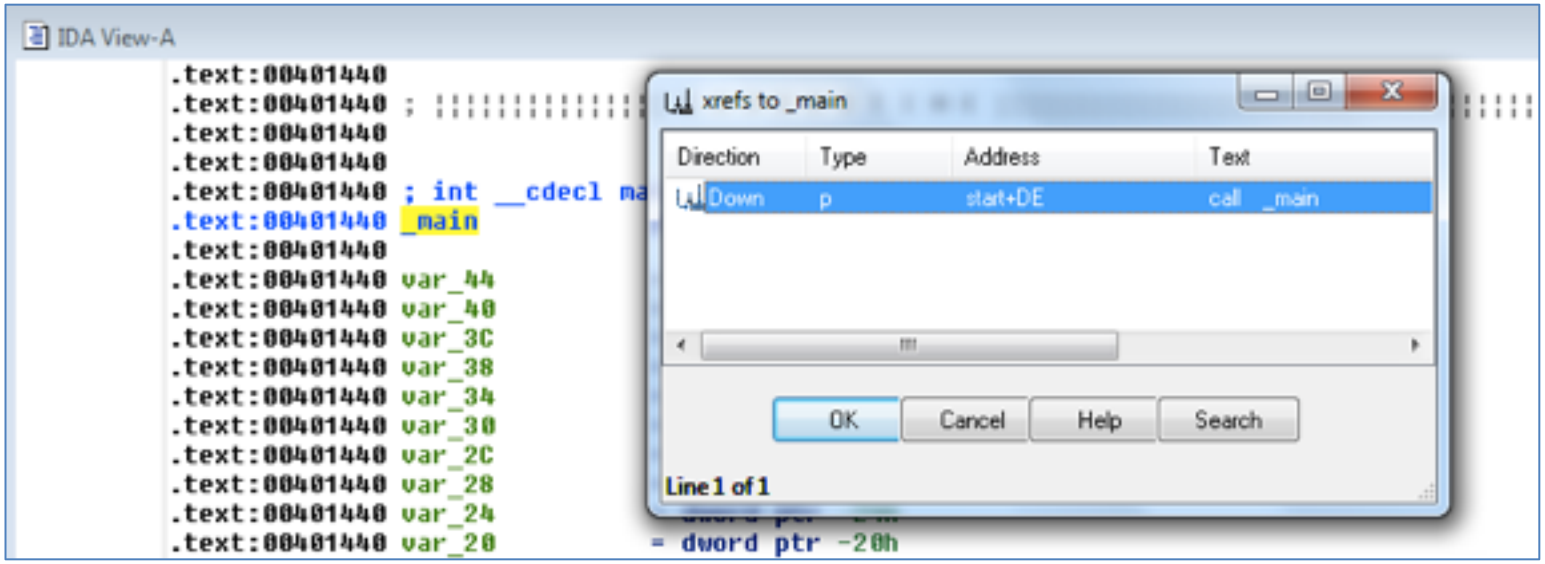
```
.text:00401440
.text:00401440 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||||
.text:00401440
.text:00401440
.text:00401440 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401440 _main           proc near               ; CODE XREF: start+DE↓p
.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
```

```
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



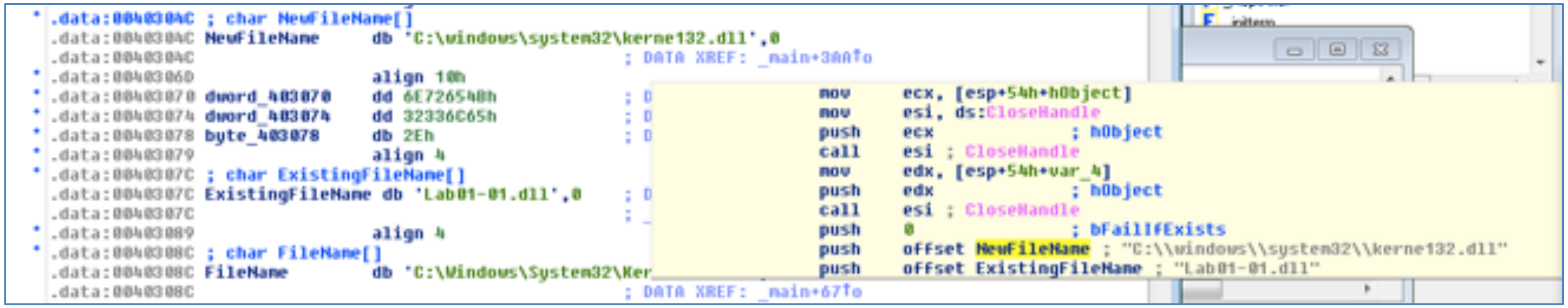
```
.text:00401440
.text:00401440 ; ||||||||||||||
.text:00401440
.text:00401440
.text:00401440 ; int __cdecl ma
.text:00401440 _main
.text:00401440
.text:00401440 var_44
.text:00401440 var_40
.text:00401440 var_3C
.text:00401440 var_38
.text:00401440 var_34
.text:00401440 var_30
.text:00401440 var_2C
.text:00401440 var_28
.text:00401440 var_24
.text:00401440 var_20          = dword ptr -20h
```

xrefs to _main

Direction	Type	Address	Text
Down	p	start+DE	call _main

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



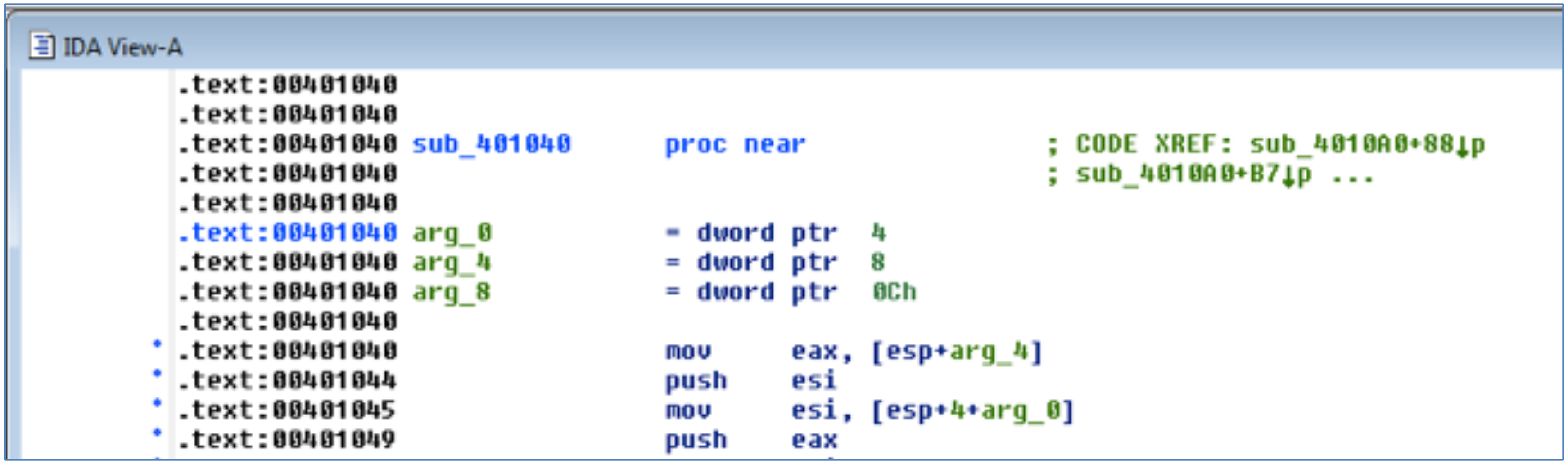
```
.data:0040304C ; char NewFileName[]
.data:0040304C NewFileName     db 'C:\windows\system32\kerne132.dll',0
.data:0040304C                                          ; DATA XREF: _main+3AA↑o
.data:0040306D                 align 10h
.data:00403070 dword_403070    dd 6E726540h             ; D
.data:00403074 dword_403074    dd 32336C65h             ; D
.data:00403078 byte_403078     db 2Eh                   ; D
.data:00403079                 align 4
.data:0040307C ; char ExistingFileName[]
.data:0040307C ExistingFileName db 'Lab01-01.dll',0     ; D
.data:0040307C                                          ; _
.data:00403089                 align 4
.data:0040308C ; char FileName[]
.data:0040308C FileName        db 'C:\Windows\System32\Ker
.data:0040308C                                          ; DATA XREF: _main+67↑o
```

```
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"
```

Analyzing Functions

Function and Argument Recognition

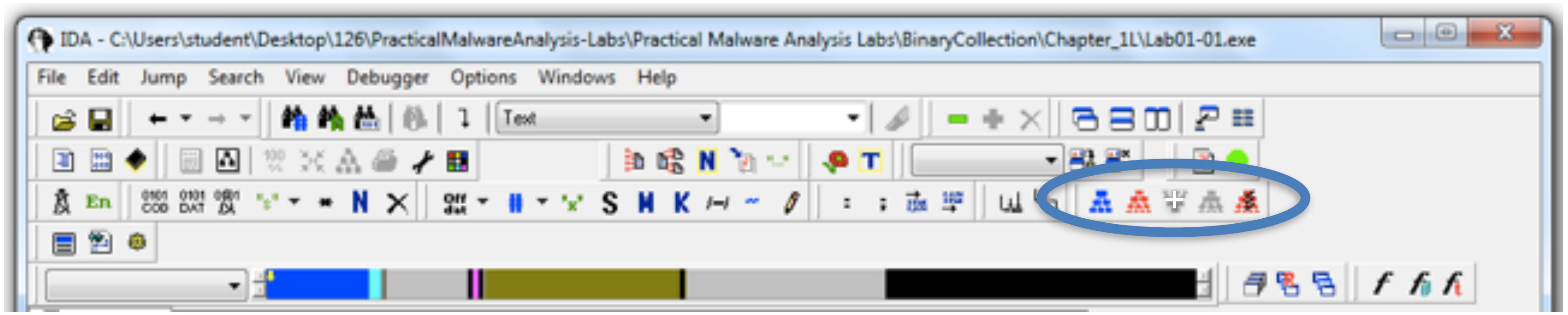
- IDA Pro identifies a function, names it, and also names the local variables
- It's not always correct

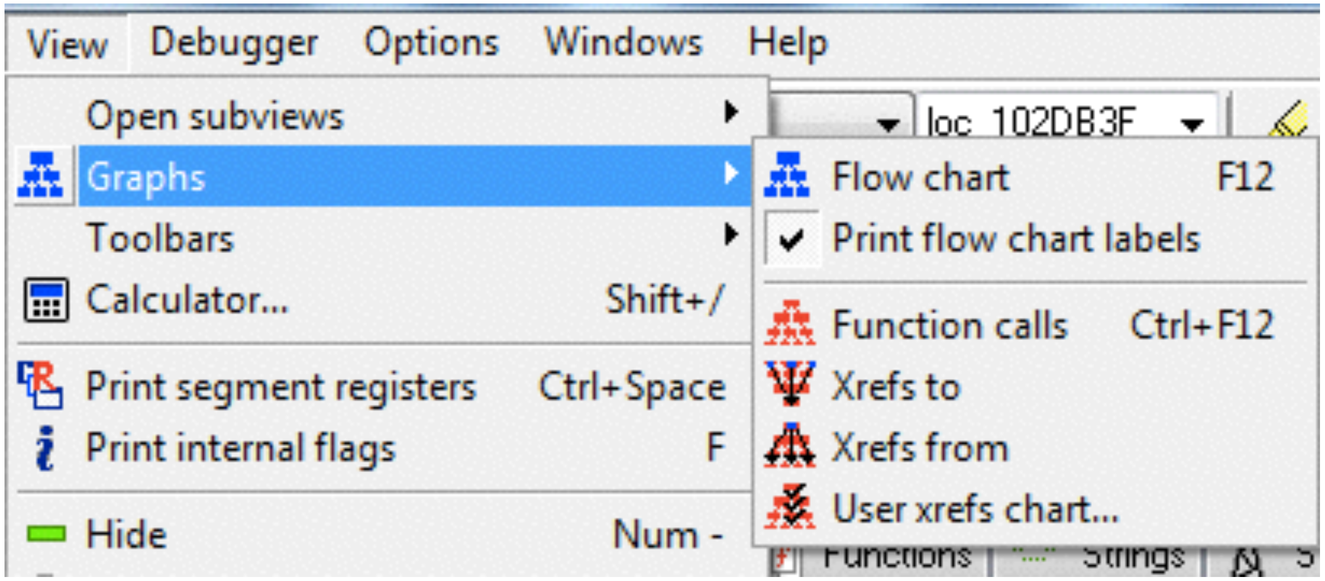


```
.text:00401040
.text:00401040
.text:00401040 sub_401040      proc near               ; CODE XREF: sub_4010A0+88↓p
.text:00401040                                         ; sub_4010A0+B7↓p ...
.text:00401040
.text:00401040 arg_0           = dword ptr  4
.text:00401040 arg_4           = dword ptr  8
.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
```

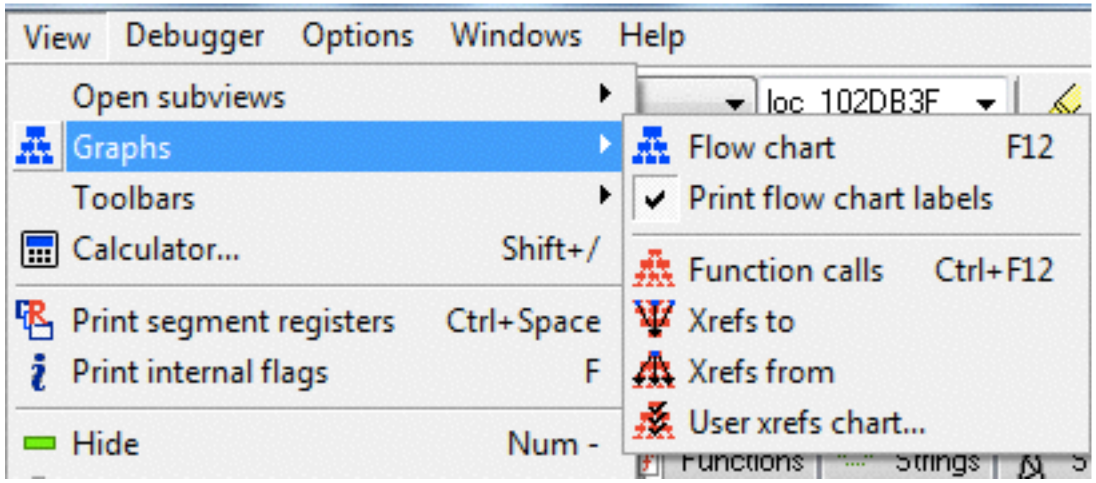
Using Graphing Options

Graphing Options


IDA - C:\Users\student\Desktop\126\PracticalMalwareAnalysis-Labs\Practical Malware Analysis Labs\BinaryCollection\Chapter_1L\Lab01-01.exe
File Edit Jump Search View Debugger Options Windows Help
Text

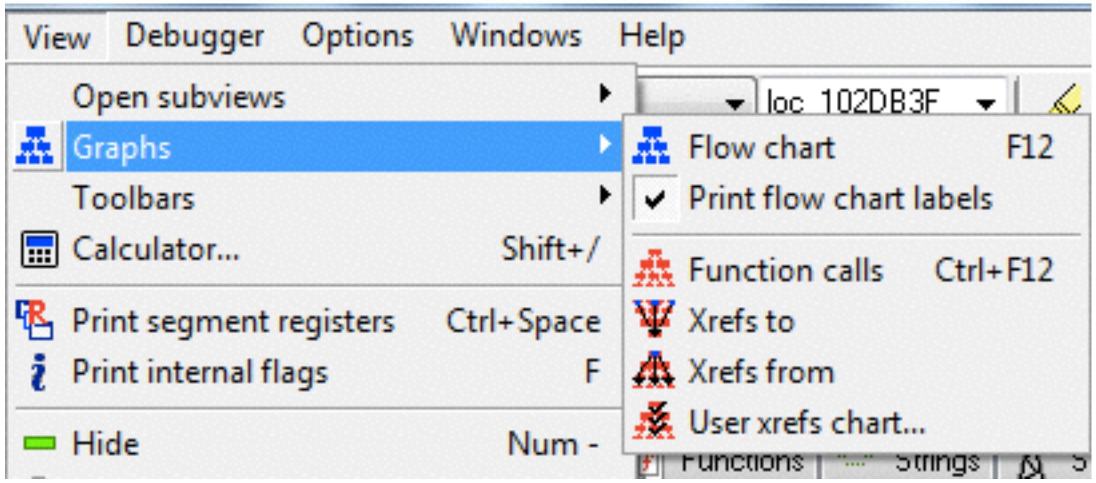


Graphing Options



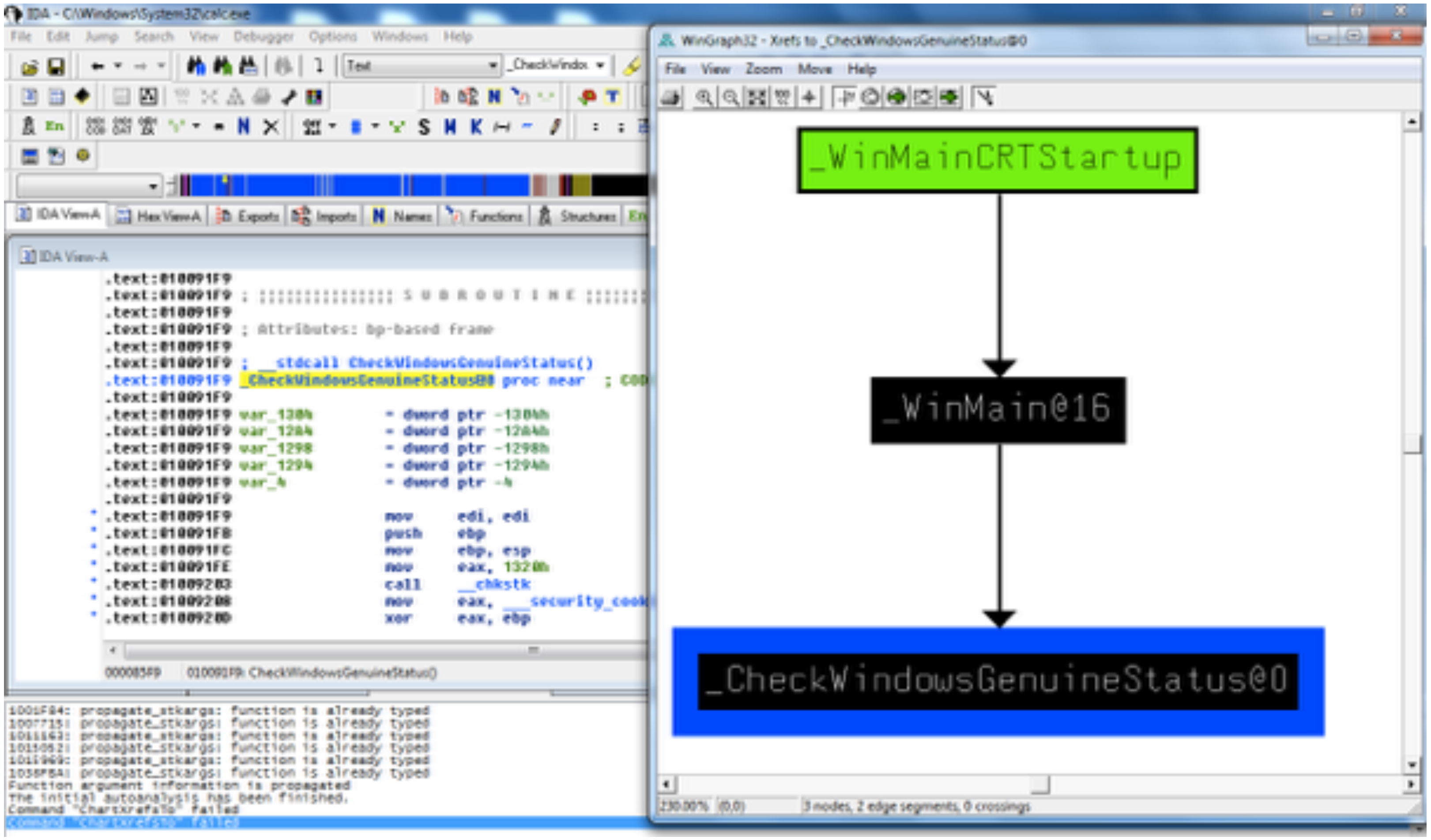
- These are "Legacy Graphs" and cannot be manipulated with IDA
- The first two seem obsolete
  - **Flow chart**
    - Create flow chart of current function
  - **Function calls**
    - Graph function calls for entire program

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

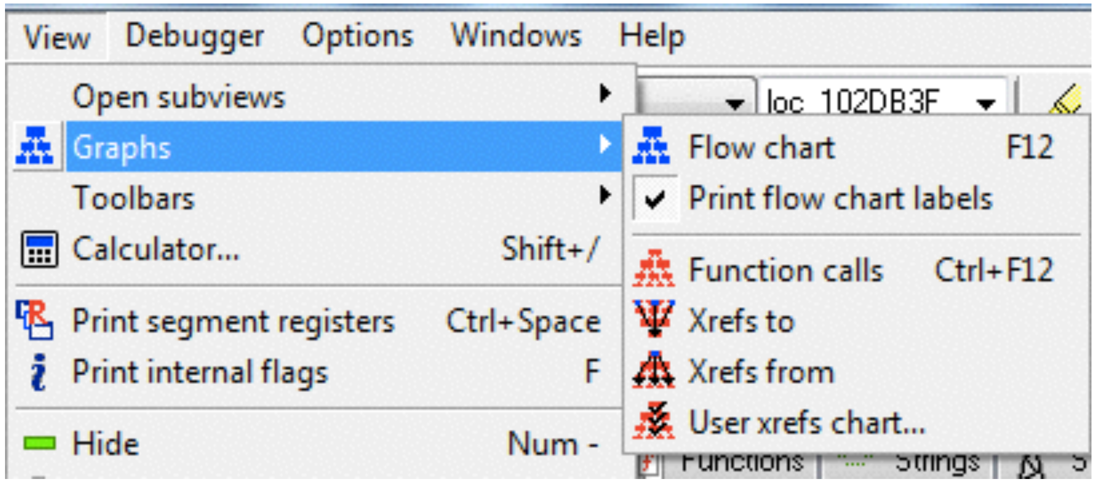

IDA - C:\Windows\System32\calc.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Structures
WinGraph32 - Xrefs to _CheckWindowsGenuineStatus@0
File View Zoom Move Help
_WinMainCRTStartup
_WinMain@16
_CheckWindowsGenuineStatus@0
230.00% (0,0) 3 nodes, 2 edge segments, 0 crossings

```
.text:010091F9
.text:010091F9 ; ::::::::::::::::::::: S U B R O U T I N E :::::::
.text:010091F9
.text:010091F9 ; Attributes: bp-based frame
.text:010091F9
.text:010091F9 ; __stdcall CheckWindowsGenuineStatus()
.text:010091F9 _CheckWindowsGenuineStatus@0 proc near   ; COD
.text:010091F9
.text:010091F9 var_1304          = dword ptr -1304h
.text:010091F9 var_12A4          = dword ptr -12A4h
.text:010091F9 var_1298          = dword ptr -1298h
.text:010091F9 var_1294          = dword ptr -1294h
.text:010091F9 var_4             = dword ptr -4
.text:010091F9
.text:010091F9                   mov     edi, edi
.text:010091FB                   push    ebp
.text:010091FC                   mov     ebp, esp
.text:010091FE                   mov     eax, 1320h
.text:01009203                   call    __chkstk
.text:01009208                   mov     eax, ___security_cook
.text:0100920D                   xor     eax, ebp
```

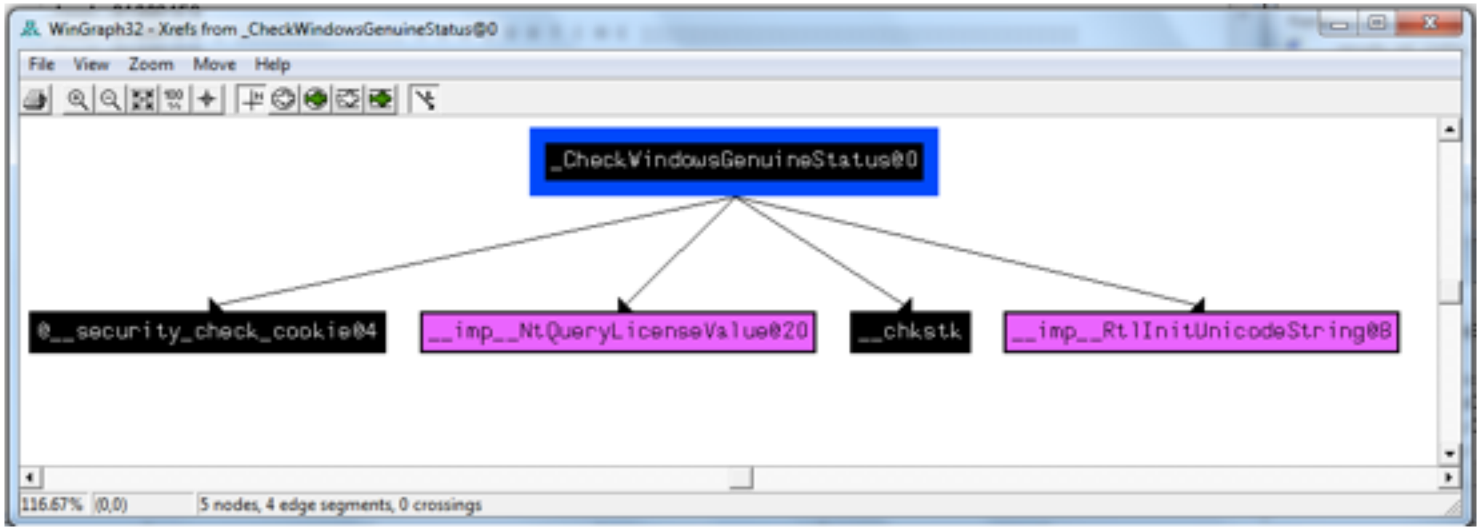
```
000085F9    010091F9: CheckWindowsGenuineStatus()
```

```
1001F84: propagate_stkargs: function is already typed
1007715: propagate_stkargs: function is already typed
1011163: propagate_stkargs: function is already typed
1015062: propagate_stkargs: function is already typed
1011969: propagate_stkargs: function is already typed
1038FBA: propagate_stkargs: function is already typed
Function argument information is propagated
The initial autoanalysis has been finished.
Command "ChartXrefsTo" failed
Command "ChartXrefsTo" failed
```

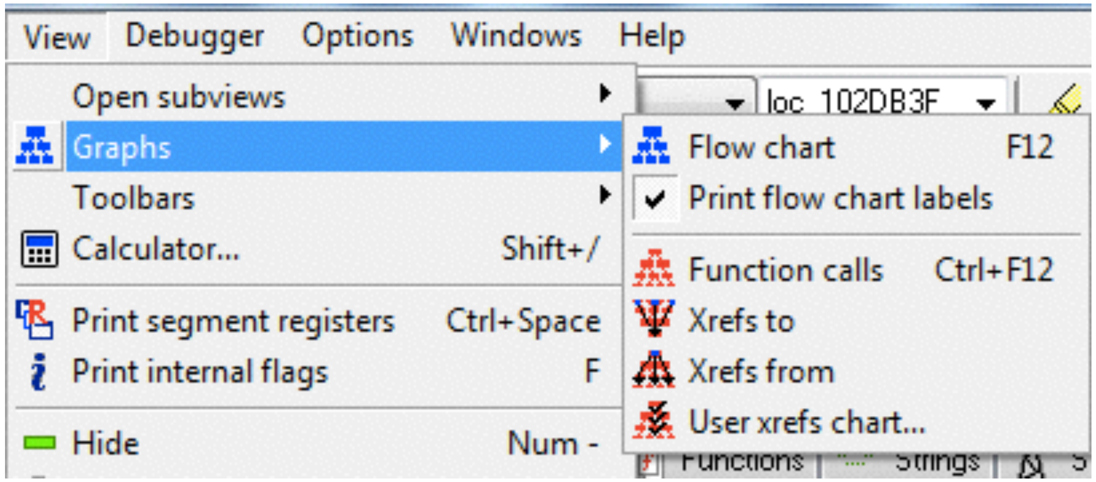
Graphing Options



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


WinGraph32 - Xrefs from _CheckWindowsGenuineStatus@0
File View Zoom Move Help
_CheckWindowsGenuineStatus@0
@__security_check_cookie@4
__imp__NtQueryLicenseValue@20
__chkstk
__imp__RtlInitUnicodeString@8
116.67% (0,0)
5 nodes, 4 edge segments, 0 crossings

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

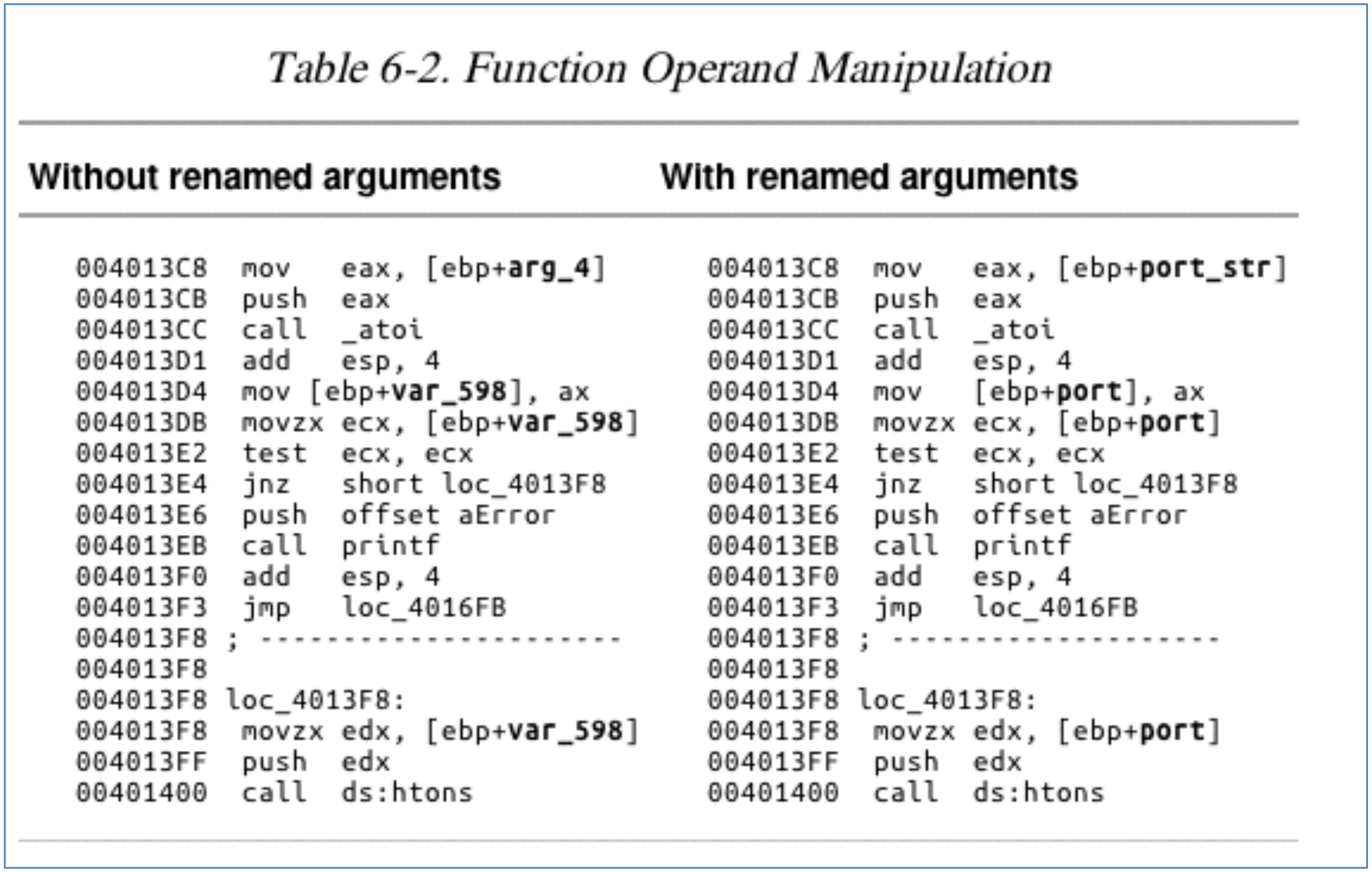


Table 6-2. Function Operand Manipulation

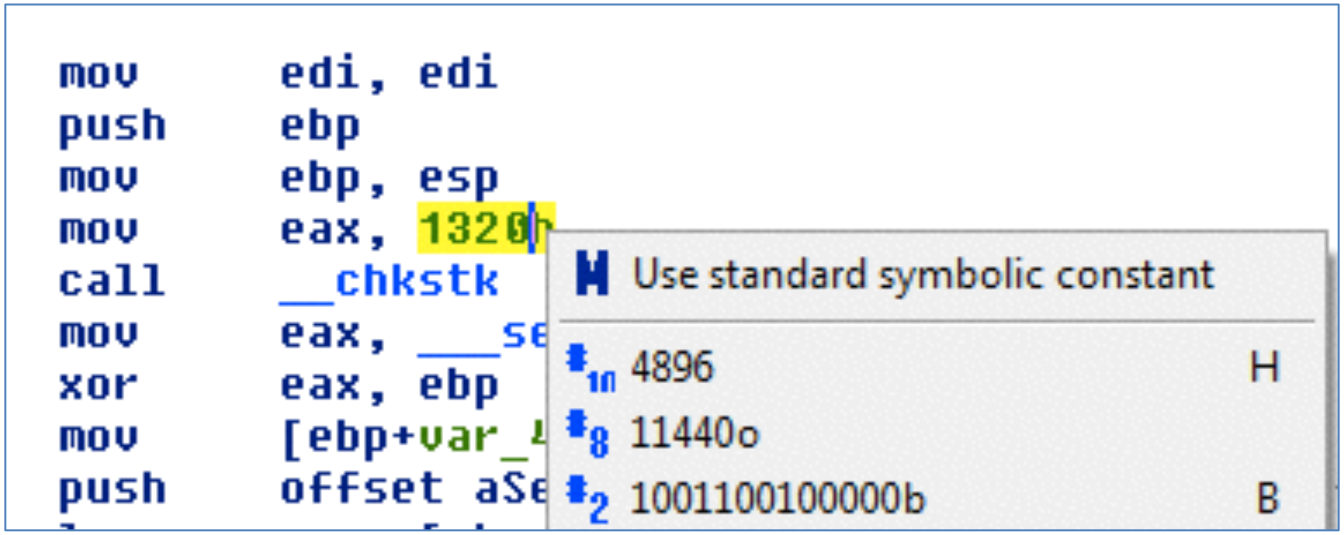
Without renamed arguments	With renamed arguments
004013C8 mov eax, [ebp+**arg_4**]	004013C8 mov eax, [ebp+**port_str**]
004013CB push eax	004013CB push eax
004013CC call _atoi	004013CC call _atoi
004013D1 add esp, 4	004013D1 add esp, 4
004013D4 mov [ebp+**var_598**], ax	004013D4 mov [ebp+**port**], ax
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

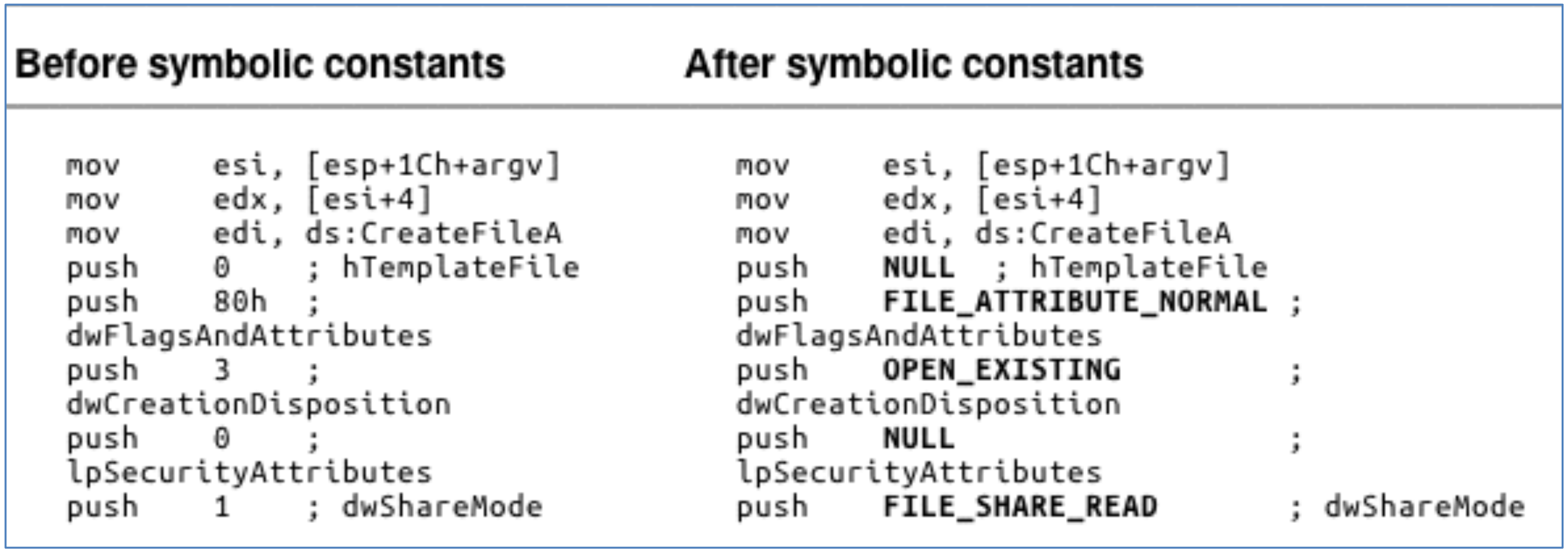
- Hexadecimal by default
- Right-click to use other formats



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

Using Named Constants

- Makes Windows API arguments clearer



Before symbolic constants

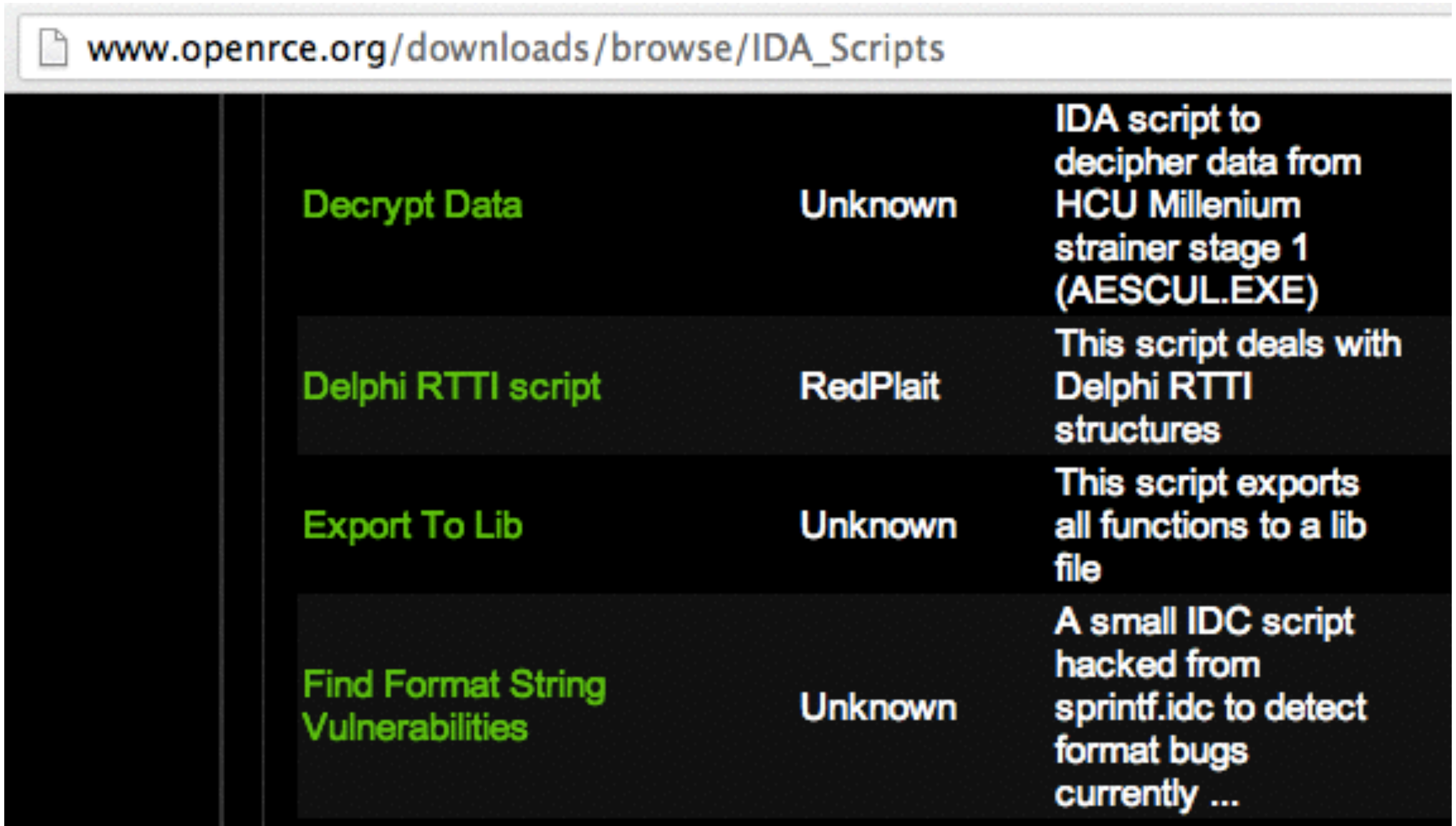
```
mov     esi, [esp+1Ch+argv]
mov     edx, [esi+4]
mov     edi, ds:CreateFileA
push    0    ; hTemplateFile
push    80h  ;
dwFlagsAndAttributes
push    3    ;
dwCreationDisposition
push    0    ;
lpSecurityAttributes
push    1    ; dwShareMode
```

After symbolic constants

```
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
```

Extending IDA with Plug-ins

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



www.openrce.org/downloads/browse/IDA_Scripts

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 ...