

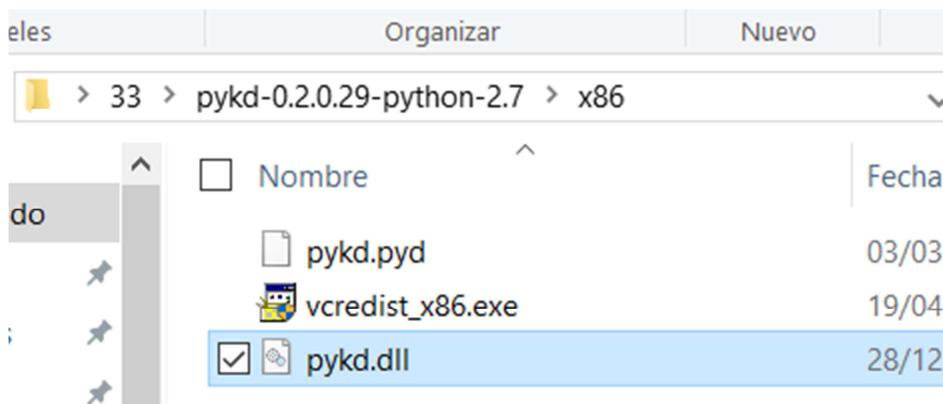
REVERSING WITH IDA PRO FROM SCRATCH

PART 33

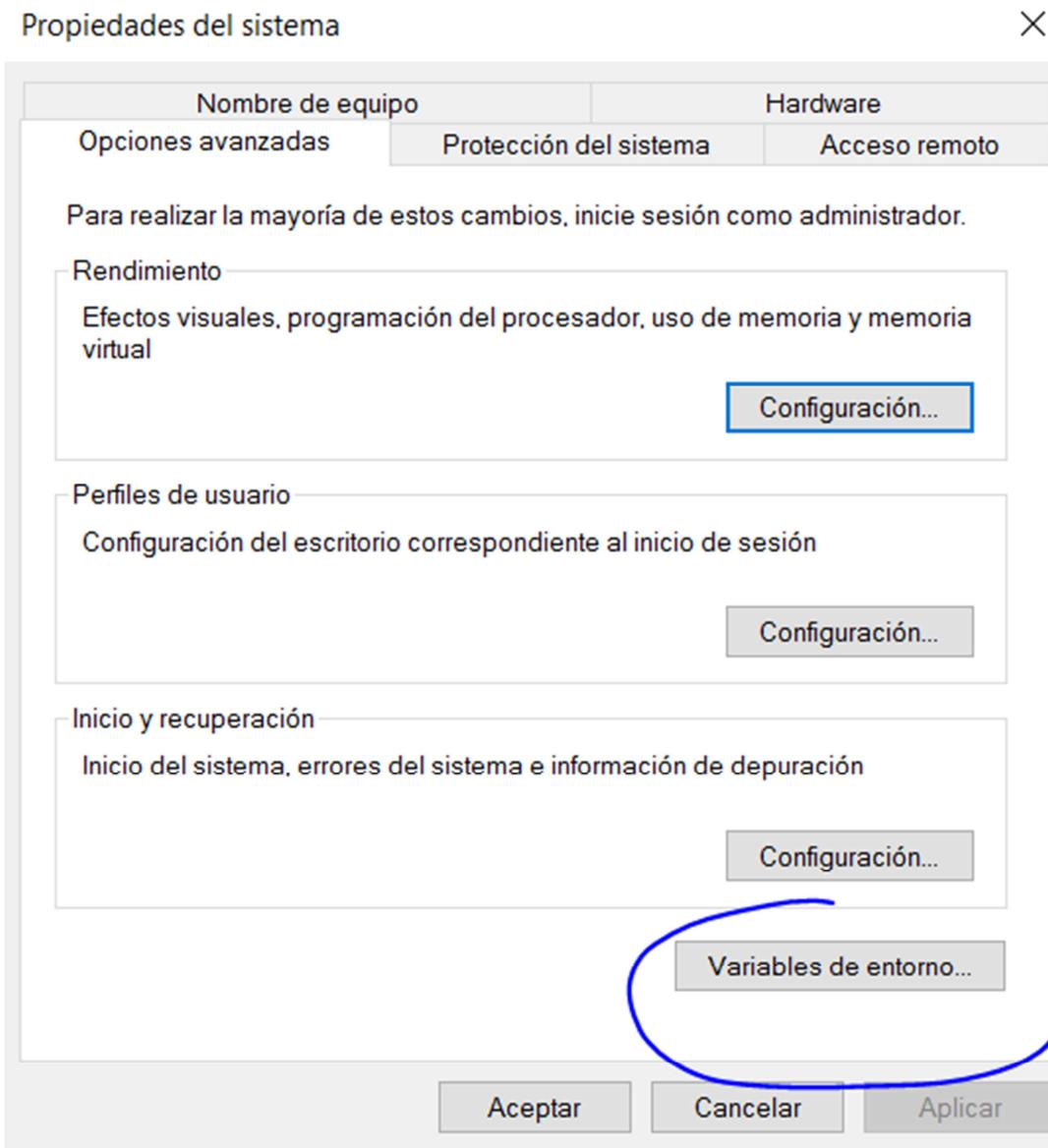
Let's install some plugins for WinDbg that will help us more ahead when we work.

Unfortunately, these plugins only run in WinDbg, but if you run them in WinDbg inside IDA, it crashes because it has conflicts with the Python included in IDA, but we'll use them separately in WinDbg when we need them.

Copy the files that I attached in the **winext** folder located in WinDbg installation folder and install the **vcredist_x86.exe** runtime.

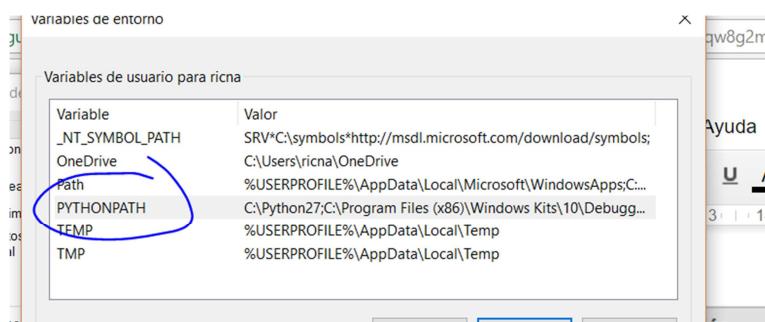


Then, set the system environment variables.

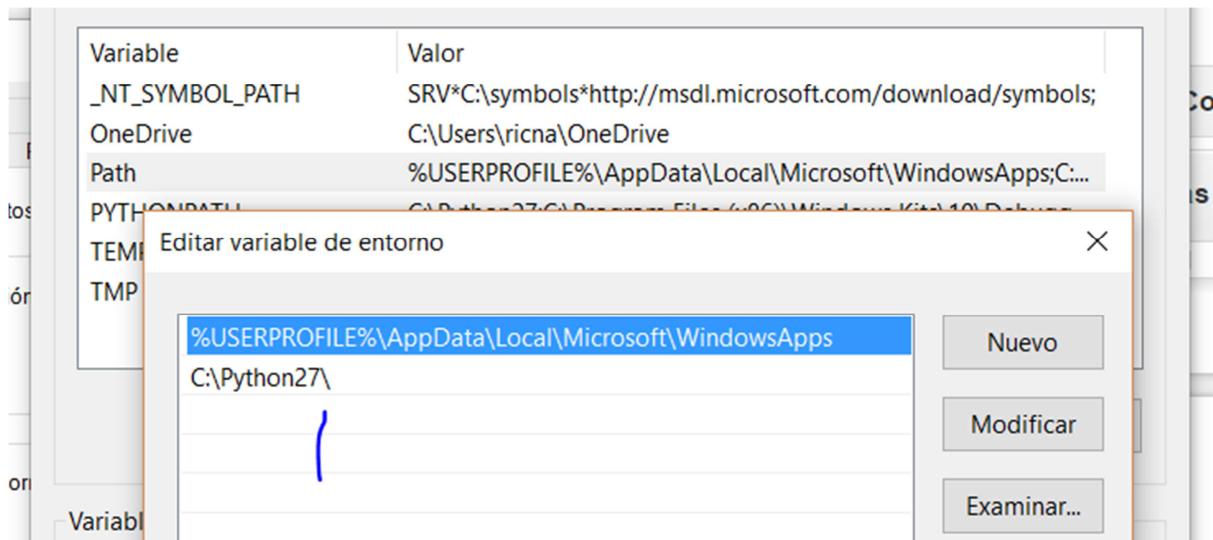


Add the PYTHONPATH variable to which I added the Python path. Then, a semicolon ';' and the winext path. In my case, it is like this:

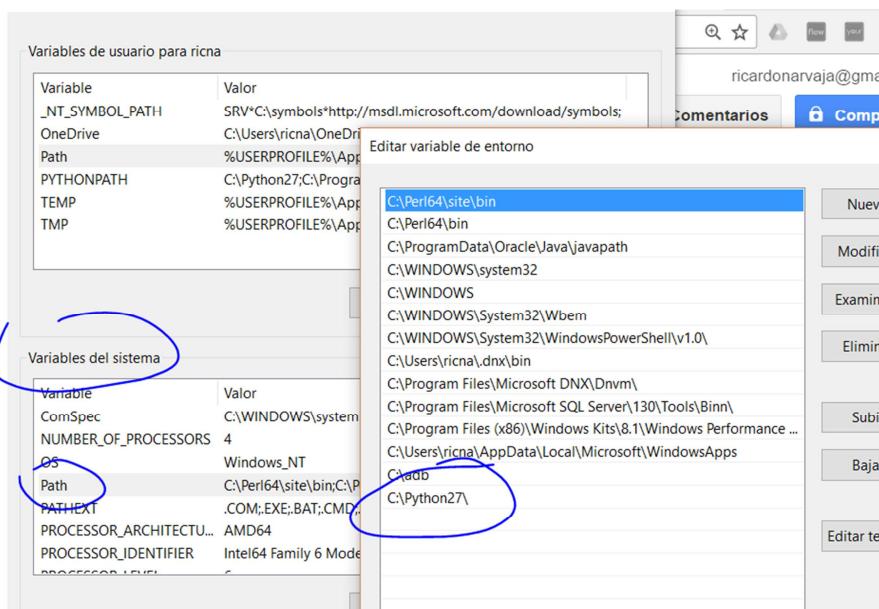
C:\Python27;C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\winext



And I add a **semicolon** to the end of the **Path** variable that already existed and then **C:\Python27**



I make sure it has the final bar. I also add it to **System Variables** in **Path**.



After restarting the PC or killing the Windows explorer, we could type **python** in any console and it should accept it.

```
U Símbolo del sistema - python
O Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

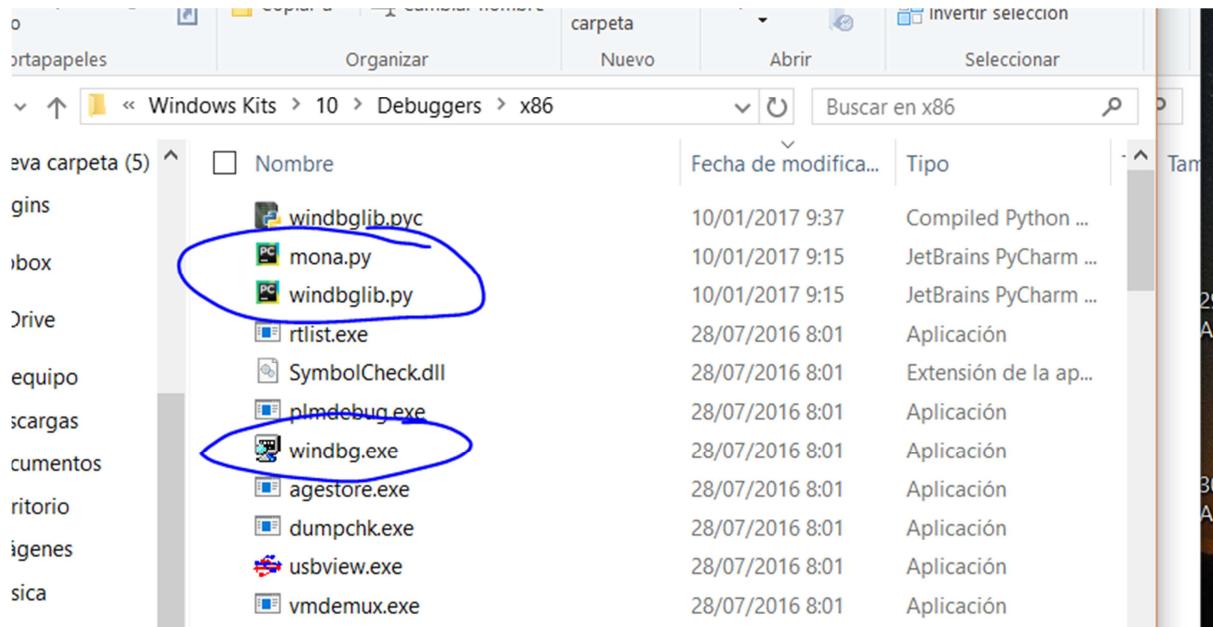
C:\Users\ricna>python
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The last thing is to download the latest versions of:

windbglib.py <https://github.com/corelan/windbglib/raw/master/windbglib.py>

mona.py from <https://github.com/corelan/mona/raw/master/mona.py>

And copy them in the same folder where windbg.exe is.



With that, it should work. Let's run WinDbg outside IDA. If it fails, you need some runtime. If not, it should run correctly.

Open any executable with CTRL + E.

When it stops, type:

!load pykd.pyd

Nothing would happen, but it must not give any error.

```
0:000> !load pykd.pyd
0:000> !py
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (...)
Type "help", "copyright", "credits" or "license" for more information
(InteractiveConsole)
>>>
<
Input>
```


Module info :											
Base	Top	Size	Rebase	SafeSEH	ASLR	NXCompat	OS Dll	Version	Modulename & Path		
0x74740000	0x748e1000	0x001a1000	False	True	True	False	True	10.0.14393.479	[KERNELBASE.dll] (C:\WINDOWS\Sys		
0x77c00000	0x77d83000	0x00183000	False	True	True	False	True	10.0.14393.479	[ntdll.dll] (ntdll.dll)		
0x751b0000	0x75290000	0x000e0000	False	True	True	False	True	10.0.14393.0	[KERNEL32.DLL] (C:\WINDOWS\SysWoW64\m		
0x75c70000	0x75d2e000	0x000be000	False	True	True	False	True	7.0.14393.0	[msvcr7.dll] (C:\WINDOWS\SysWoW64\m		
0x00400000	0x00406000	0x00006000	False	False	False	False	False	-1.0-	[IDA1.exe] (image00400000)		

We see the running modules. We'll study them later. Now, we are preparing the workplace to have everything ready for the war.

!py mona rop

That would last so much. It will try to see if there is some module where it could generate a ROP (Later, we'll see what it is) and try to create it.

```
Want more info about a given command ? Run !mona help
0:000> !py mona rop
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py rop
----- Mona command started on 2017-01-10 11:48:28 (v2.0, rev 567) -----
[+] Processing arguments and criteria
  - Pointer access level : X
[+] Generating module info table, hang on...
  - Processing modules
  - Done. Let's rock 'n roll.
[+] Preparing output file '_rop_progress_IDA1.exe_8316.log'
  - (Re)setting logfile _rop_progress_IDA1.exe_8316.log
[+] Progress will be written to _rop_progress_IDA1.exe_8316.log
[+] Maximum offset : 40
[+] (Minimum/optional maximum) stackpivot distance : 8
[+] Max nr of instructions : 6
[+] Split output into module rop files ? False
[+] Enumerating 22 endings in 1 module(s)...
  - Querying module IDA1.exe
  - Search complete :
    Ending : RETN, Nr found : 14
    Ending : RETN 0x04, Nr found : 2
  - Filtering and mutating 16 gadgets
    - Progress update : 16 / 16 items processed (Tue 2017/01/10 11:48:30 AM) - (100%)
[+] Creating suggestions list
[+] Processing suggestions
[+] Launching ROP generator
[+] Attempting to produce rop chain for VirtualProtect
```

Sometimes, it could create it, but at least, it is working.

```

def create_rop_chain()
    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
        0x00000000, # [-] Unable to find gadgets to pickup the desired API pointer into esi
        0x75231150, # ptr to &VirtualAlloc() (skipped module criteria, check if pointer is reliable)
        0x0040172f, # POP EBP # RETN [IDA1.exe]
        0x00000000, # & [Unable to find ptr to 'JMP ESP']
        0x004013c7, # POP EBX # POP EBP # RETN [IDA1.exe]
        0x00000001, # Ox00000001-> ebx
        0x41414141, # Filler (compensate)
        0x00000000, # [-] Unable to find gadget to put 00001000 into edx
        0x00000000, # [-] Unable to find gadget to put 00000040 into ecx
        0x0040152b, # POP EDI # POP EBP # RETN [IDA1.exe]
        0x00401346, # RETN (ROP NOP) [IDA1.exe]
        0x41414141, # Filler (compensate)
        0x00000000, # [-] Unable to find gadget to put 90909090 into eax
        0x00000000, # [-] Unable to find pushad gadget
    ].flatten.pack("V*")
    return rop_gadgets
end

```

It returned what it could and as I didn't run WinDbg as administrator, it couldn't write the file with the output, but it prints it anyways.

```

"%u172f%u0040" + // 0x0040172f : [# POP EBP # RETN [IDA1.exe] ]
"%u0000%u0000" + // 0x00000000 : [# &[Unable to find ptr to 'JMP ESP']]
"%u13c7%u0040" + // 0x004013c7 : [# POP EBX # POP EBP # RETN [IDA1.exe]]
"%u0001%u0000" + // 0x00000001 : # Ox00000001-> ebx
"%u4141%u4141" + // 0x41414141 : # Filler (compensate)
"%u0000%u0000" + // 0x00000000 : # [-] Unable to find gadget to put 00001000 into edx
"%u0000%u0000" + // 0x00000000 : # [-] Unable to find gadget to put 00000040 into ecx
"%u152b%u0040" + // 0x0040152b : # POP EDI # POP EBP # RETN [IDA1.exe]
"%u1346%u0040" + // 0x00401346 : # RETN (ROP NOP) [IDA1.exe]
"%u4141%u4141" + // 0x41414141 : # Filler (compensate)
"%u0000%u0000" + // 0x00000000 : # [-] Unable to find gadget to put 90909090 into eax
"%u0000%u0000" + // 0x00000000 : # [-] Unable to find pushad gadget
""); // :

```

```

-----  

ROP generator finished  

+] Writing stackpivots to file stackpivot.txt  

Wrote 0 pivots to file  

+] Writing suggestions to file rop_suggestions.txt  

*****  

raceback (most recent call last):  

File "C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py", line 18183, in main  

    commands[command].parseProc(opts)  

File "C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py", line 11341, in procROP  

    findROPGADGETS(modulecriteria,criteria,endings,maxoffset,depth,split,thedistance,fast_mode)  

File "C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py", line 6345, in findROPGADGETS  

    with open(thislog, "a") as fh:  

OSError: [Errno 13] Permission denied: 'rop_suggestions.txt'  

*****

```

We can check if there is a mona update.

```

...>>> !Load pykd.pyd
>>> !py mona update
told on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py update
[+] Version compare :
    Current Version : '2.0', Current Revision : 567
    Latest Version : '2.0', Latest Revision : 567
[+] You are running the latest version
[+] Locating windbglib path
[+] Checking if C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\windbglib.py needs an update...
[+] Version compare :
    Current Version : '1.0', Current Revision : 141
    Latest Version : '1.0', Latest Revision : 141
[+] You are running the latest version
[+] This mona.py action took 0:00:07.741000

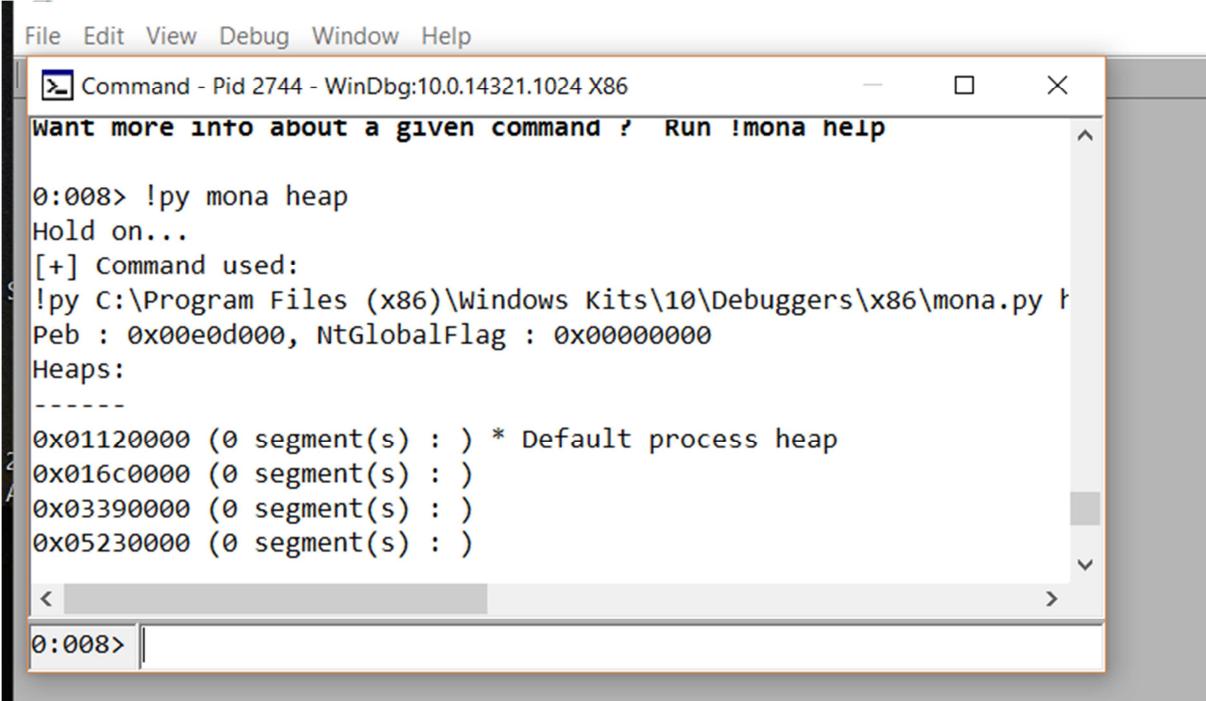
```

```
+] This mona.py action took 0:00:07.741000
:000> !py mona heap
old on...
+] Command used:
py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py heap
eb : 0x0020f000, NtGlobalFlag : 0x00000070
eaps:
-----
x00720000 (0 segment(s) : ) * Default process heap

lease specify a valid searchtype -t
alid values are :
    lal
    lfh
    all
    segments
    chunks
    layout
    fea
    bea

+] This mona.py action took 0:00:00.008000
```

Attaching a running process, in this case, notepad++.



The screenshot shows the WinDbg command window with the title "Command - Pid 2744 - WinDbg:10.0.14321.1024 X86". The window displays the results of the !py mona heap command. The output includes the command used, memory addresses, and heap status. The command window has a scroll bar and a status bar at the bottom showing "0:008>".

```
File Edit View Debug Window Help
Command - Pid 2744 - WinDbg:10.0.14321.1024 X86
Want more info about a given command ? Run !mona help

0:008> !py mona heap
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py h
Peb : 0x00e0d000, NtGlobalFlag : 0x00000000
Heaps:
-----
0x01120000 (0 segment(s) : ) * Default process heap
0x016c0000 (0 segment(s) : )
0x03390000 (0 segment(s) : )
0x05230000 (0 segment(s) : )

< >
0:008>
```

I can see the heap status.

We'll get in deep about this soon. Anyways, WinDbg also has heap commands that we can use inside IDA without using **mona**.

```
Command
0:008> !heap -h
HEAPEXT: Unable to get address of ntdll!RtlpHeapInvalidBadAddress.
Index Address Name      Debugging options enabled
1: 01120000
    Segment at 01120000 to 0121f000 (000ff000 bytes committed)
    Segment at 03640000 to 0373f000 (000ff000 bytes committed)
    Segment at 03740000 to 0393f000 (001ff000 bytes committed)
    Segment at 05240000 to 0563f000 (00253000 bytes committed)
2: 016c0000
    Segment at 016c0000 to 016cf000 (0000f000 bytes committed)
3: 03390000
    Segment at 03390000 to 0339f000 (0000f000 bytes committed)
4: 05230000
    Segment at 05230000 to 0523f000 (00003000 bytes committed)
```

```
<-----||----->
```

So, we're OK. At least, if we need it, we have many options inside or outside IDA and we have all installed to go ahead.

More commands to have fun (it has thousands)

!py mona assemble -s "jmp esp"

```
Command
0x000250000 (0 segments) . .

0x03640000 is not a valid heap base address

[+] This mona.py action took 0:00:00.017000
0:008> !py mona assemble -s "jmp esp"
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py assemble -s jmp esp
Opcode results :
-----
jmp esp = \xff\xe4
Full opcode : \xff\xe4
[+] This mona.py action took 0:00:00.002000
```

```
<-----||----->
0:008> !py mona assemble -s "jmp esp"
```

!py mona getiat

It really has many useful commands to see the imported functions, although it lasts too much.

```

Command
0x10003044 | At 0x10003044 in nppexport (base + 0x000003044) : 0x751ceav (ptr to KERNEL32.getModuleHandleA)
0x10003048 | At 0x10003048 in nppexport (base + 0x000003048) : 0x77c45b00 (ptr to ntdll.rtlanstringtou
0x1000308c | At 0x1000308c in nppexport (base + 0x00000308c) : 0x75885440 (ptr to USER32.screenToClient)
0x1000304c | At 0x1000304c in nppexport (base + 0x00000304c) : 0x751c9f90 (ptr to KERNEL32.getProcessId)
0x10003050 | At 0x10003050 in nppexport (base + 0x000003050) : 0x751c3fc0 (ptr to KERNEL32.heapQueryInfor
0x10003054 | At 0x10003054 in nppexport (base + 0x000003054) : 0x751c50cd (ptr to KERNEL32.heap32First)
0x10003058 | At 0x10003058 in nppexport (base + 0x000003058) : 0x751d9660 (ptr to KERNEL32.closePrivateN
0x10003060 | At 0x10003060 in nppexport (base + 0x000003060) : 0x761aa160 (ptr to SHLWAPI.0x761aa160) -
0x10003090 | At 0x10003090 in nppexport (base + 0x000003090) : 0x7588a3e0 (ptr to USER32.registerMessage)
0x10003064 | At 0x10003064 in nppexport (base + 0x000003064) : 0x761acead (ptr to SHLWAPI.0x761acead) -
0x10003068 | At 0x10003068 in nppexport (base + 0x000003068) : 0x761aa810 (ptr to SHLWAPI.0x761aa810) -
0x10003070 | At 0x10003070 in nppexport (base + 0x000003070) : 0x758982c0 (ptr to USER32.controlMagnific
0x10003074 | At 0x10003074 in nppexport (base + 0x000003074) : 0x75898e10 (ptr to USER32.endDeferWindowP
0x10003078 | At 0x10003078 in nppexport (base + 0x000003078) : 0x75897050 (ptr to USER32.paintDesktop) -
0x1000307c | At 0x1000307c in nppexport (base + 0x00000307c) : 0x759022a0 (ptr to USER32.0x759022a0) -

```

Execute it as administrator to get much info in a file. We can also get info of an address.

!py mona info -a address

```

+] Information about address 0x77ca748c
  {PAGE_EXECUTE_READ}
Address is part of page 0x77c01000 - 0x77d0d000
Section : .text
Address is part of a module:
[ntdll.dll] ASLR: True, Rebase: False, SafeSEH: True, OS: True, v10.0.1439:
Offset from module base: 0xa748c

+] Disassembly:
  Instruction at 77ca748c : INT 3

Output of !address 0x77ca748c:

Usage:          Image
Base Address:   77c01000
End Address:    77d0d000
Region Size:   0010c000 ( 1.047 MB)
State:          00001000      MEM_COMMIT
Protect:        00000020      PAGE_EXECUTE_READ
Type:           01000000      MEM_IMAGE
Allocation Base: 77c00000
Allocation Protect: 00000080      PAGE_EXECUTE_WRITECOPY
Image Path:    ntdll.dll
Module Name:   ntdll
Loaded Image Name: C:\WINDOWS\SYSTEM32\ntdll.dll
Mapped Image Name:
More info:     !mv m ntdll
More info:     !lmi ntdll
More info:     In 0x77ca748c
< 0:000>

```

We had fun for a while and we installed the necessary tools to go ahead.

Ricardo Narvaja

Translated by: @IvinsonCLS