

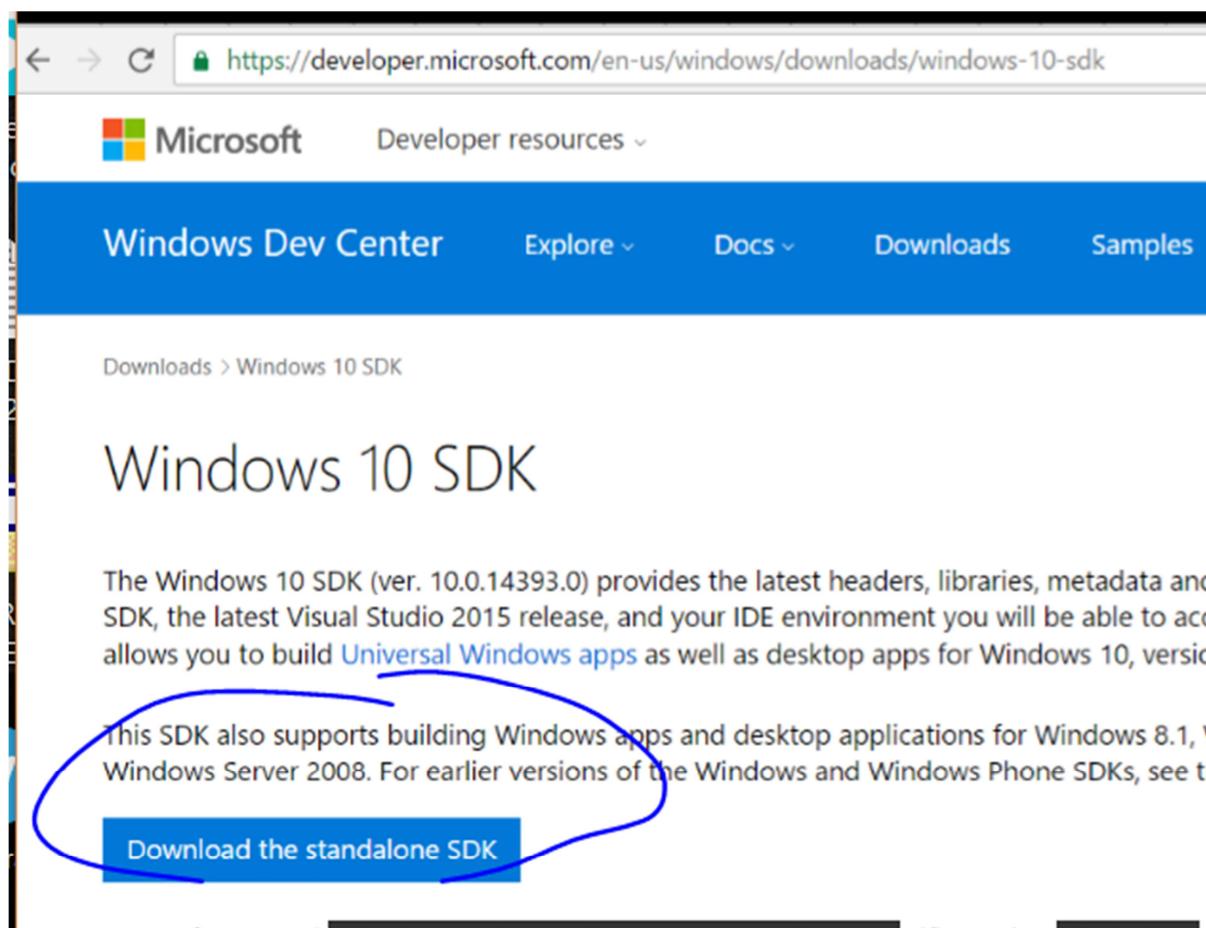
REVERSING WITH IDA PRO FROM SCRATCH

PART 32

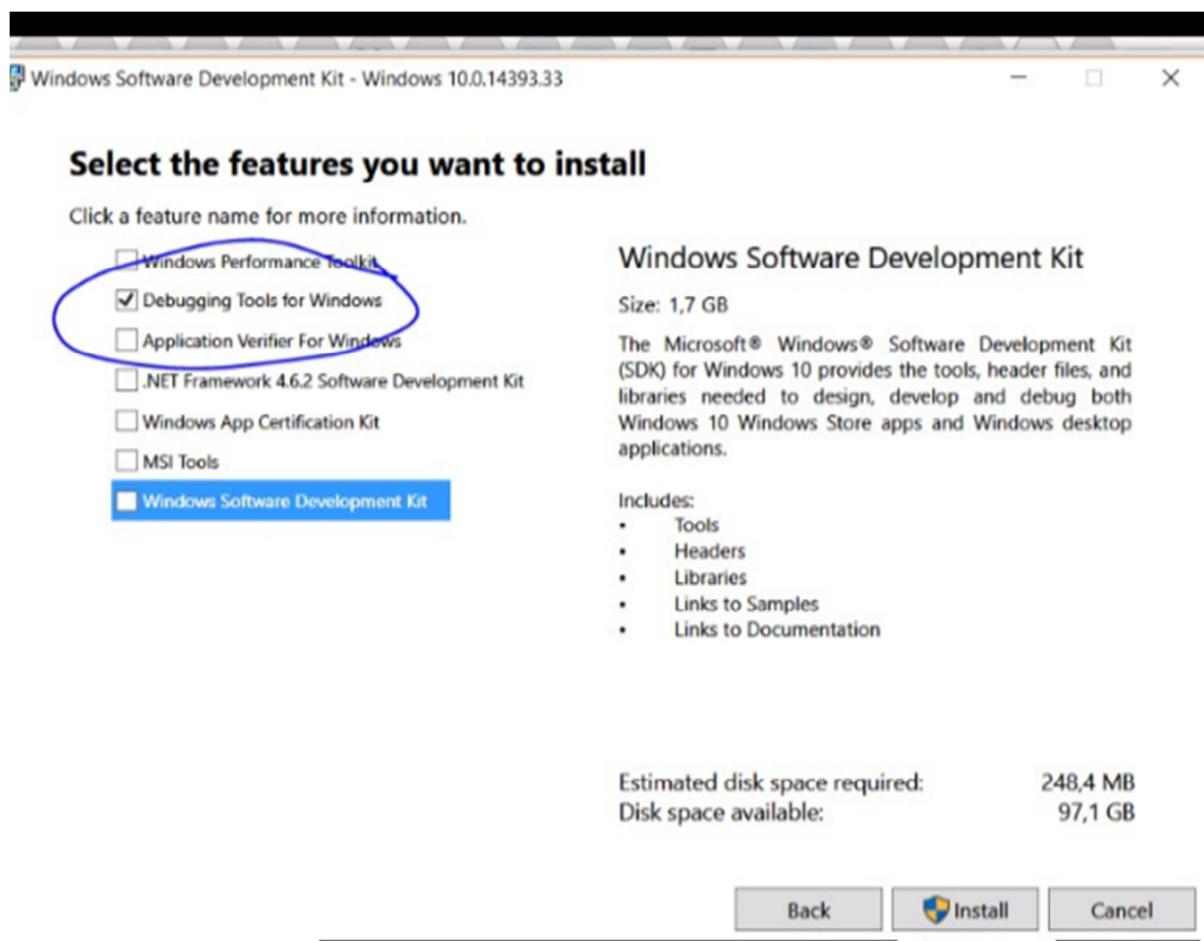
Preparing Windbg to work with IDA.

We need to install Windbg. It can be handled through IDA interface. It is a great debugger. Maybe, it lacks comfort because it has console-type commands and it is a little complicated to use it, but we can use the IDA great interface debugging with the Windbg engine. It is useful when we want to debug kernel or have good info about the heap status in heap overflow or after free bugs that we will see later.

There are many ways to install WinDbg. Unfortunately, they vary and you'll have to try the ones that work in your PC. As I am using Windows 10, I downloaded Windows 10 SDK from Microsoft website.

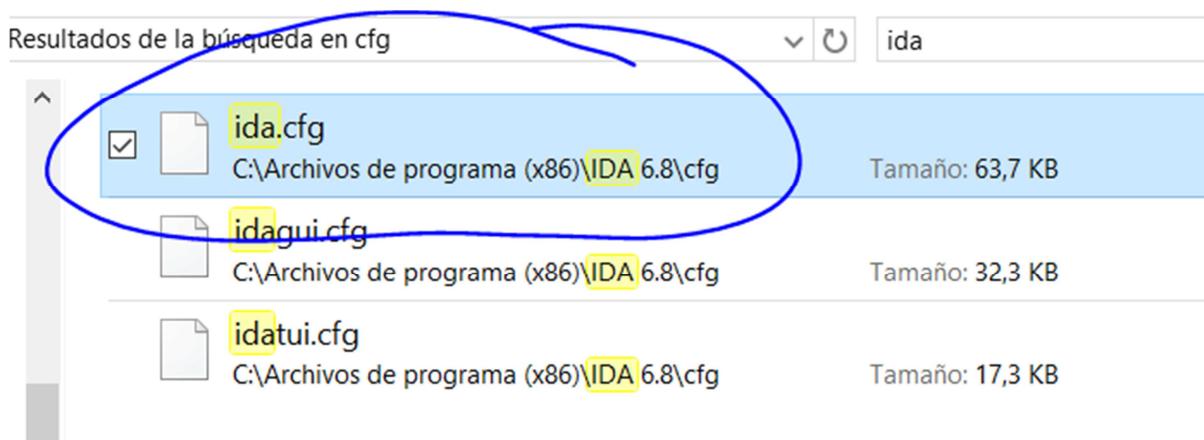


When the options appeared, I just selected DEBUGGING TOOLS FOR WINDOWS.

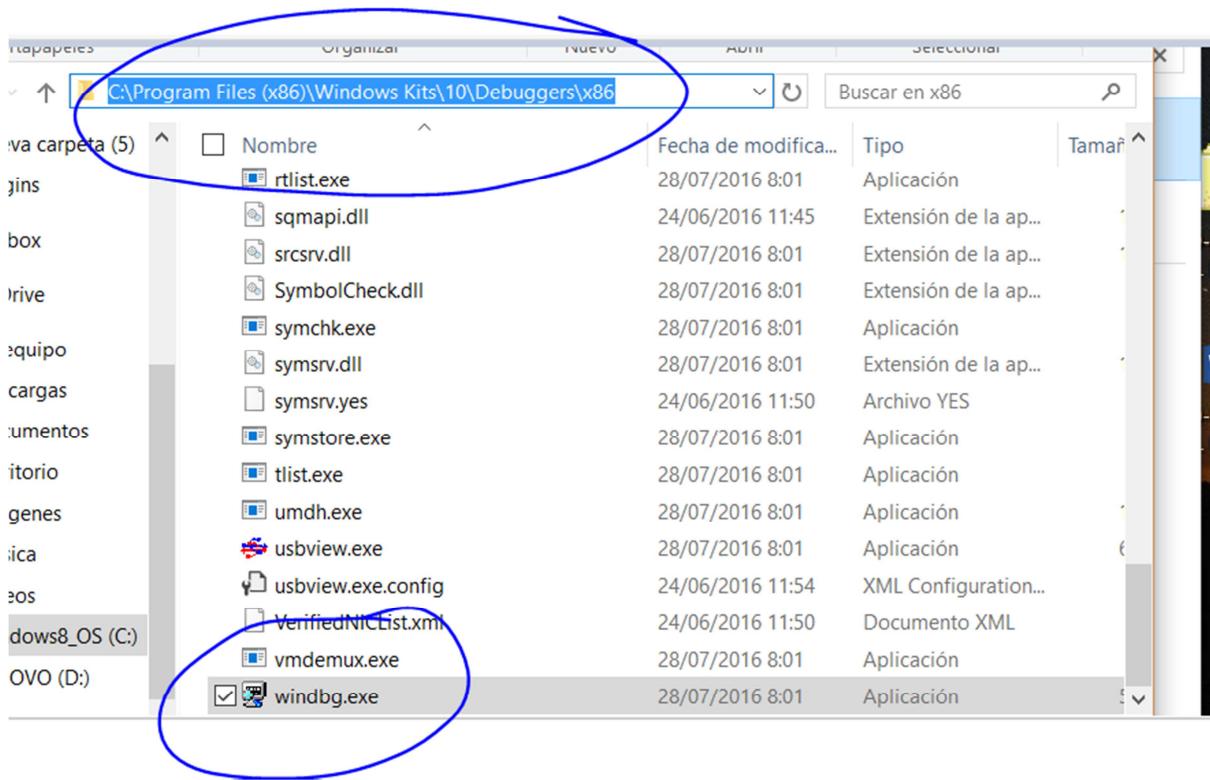


Unchecking the other options, it will install WinDbg. As each Windows has its own SDK, you could do the same in other Windows and install the correspondent one in your PC. At least, in this case, we know it goes well.

Then, I have to go to the cfg folder inside IDA installation and look for IDA.CFG



I edit that file and I add the path in DBGTOOLS where WinDbg x86 is installed. In my case, it is in:

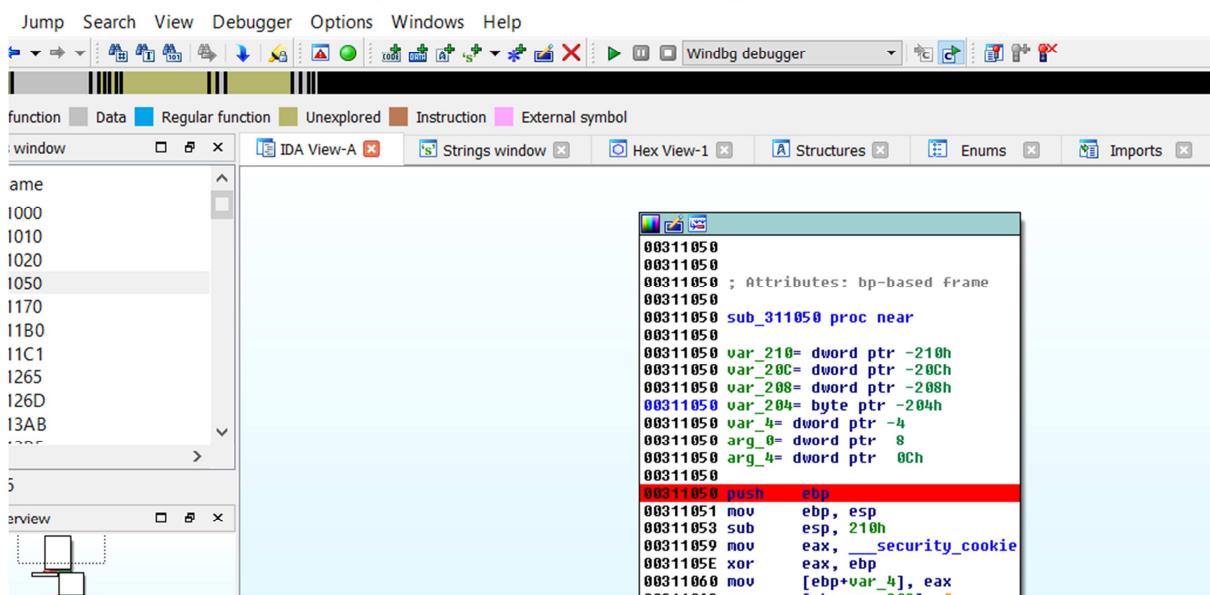


So, in the IDA.CFG file, I find DBGTOOLS.

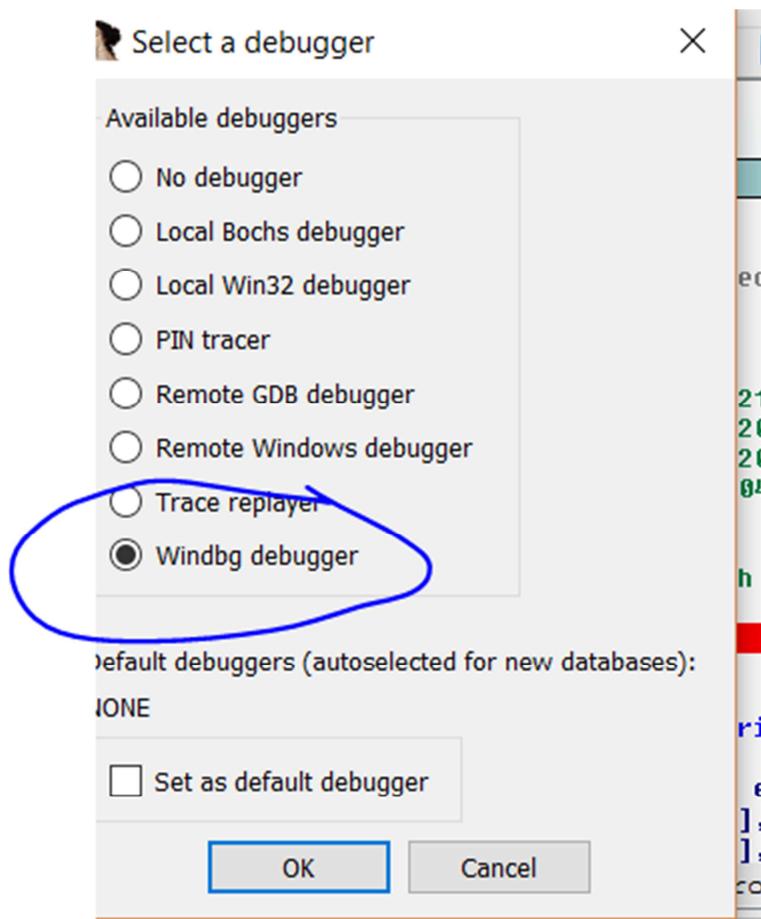
```
656 MAX_TRUSTED_IDB_COUNT = 1024
657
658 //-----
659 // Processor specific parameters
660 //-----
661 #ifdef __PC__                                // INTEL 80x86 PROCESSORS
662
663 //
664 // Location of Microsoft Debugging Engine Library (dbgeng.dll)
665 // This value is used by both the windmp (dump file loader) and the windbg
666 // debugger module. Please also refer to dbg_windbg.cfg
667 // (note: make sure there is a semicolon at the end)
668
669 //DBGTOOLS = "C:\\\\Program Files\\\\Debugging Tools for Windows (x86)\\\\";
670 DBGTOOLS = "C:\\\\Program Files (x86)\\\\Windows Kits\\\\10\\\\Debuggers\\\\x86\\\\";
671
672 USE_FPP           = YES          // Floating Point Processor
673                   // instructions are enabled
674
675 // IBM PC specific analyzer options
676
677 PC_ANALYZE_PUSH = YES          // Convert immediate operand of "push" to offset
678                   // In sequence
679                   //      push    seg
680                   //      push    num
681                   //      ..
682
```

And I add the exact path adding the double bar \\ instead of the single one to separate the folders. I left the original path commented above.

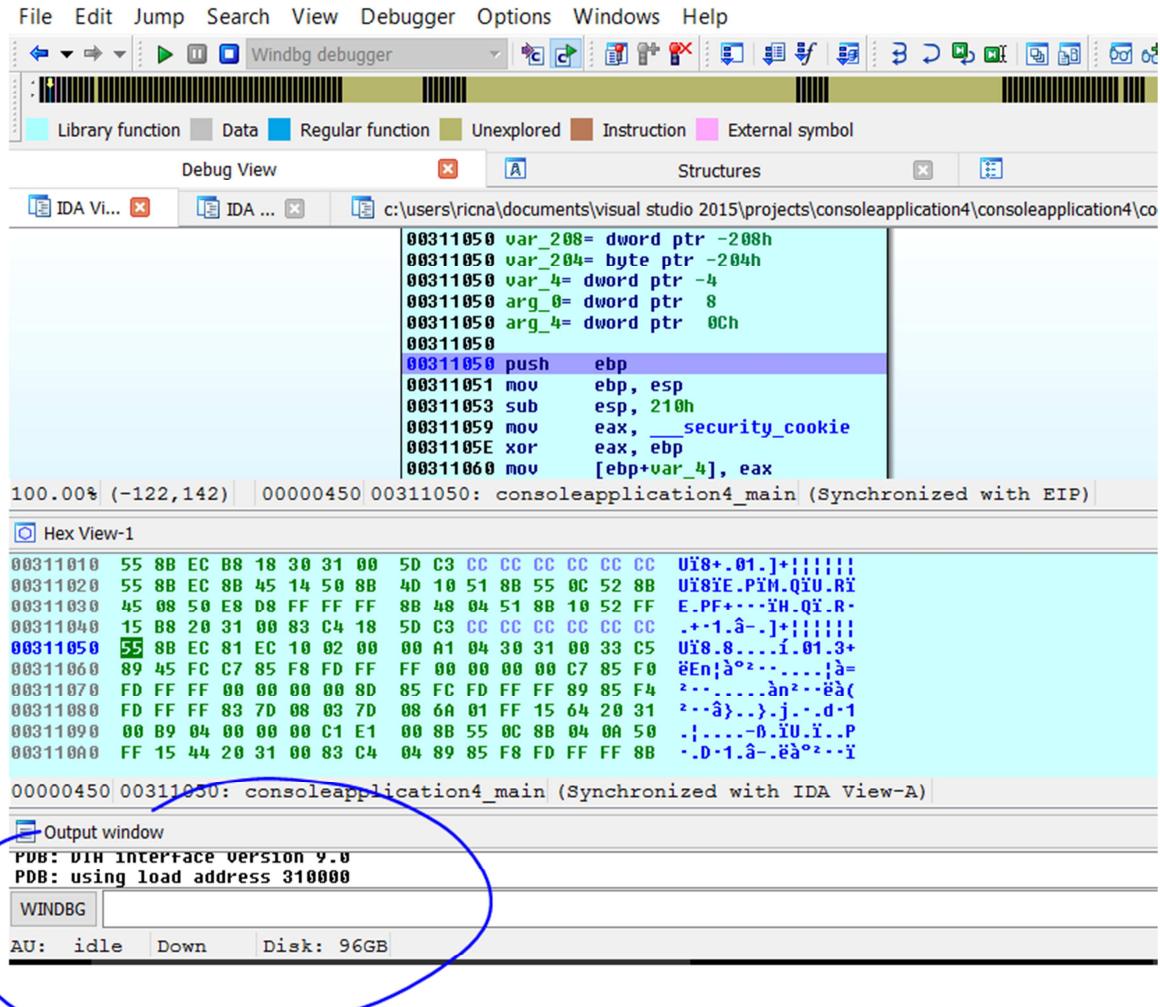
\Users\ricna\Documents\Visual Studio 2015\Projects\ConsoleApplication4\Release\ConsoleApplication4.idb (ConsoleApplication4.exe)



I open any executable and set a BreakPoint to stop and I change the debugger to WinDbg.

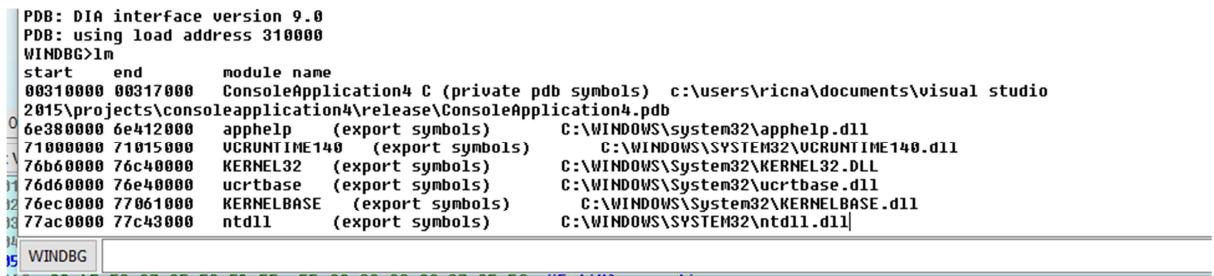


Then, we'll check if it was OK. We run the executable. If it says it doesn't find WinDbg, you should check the path or something failed in the installation.



There, it stopped as it were IDA Win32 local debugger, but we see that below where the Python bar is normally, it says WindDbg. Clicking on the word WindDbg, I can switch to the Python bar.

This means everything is fine until now. We can use IDA GUI and WinDbg commands. Let's try some of them.



!lm (List Loaded Modules)

The **!lm** command displays the specified loaded modules. The output includes the status and the path of the module.

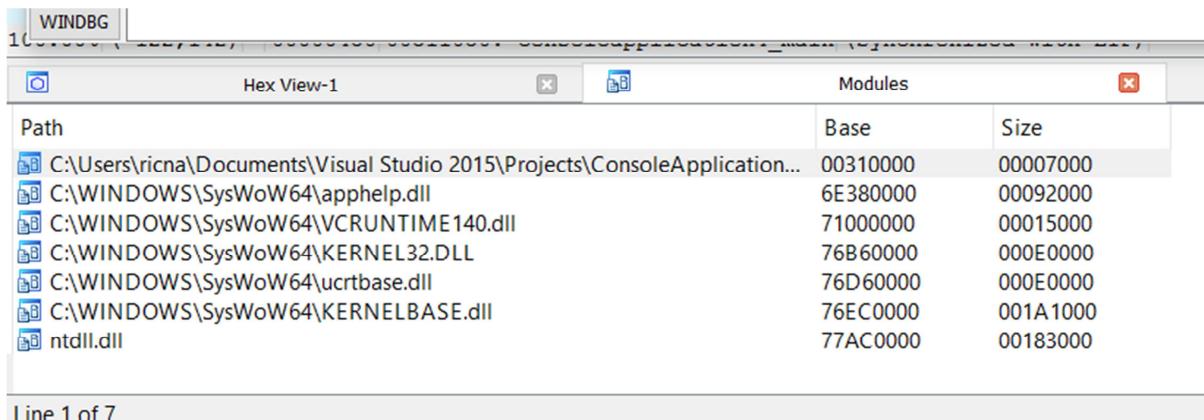
```
!lmOptions [a Address] [m Pattern | M Pattern]
```

Parameters

Options

Any combination of the following options:

It worked. I can see the module list in Windbg bar. Obviously, in IDA too.



We also have to set the symbols for WinDbg. If we type...

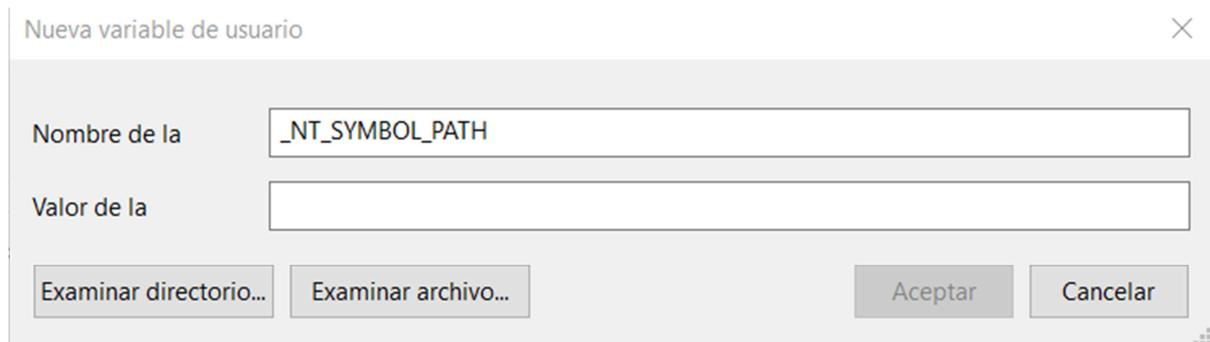
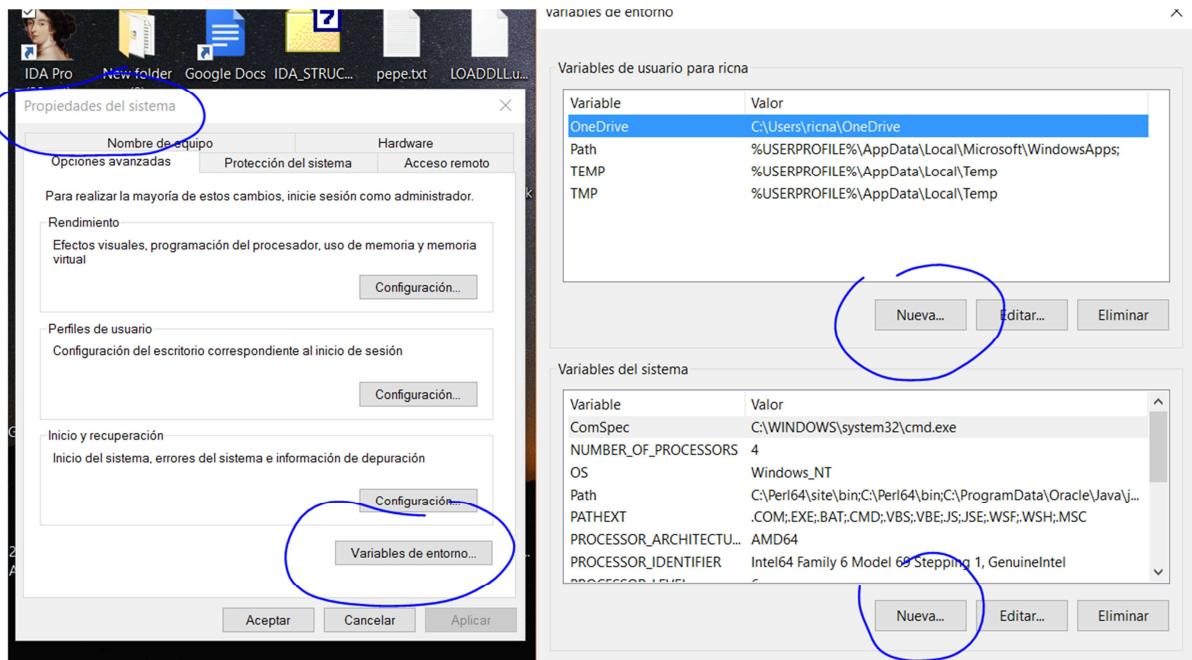
.reload

It shows error with symbols when they are not set where to save them.

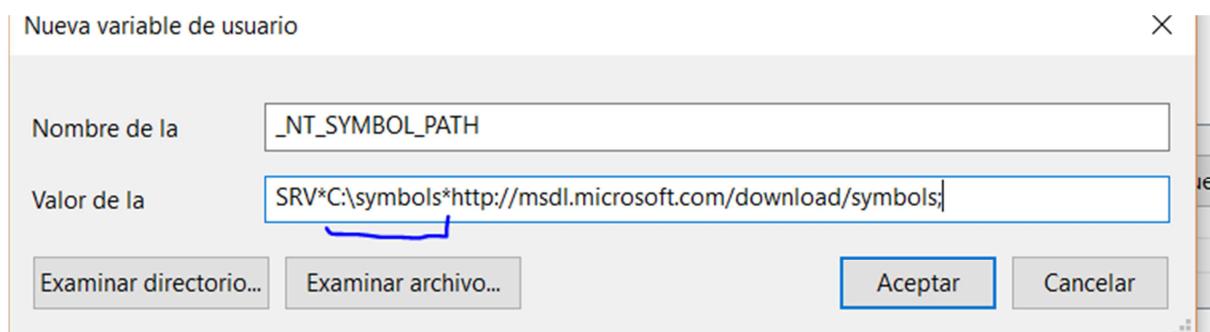
```
77ac0000-77c43000  ntdll      (export symbols)      C:\WINDOWS\SYSTEM32\ntdll.dll
WINDBG>.reload
Reloading current modules
...
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntdll.dll -
*** WARNING: Unable to verify checksum for ConsoleApplication4.exe
*****
***** Symbol Loading Error Summary *****
Module name          Error
ntdll               PDB not found : c:\users\ricna\documents\visual studio 2015\projects\consoleapplication4\release\symbols\dll\wntdll.pdb
You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.
WINDBG>
AU: idle  Down   Disk: 96GB
```

Create a folder in C for the symbols. IDA will have to run as administrator. If not, it won't write there.

In Windows ENVIRONMENT VARIABLES, we add `_NT_SYMBOL_PATH`.

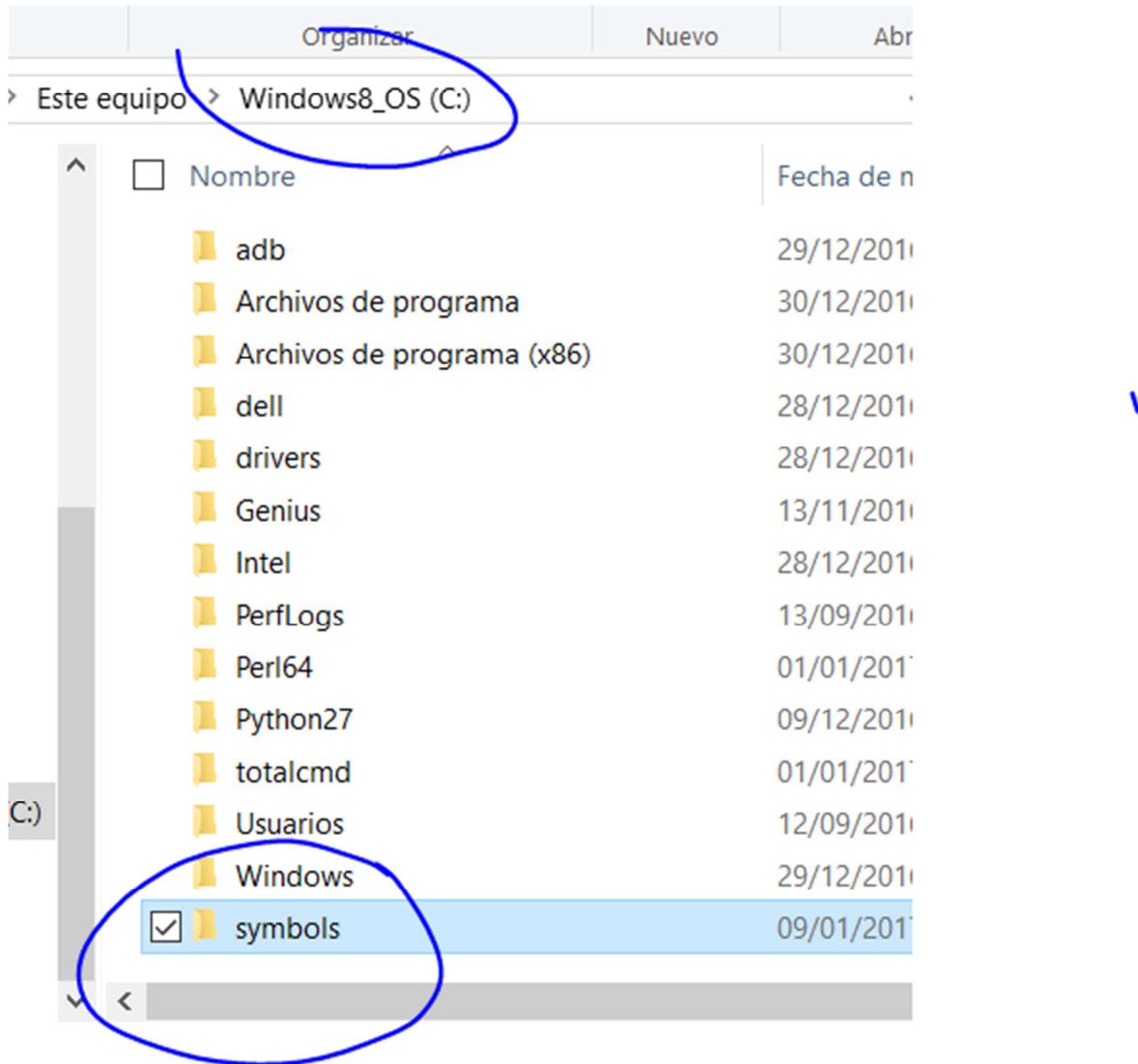


And in values, we type the following:



There, I will type the folder path where the symbols will be, in my case, it will be the **symbols** folder in C.

Alli pondre el path a la carpeta que cree donde se bajara los símbolos, en mi caso sera la carpeta symbols en C.



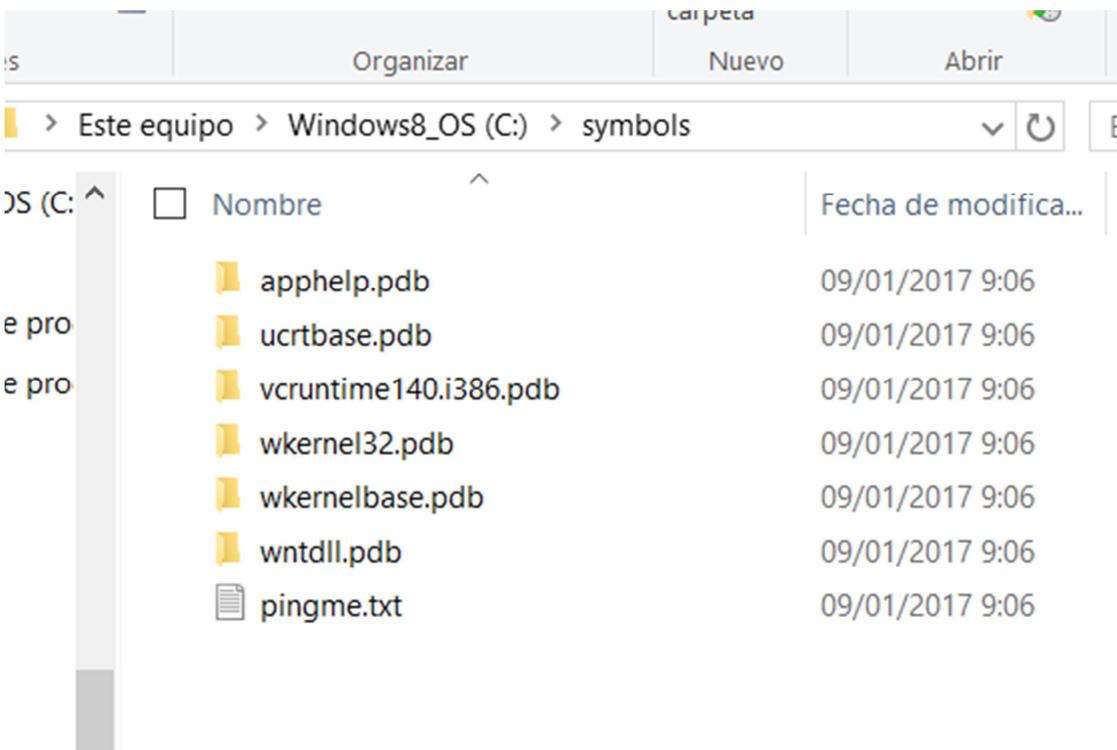
I restart my PC or the Windows explorer (explorer.exe killing and running it from the process bar)

When I run IDA again and run the debugger with WinDbg, I see it tries to download the symbols and when it finishes with lm, the path we set appears and the PDBs are inside the **symbols** folder.

```

77AF67C0: thread has started (tid=5688)
6EBE0000: loaded C:\WINDOWS\SysWOW64\VCRUNTIME140.dll
77AF67C0: thread has started (tid=21340)
PDBSRC: loading symbols for 'C:\Users\ricna\Documents\Visual Studio 2015\Projects\ConsoleApplication4\Release\ConsoleApplication4.exe'...
PDB: using DIA dll "C:\Program Files (x86)\Common Files\Microsoft Shared\VC\msdia90.dll"
PDB: DIA interface version 9.0
PDB: using load address 310000
WINDBG>lm
start end module name
00310000 00317000 ConsoleApplication4 C (private pdb symbols) c:\users\ricna\documents\visual studio
2015\projects\consoleapplication4\release\ConsoleApplication4.pdb
6e380000 6e412000 apphelp (pdb symbols) c:\symbols\apphelp.pdb\8D0F773372F460AAF38944E193F7E331\apphelp.pdb
6ebf5000 6ebf5000 VCRUNTIME140 (private pdb symbols) c:\symbols\vcruntime140.i386.pdb\87F8B303F87E49D2A009E36621F4D731\vcruntime140.i386.pdb
76b60000 76c40000 KERNEL32 (pdb symbols) c:\symbols\wkernel32.pdb\E8898005F9941FB8E728782D0508C581\wkernel32.pdb
76d60000 76e40000 ucrtbase (pdb symbols) c:\symbols\ucrtbase.pdb\14E1C8EC74D4CE09B29C0D25BF7EC0D1\ucrtbase.pdb
76ec0000 77061000 KERNELBASE (pdb symbols) c:\symbols\wkernelbase.pdb\4DAD89FB11074179A0B1D6168D930D1\wkernelbase.pdb
77ac0000 77c43000 ntdll (pdb symbols) c:\symbols\wntdll.pdb\9D5E8B427B3449C00A160009A90706251\wntdll.pdb

```



It is ready to work.

```

76ec0000 77061000 KERNELBASE (pdb symbols) c:\symbols\wkernelbase.pdb\4DAD89FB1
77ac0000 77c43000 ntdll (pdb symbols) c:\symbols\wntdll.pdb\9D5E8B427B3449C01
WINDBG>x kernel32!
76b82840 KERNEL32!CreateActCtxA (void)
76baF9fd KERNEL32!InitializeGeoIDExclusionList (void)
76bb6d64 KERNEL32!RtlStringCopyWideCharArrayWorker (void)
76b885d0 KERNEL32!TermServiceDeleteKey (void)
76b799aa KERNEL32!RtlStringCchCopyW (void)
76b7698a KERNEL32!FSPErrorMessages::Config::OpenPerUserKey (void)
76b799df KERNEL32!AslPathIsTemporaryInternetFile (void)
76ba89c7 KERNEL32!Internal_InvokeSwitchCallbacksOnINIT (void)
76b7d300 KERNEL32!RegisterWaitForSingleObject (void)
76b801f3 KERNEL32!BaseFindActCtxSection_CheckAndConvertParameters (void)
76b793dc KERNEL32!WerpCurrentPeb (void)
76b75620 KERNEL32!GetLongPathNameW (void)

```

We can see the kernel32 function list. We can also use wildcards.

Output window

```
WINDBG x kernel32!He*
76b88540 KERNEL32!HeapDestroyStub (<no parameter info>)
76b73fc0 KERNEL32!HeapFreeStub (<no parameter info>)
76b82a30 KERNEL32!HeapUnlockStub (<no parameter info>)
76b97760 KERNEL32!HeapQueryInformationStub (<no parameter info>)
76bbcfc0 KERNEL32!Heap32ListFirst (<no parameter info>)
76b80310 KERNEL32!HeapValidateStub (<no parameter info>)
76be84b0 KERNEL32!HebrewTable = <no type information>
76bac12c KERNEL32!HebrewToAbsolute (<no parameter info>)
76bbd100 KERNEL32!Heap32Next (<no parameter info>)
76b7d490 KERNEL32!HeapCreateStub (<no parameter info>)
76b977a0 KERNEL32!HeapWalkStub (<no parameter info>)
76bac169 KERNEL32!HebrewToGregorian (<no parameter info>)
76b97740 KERNEL32!HeapCompactStub (<no parameter info>)
76b7dc00 KERNEL32!HeapSetInformationStub (<no parameter info>)
76bbd060 KERNEL32!Heap32ListNext (<no parameter info>)
76b97780 KERNEL32!HeapSummaryStub (<no parameter info>)
```

The ones that start with He.

I can also right click - LOAD SYMBOLS on any modules in IDA

Path	Base	Size
C:\Users\ricna\Documents\Visual Studio 2015\Projects\ConsoleApplication...	00310000	00007000
C:\WINDOWS\SyWoW64\apphelp.dll	6E380000	00092000
C:\WINDOWS\SyWoW64\VCRUNTIME140.dll	6EBE0000	00015000
C:\WINDOWS\SyWoW64\KERNEL32.DLL	76B60000	000E0000
C:\WINDOWS\SyWoW64\lucrbase.dll	76D60000	000E0000
C:\WINDOWS\SyWoW64\KERNELBASE.dll	76EC0000	001A1000
ntdll.dll	77AC0000	00183000

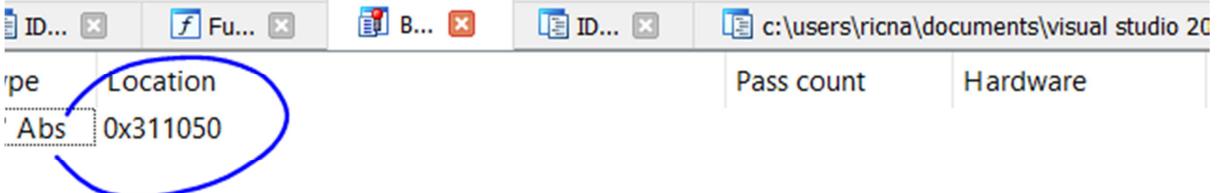
I will also have them in IDA interface.

I can set breakpoints in IDA interface as we do it normally with F2 and through WinDbg bar that will be handled by it.

```
Caching functions window ... ok
WINDBG>x kernel32!HeapFr*
76b73fc0      KERNEL32!HeapFreeStub (<no parameter info>)
WINDBG>bp kernel32!HeapFreeStub
```

I type **BL** to list WinDbg breakpoints.

```
WINDBG>x kernel32!HeapFr*
76b73fc0      KERNEL32!HeapFreeStub (<no parameter info>)
WINDBG>bp kernel32!HeapFreeStub
WINDBG>bl
0 e 00311050    0001 (0001)  0:***** ConsoleApplication4!main
1 d 00000000    0001 (0001)  0:*****
2 e 76b73fc0    0001 (0001)  0:***** KERNEL32!HeapFreeStub
```



```
Output window

Type library 'vc6win' loaded. Applying types...
Types applied to 0 names.
PDB: using load address 76B60000
PDB: loaded 0 types
PDB: total 4170 symbols loaded for C:\WINDOWS\SysWoW64\KERNEL32.DLL
Caching 'Functions window'... ok
Caching 'Functions window'... ok
WINDBG>bp kernel32!HeapFree
Couldn't resolve error at 'kernel32!HeapFree'
Caching 'Functions window'... ok
WINDBG>x kernel32!HeapFr*
76b73fc0      KERNEL32!HeapFreeStub (<no parameter info>)
WINDBG>bp kernel32!HeapFreeStub
WINDBG>bl
0 e 00311050    0001 (0001)  0:***** ConsoleApplication4!main
1 d 00000000    0001 (0001)  0:*****
2 e 76b73fc0    0001 (0001)  0:***** KERNEL32!HeapFreeStub
```

In WinDbg, all BPs are listed while in IDA, it only shows IDA BPs, but it will always stop at all them.

It is important that you prepare the installation to make it work well. Sometimes, there are problems with that. So, I will continue in the next part.

Ricardo Narvaja

Translated by: @IvinsonCLS