REVERSING WITH IDA PRO FROM SCRATCH

PART 17

We reached the part 17 and you should have tried to solve the exercise at least.

http://ricardonarvaja.info/WEB/INTRODUCCION%20AL%20REVERSING%20CON%20IDA%20PRO%20DESDE%20CERO/EJERCICIOS/

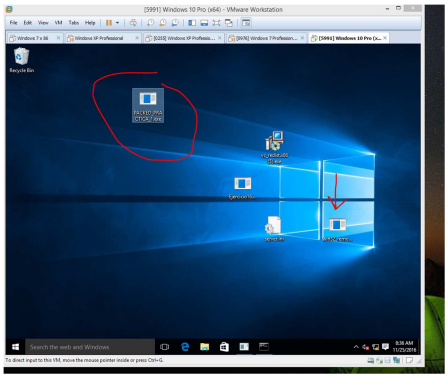
It is very easy. We have to unpack it as we did in previous parts and then, reverse it to find its solution and make a keygen in Python if possible.

In this case, we'll change a little. I will unpack it in a remote way to a virtual machine I have with Windows 10 in VMWare Workstation.

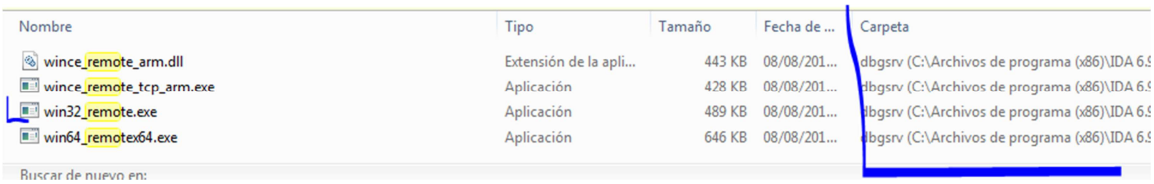
The main machine to debug remotely is where I have IDA running with Windows 7, but it could be any Windows, Linux, IOS, etc. where we have IDA installed.

The problems we could have unpacking remotely are the same we could have debugging in Windows 10 directly because it is there where the packed file will run. Its name is PACKED_PRACTICA_1.exe.

To abbreviate, I will call the main machine where I have IDA installed "MAIN" and I will call the Windows 10 image with VMWARE "TARGET".

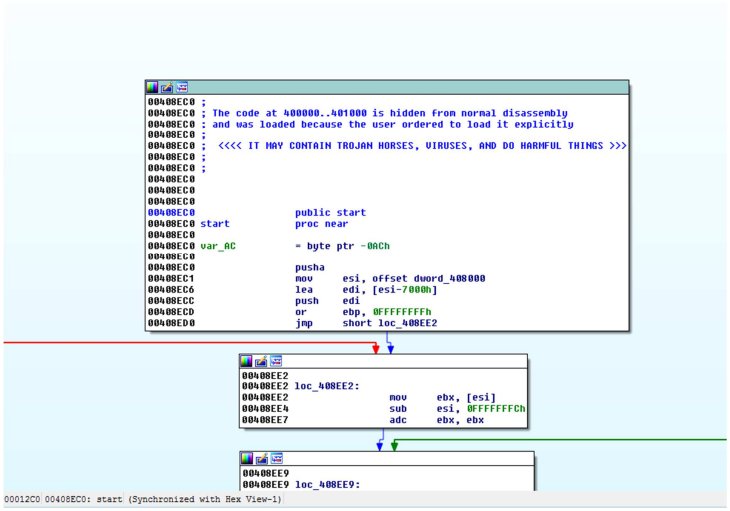


There, it is the packed file in the TARGET. I have to copy the remote server called win32_remote.exe there too. My Windows 10 is 64 bits, but the packed file is 32 bits. So, I have to run the 32-bit server.



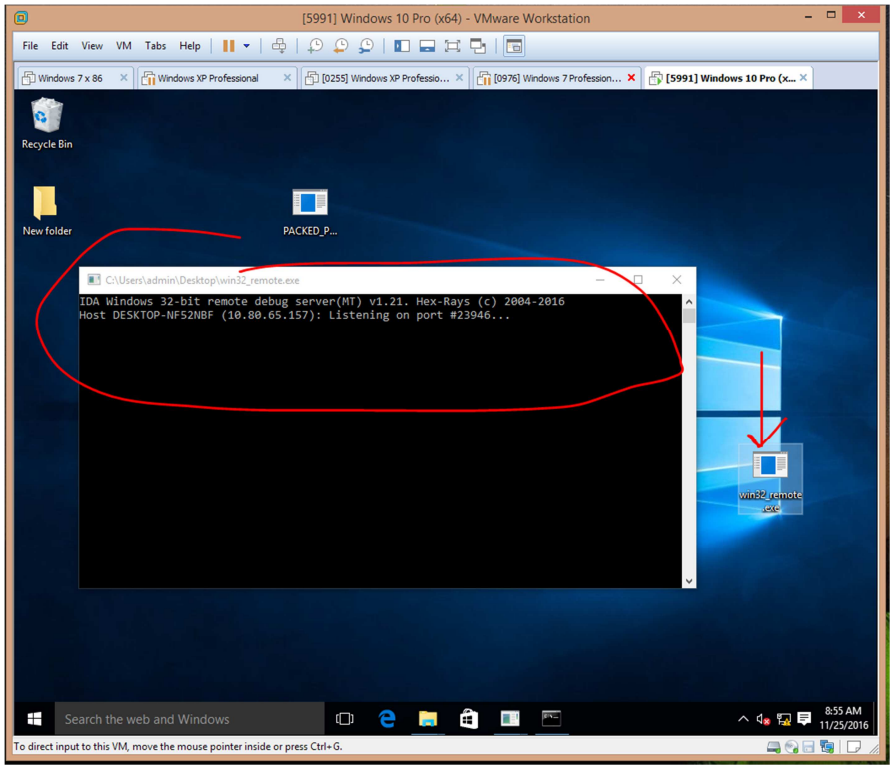
Let's copy PACKED_PRACTICA_1.exe in MAIN to do the local analysis and we open it in the loader checking MANUAL LOAD for it to analyze it and load all sections.

There, we are in the loader showing the packed program EntryPoint. We haven't run the debugger yet.



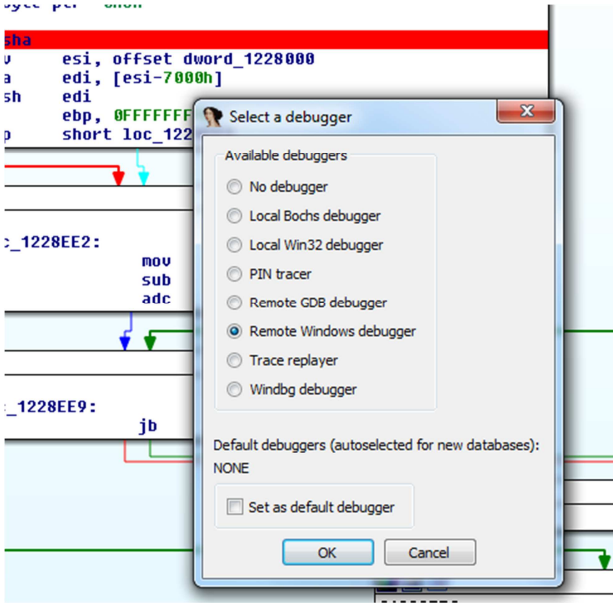
It's very important not to rename the IDB. That means that if the executable is PACKED_PRACTICA_1.exe in MAIN, when analyzing, it will save an IDB in the same folder and it should be PACKED_PRACTICA_1.idb no other name because there will be problems recognizing that the remote process belongs to the same executable analyzed locally.

Let's run the win32_remote.exe remote server in the TARGET.

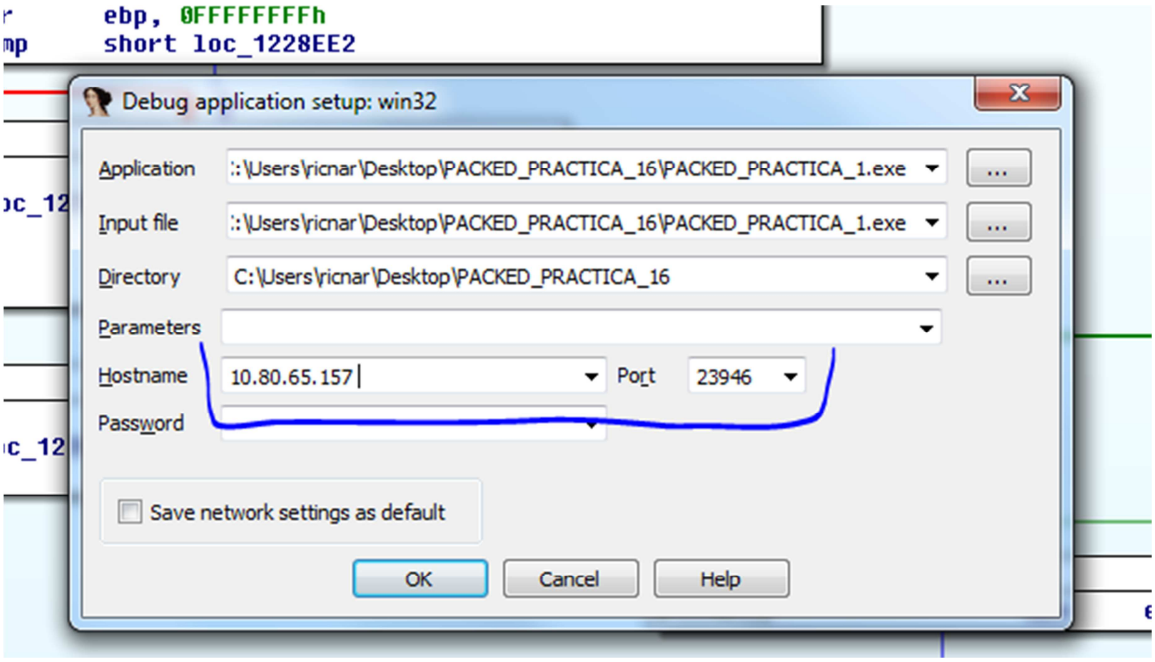


There, the IDA remote server runs and it will listen with the IP and port that it says there. In my case, IP 10.80.65.157 and port 23946. Copy the data it shows for you.

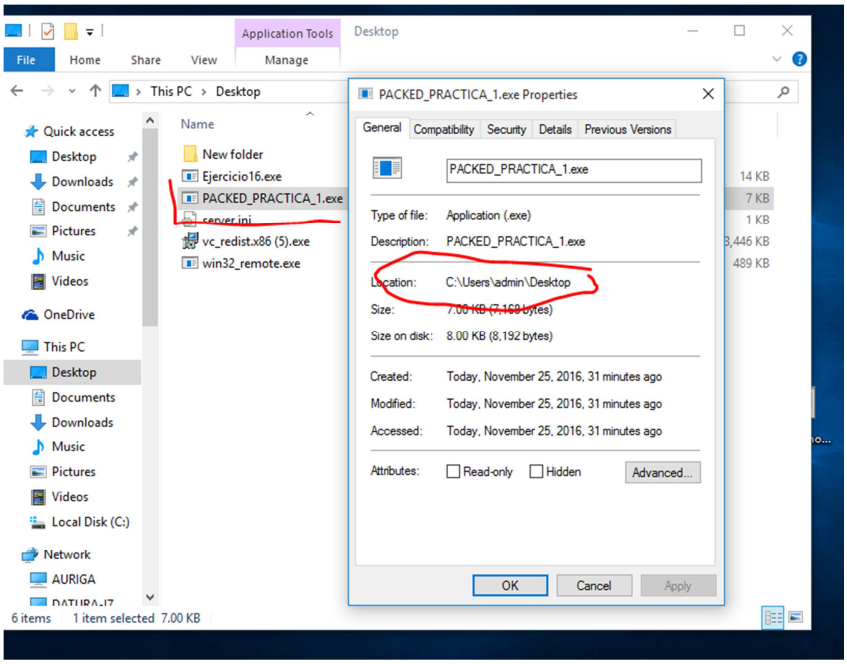
Change the debugger to:



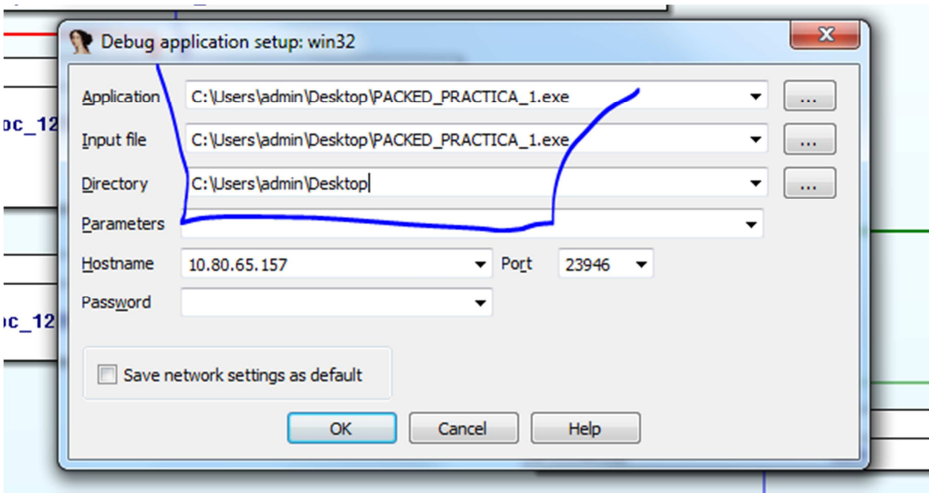
In PROCESS OPTIONS, enter IDA Server IP and PORT and as we need to run it from the beginning to unpack it because we can't attach it. We would find it running; we have to fix the paths that it shows to match the executable location in the TARGET.



In my case, the unpacked file in the TARGET is on the DESKTOP and the path for my PC will be **C:\Users\admin\Desktop\PACKED_PRACTICA_1.exe**

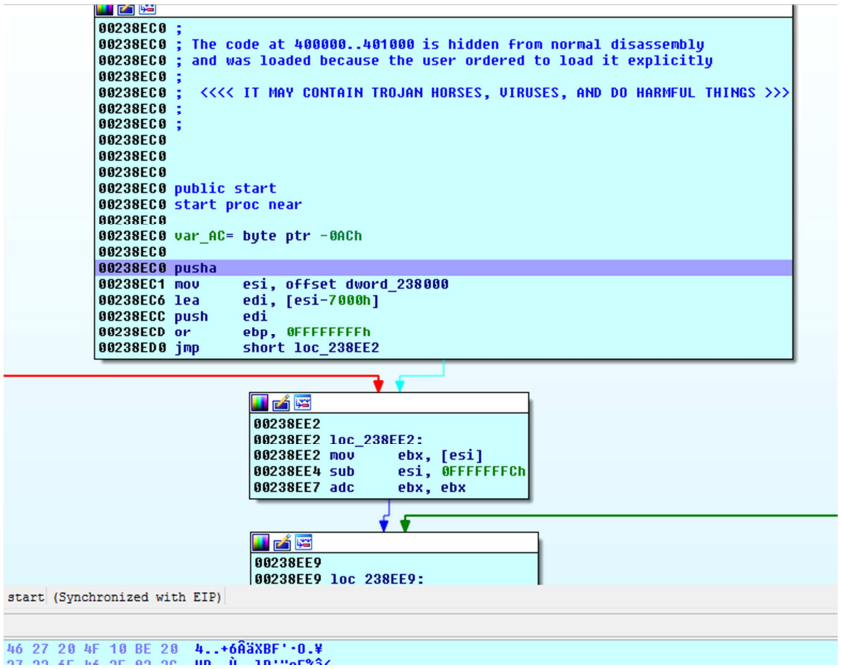


So, I fix it.



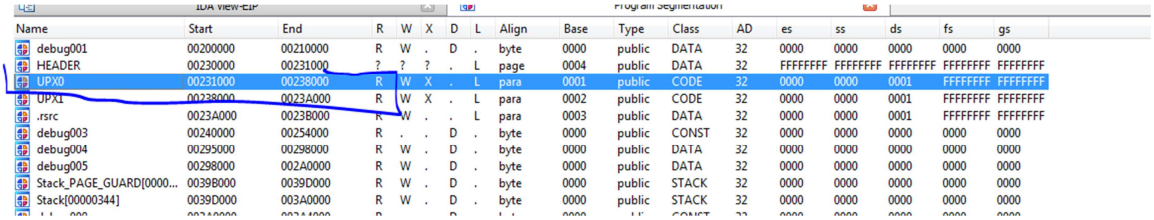
Now, we click START PROCESS and it will stop at the EntryPoint in Debugger Mode.

The executable has randomization so the addresses vary in each run. So, it's important since we run it, we dump it and fix the IAT in the same process without closing it for the address not to change.



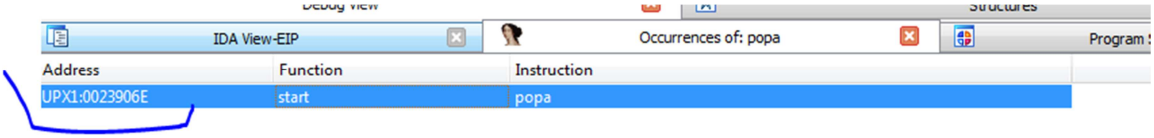
Let's see the first code section after the header in SEGMENTS.

In my case, it starts at 231000 and ends at 238000.



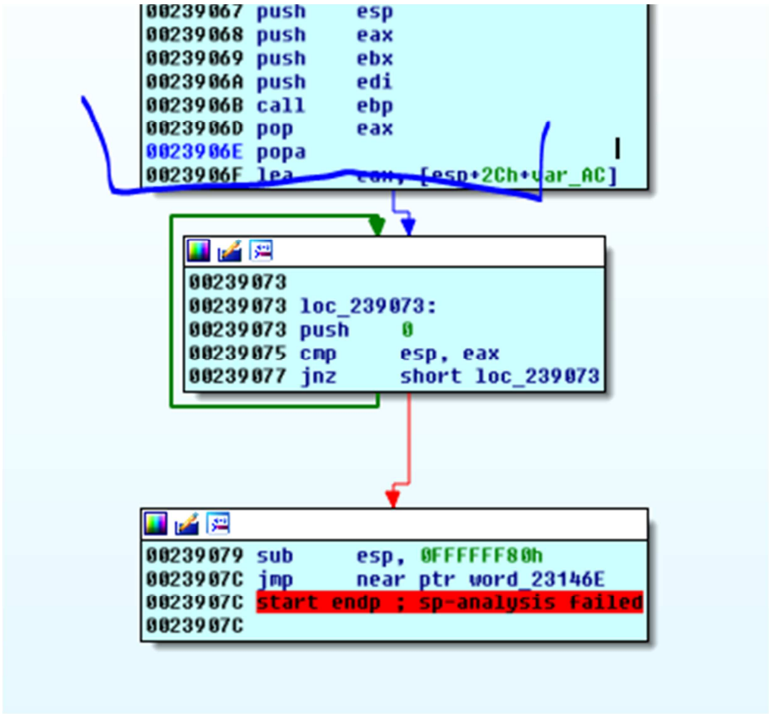
Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	es	ss	ds	fs	gs
debug001	00200000	00210000	R	W	.	D	.	byte	0000	public	DATA	32	0000	0000	0000	0000	0000
HEADER	00230000	00231000	?	?	?	.	L	page	0004	public	DATA	32	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
UPX0	00231000	00238000	R	W	X	.	L	para	0001	public	CODE	32	0000	0000	0001	FFFFFFFF	FFFFFFFF
UPX1	00238000	0023A000	R	W	X	.	L	para	0002	public	CODE	32	0000	0000	0001	FFFFFFFF	FFFFFFFF
.rsrc	0023A000	0023B000	R	W	.	.	L	para	0003	public	DATA	32	0000	0000	0001	FFFFFFFF	FFFFFFFF
debug003	00240000	00254000	R	.	.	D	.	byte	0000	public	CONST	32	0000	0000	0000	0000	0000
debug004	00295000	00298000	R	W	.	D	.	byte	0000	public	DATA	32	0000	0000	0000	0000	0000
debug005	00298000	002A0000	R	W	.	D	.	byte	0000	public	DATA	32	0000	0000	0000	0000	0000
Stack_PAGE_GUARD[0000...	0039B000	0039D000	R	W	.	D	.	byte	0000	public	STACK	32	0000	0000	0000	0000	0000
Stack[00000344]	0039D000	003A0000	R	W	.	D	.	byte	0000	public	STACK	32	0000	0000	0000	0000	0000

If we just use SEARCH FOR TEXT to find the POPAD or POPA, in this case, we would find the one that is executed before jumping to the OEP.



IDA View-EIP | Occurrences of: popa | Program

Address	Function	Instruction
UPX1:0023906E	start	popa



```
00239067 push    esp
00239068 push    eax
00239069 push    ebx
0023906A push    edi
0023906B call    ebp
0023906D pop     eax
0023906E popa
0023906F lea     esp, [esp+2Ch+var_AC]
```

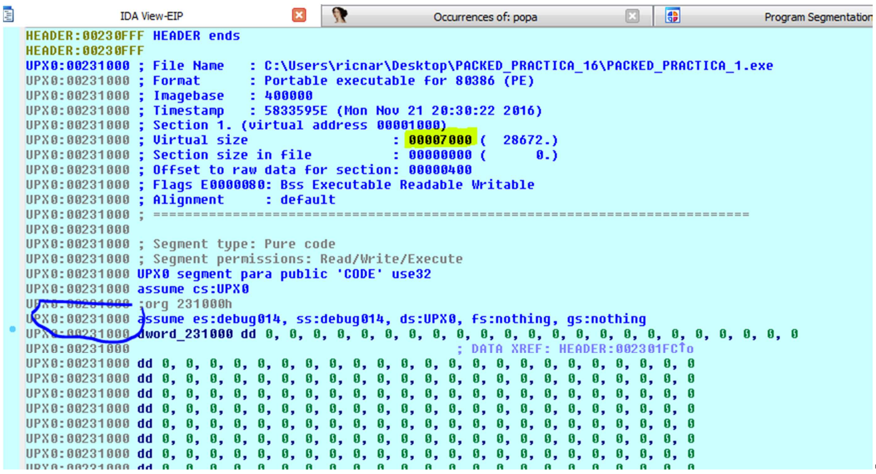
```
00239073
00239073 loc_239073:
00239073 push    0
00239075 cmp     esp, eax
00239077 jnz     short loc_239073
```

```
00239079 sub     esp, 0FFFFFF80h
0023907C jmp     near ptr word_23146E
0023907C start endp ; sp-analysis failed
0023907C
```

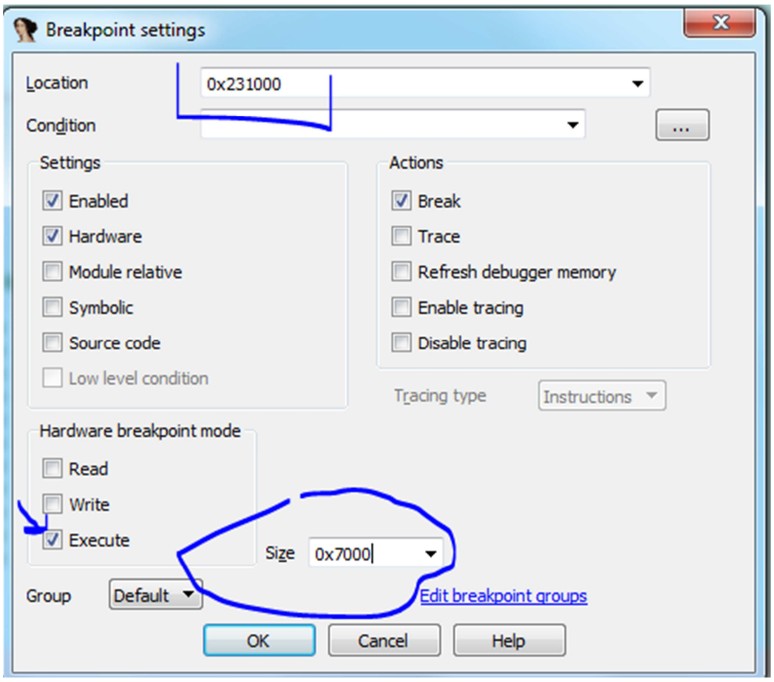
There, we would know that the OEP is at 0x23146E, but I can find it setting a BreakPoint on Execution that covers all the first section.

I go to the start of it at 0x231000.

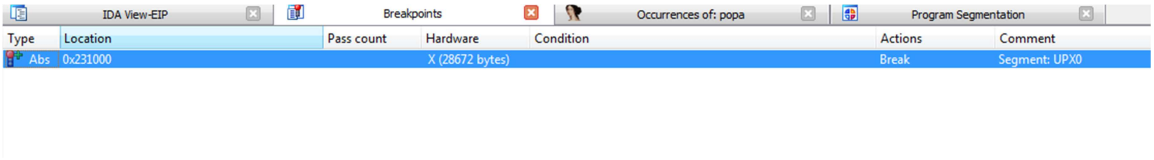
There, I see the header data; obviously, the addresses don't match because of the randomization. The ImageBase is not 0x400000, but the VIRTUAL SIZE is the same 0x7000. The section size in memory matches because it starts at 0x231000 and finishes at 0x238000. The difference is 0x7000 bytes.



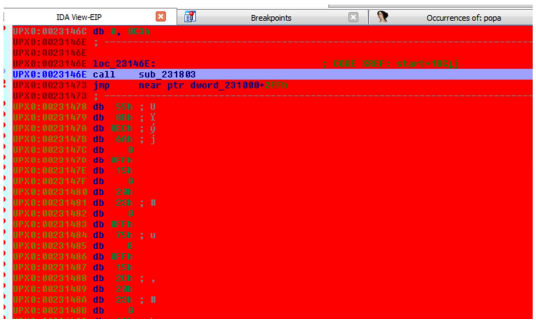
I will set a BreakPoint with F2 at the start of the section, in my case, 0x231000 or the correspondent address in your machine with a size of 0x7000.



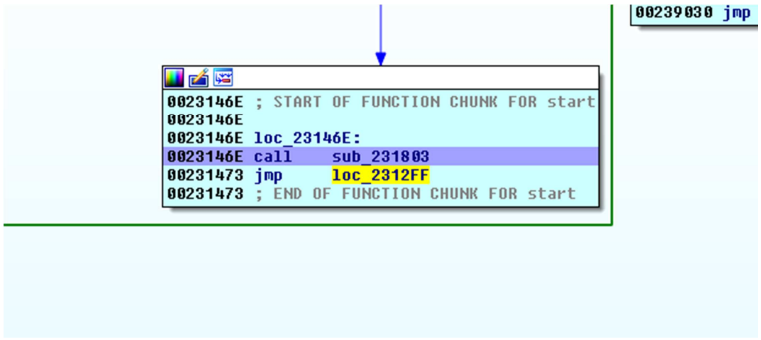
I clear the other BreakPoints. I will just leave this one and press F9 to run it.



It stops there and matches the address we had seen that was the OEP. Now, I clear the BreakPoint to get rid of the red background.

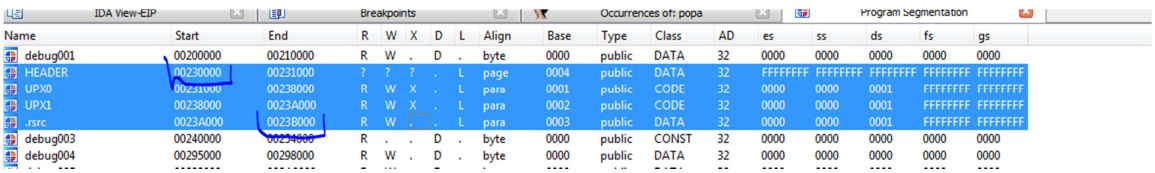


Right click on the left upper corner to REANALYZE PROGRAM.

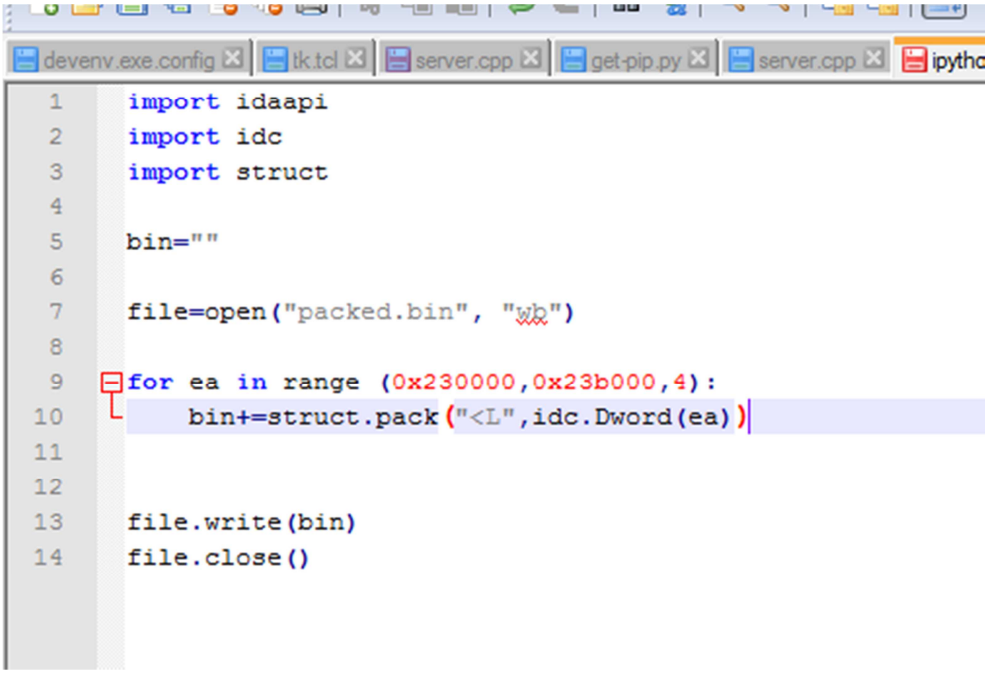


In this case, it took it as a block of the same STUB. That's why it didn't add the sub_ suffix and left it as loc_. No problem anyway.

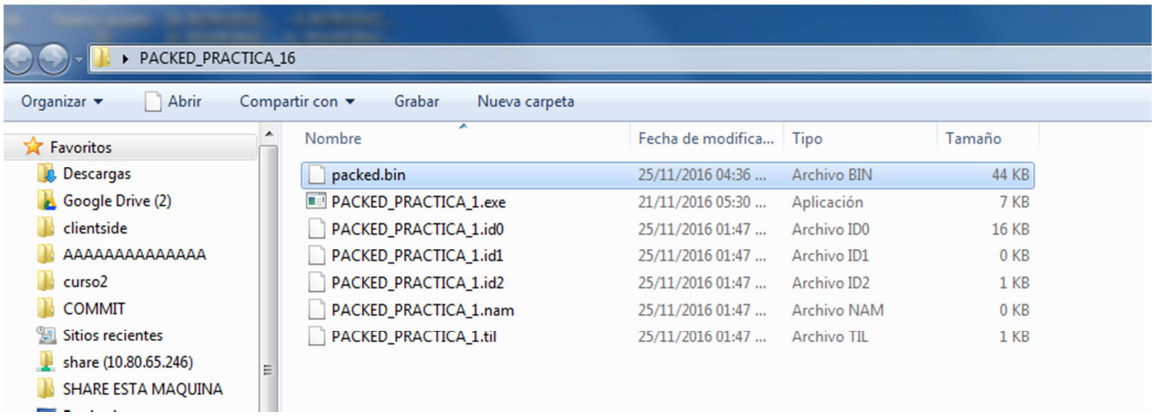
I edit the dumper script in Python to show my real ImageBase and the end of the file.



The ImageBase, in my case, is 0x230000 and the end is 0x23B000.

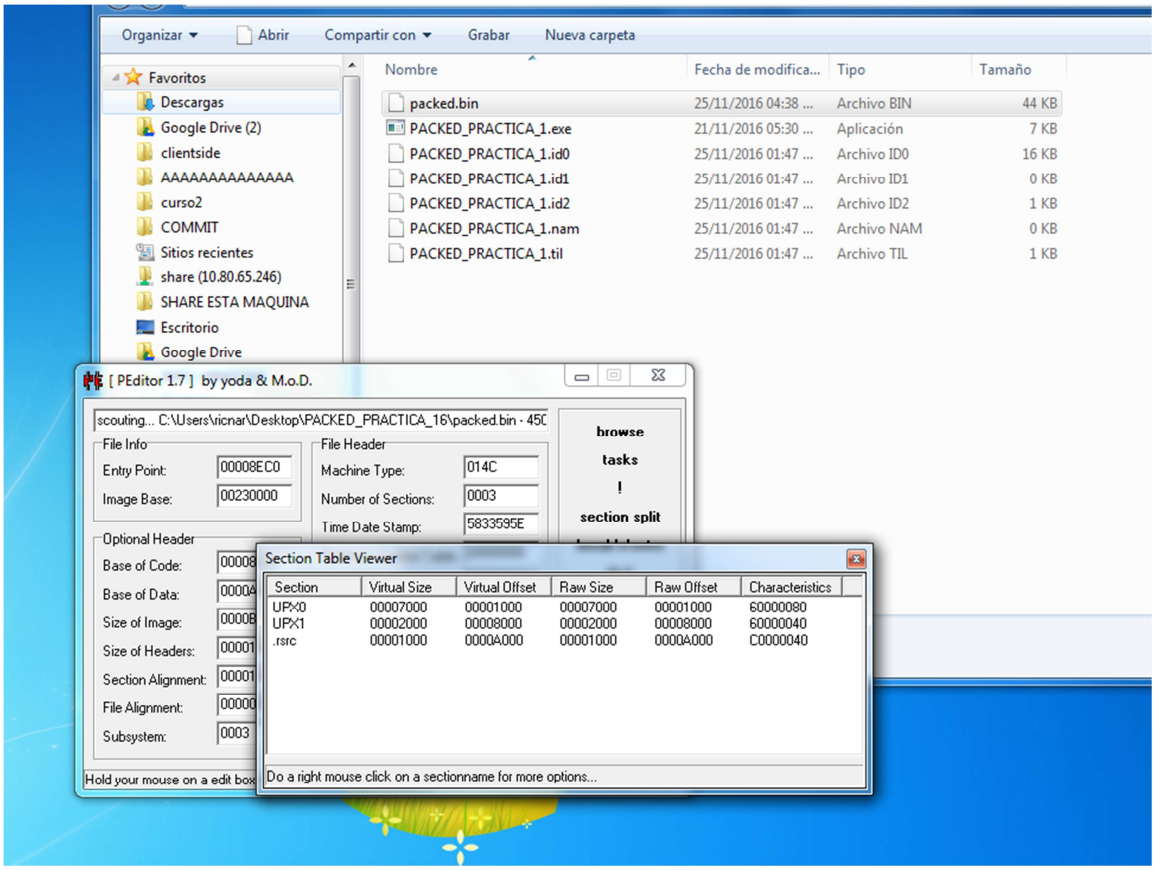


There, I change the values and run it with FILE-SCRIPT-FILE.

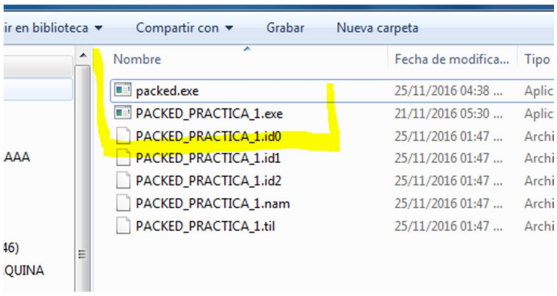


It saves it there because the script runs in the MAIN machine.

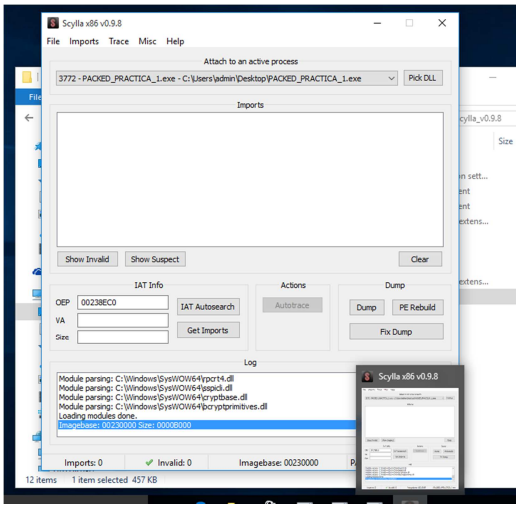
I open it with PEEDITOR and I select DUMPFIXER.



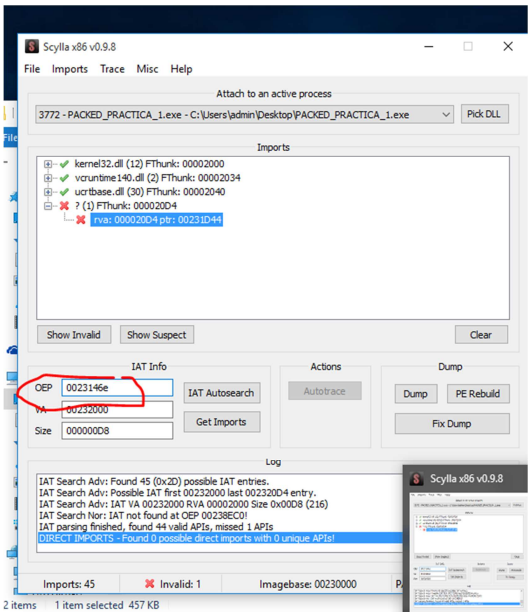
I rename it as exe.



It appears with the same icon of the packed file. I open Scylla in the TARGET and copy the packed.exe there.



I press IAT AUTOSEARCH.

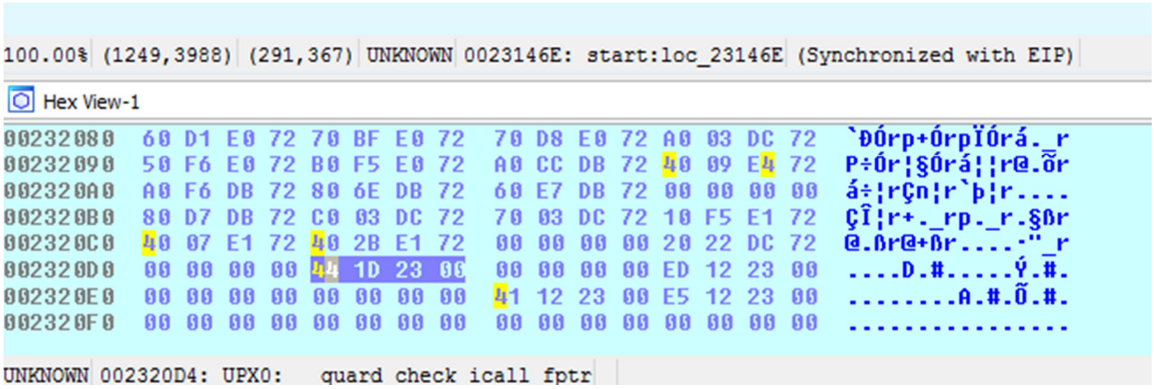


I change the OEP to 0x23146E that is the value we found.

There is only one invalid API after IAT AUTOSEARCH and GET IMPORTS.

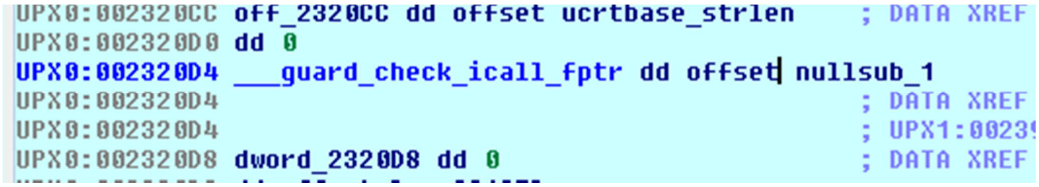
Adding the ImageBase to the invalid API.

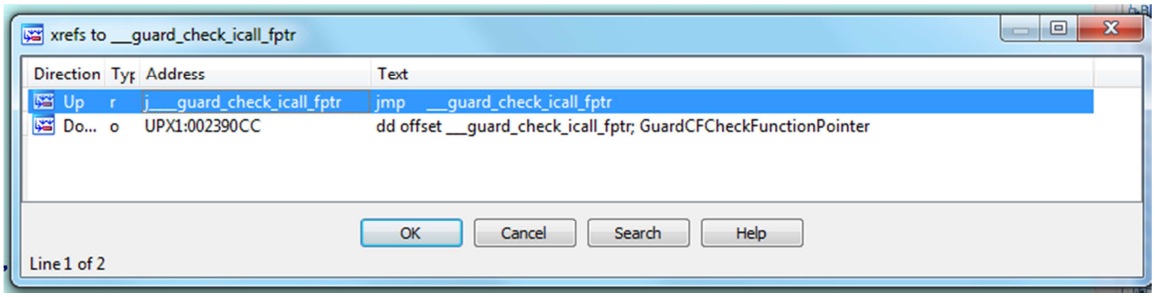
Python>hex(0x230000+0x20d4)
0x2320d4



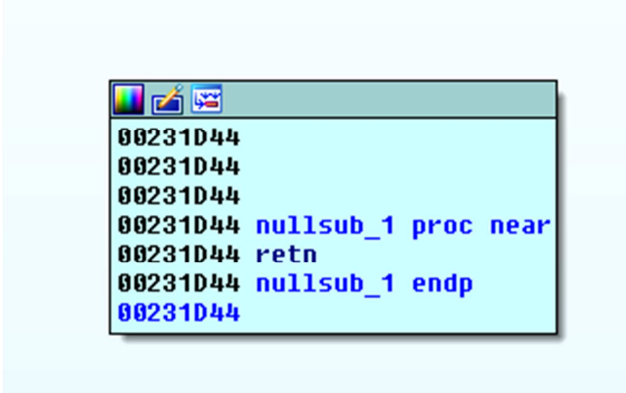
It doesn't look like a part of the IAT. Let's check by seeing in the disassembly list if that address has references.

We see it really does.



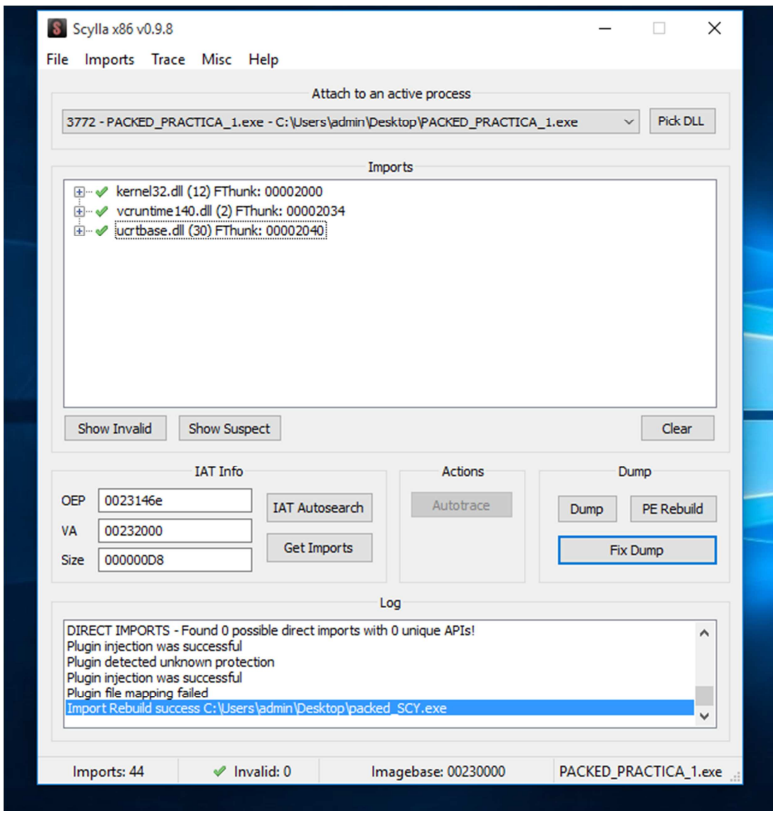


Let's see where it goes.



It ends up going to a RET.

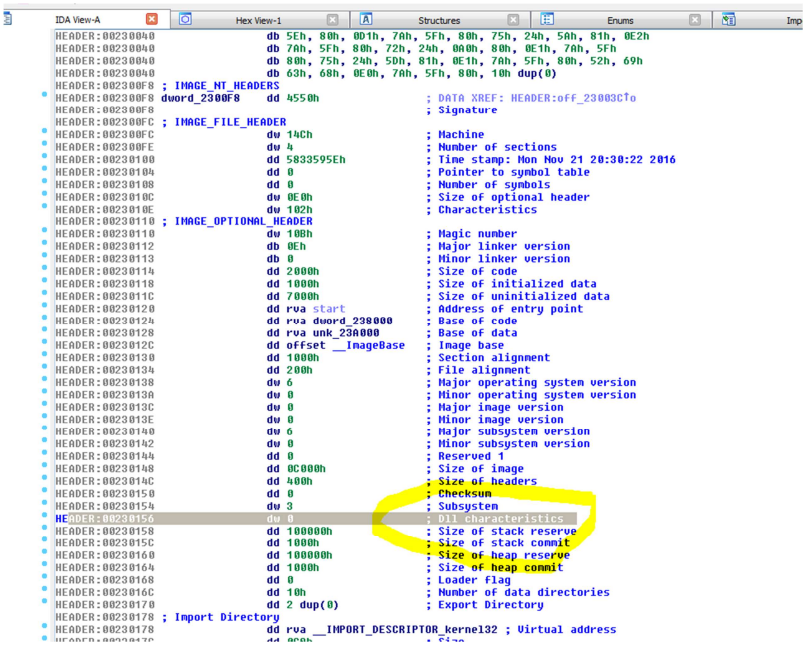
There no problem. It goes to a constant value in the program that is a RET and not an API. So, I will right click - CUT THUNK to delete it from the IAT.



Now, I press FIX DUMP in the packed.exe and I have packed_SCY.exe, but it doesn't run. This happens because, in the dumped, the addresses created in runtime that don't belong to the IAT are not reallocated because they always change. So, I will clear the randomization. I open the packed_SCY.exe in IDA with Manual Load loading the header too and I go there.

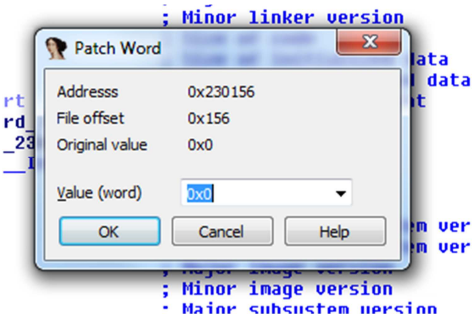
```
HEADER:00230000 ; Segment type: Pure data
HEADER:00230000 HEADER          segment page public 'DATA' use32
HEADER:00230000                 assume cs:HEADER
HEADER:00230000                 ;org 230000h
HEADER:00230000 ; unsigned __int8 _ImageBase
HEADER:00230000 __ImageBase     dw 5A4Dh                ; DATA XREF: HEADER:off_23003C↓o
HEADER:00230000                                         ; HEADER:00230120↓o ...
HEADER:00230000                                         ; PE magic number
HEADER:00230002                 dw 90h                  ; Bytes on last page of file
HEADER:00230004                 dw 3                    ; Pages in file
HEADER:00230006                 dw 0                    ; Relocations
HEADER:00230008                 dw 4                    ; Size of header in paragraphs
HEADER:0023000A                 dw 0                    ; Minimum extra paragraphs needed
HEADER:0023000C                 dw 0FFFFh               ; Maximum extra paragraphs needed
HEADER:0023000E                 dw 0                    ; Initial (relative) SS value
HEADER:00230010                 dw 0B8h                 ; Initial SP value
HEADER:00230012                 dw 0                    ; Checksum
HEADER:00230014                 dw 0                    ; Initial IP value
HEADER:00230016                 dw 0                    ; Initial (relative) CS value
HEADER:00230018 word_230018     dw 40h                  ; DATA XREF: ___scrt_is_nonwritable_in_current_image+34↓r
HEADER:00230018                                         ; File address of relocation table
HEADER:0023001A                 dw 0                    ; Overlay number
HEADER:0023001C                 dw 4 dup(0)             ; Reserved words
HEADER:00230024                 dw 0                    ; OEM identifier (for e_oeminfo)
HEADER:00230026                 dw 0                    ; OEM information; e_oemid specific
HEADER:00230028                 dw 0Ah dup(0)           ; Reserved words
HEADER:0023003C off_23003C      dd offset dword_2300F8 - offset __ImageBase
HEADER:0023003C                                         ; DATA XREF: ___scrt_is_nonwritable_in_current_image+1E↓r
HEADER:0023003C                                         ; File address of new exe header
HEADER:00230040                 db 0Eh, 1Fh, 0BAh, 0Eh, 0, 0B4h, 9, 0CDh, 21h, 0B8h, 1 ; DOS Stub code
HEADER:00230040                 db 4Ch, 0CDh, 21h, 54h, 68h, 69h, 73h, 20h, 70h, 72h, 6Fh
HEADER:00230040                 db 67h, 72h, 61h, 6Dh, 20h, 63h, 61h, 2 dup(6Eh), 6Fh
HEADER:00230040                 db 74h, 20h, 62h, 65h, 20h, 72h, 75h, 6Eh, 20h, 69h, 6Eh
HEADER:00230040                 db 20h, 44h, 4Fh, 53h, 20h, 6Dh, 6Fh, 64h, 65h, 2Eh, 2 dup(0Dh)
HEADER:00230040                 db 0Ah, 24h, 7 dup(0), 0A4h, 1Bh, 31h, 0D3h, 0E0h, 7Ah
HEADER:00230040                 db 5Fh, 80h, 0E0h, 7Ah, 5Fh, 80h, 0E0h, 7Ah, 5Fh, 80h
HEADER:00230040                 db 0E9h, 2, 0CCh, 80h, 0EEh, 7Ah, 5Fh, 80h, 0DBh, 24h
HEADER:00230040                 db 5Eh, 81h, 0E3h, 7Ah, 5Fh, 80h, 0DBh, 24h, 5Ch, 81h
HEADER:00230040                 db 0E1h, 7Ah, 5Fh, 80h, 0DBh, 24h, 5Ah, 81h, 0F2h, 7Ah
HEADER:00230040                 db 5Fh, 80h, 0DBh, 24h, 5Bh, 81h, 0EDh, 7Ah, 5Fh, 80h
HEADER:00230040                 db 3Dh, 85h, 94h, 80h, 0E2h, 7Ah, 5Fh, 80h, 0E0h, 7Ah
HEADER:00230040                 db 5Eh, 80h, 0D1h, 7Ah, 5Fh, 80h, 75h, 24h, 5Ah, 81h, 0E2h
HEADER:00230040                 db 7Ah, 5Fh, 80h, 72h, 24h, 0A0h, 80h, 0E1h, 7Ah, 5Fh
HEADER:00230040                 db 80h, 75h, 24h, 5Dh, 81h, 0E1h, 7Ah, 5Fh, 80h, 52h, 69h
HEADER:00230040                 db 63h, 68h, 0E0h, 7Ah, 5Fh, 80h, 10h dup(0)
HEADER:002300F8 ; IMAGE_NT_HEADERS
HEADER:002300F8 dword_2300F8    dd 4550h                ; DATA XREF: HEADER:off_23003C↑o
HEADER:002300F8                                         ; Signature
HEADER:002300FC ; IMAGE_FILE_HEADER
HEADER:002300FC                 dw 14Ch                 ; Machine
HEADER:00230100 HEADER:__ImageBase (Synchronized with Hex View-1)
```

There, we have the DLL CHARACTERISTICS. In yours, it will be different from 0 because I already changed it.



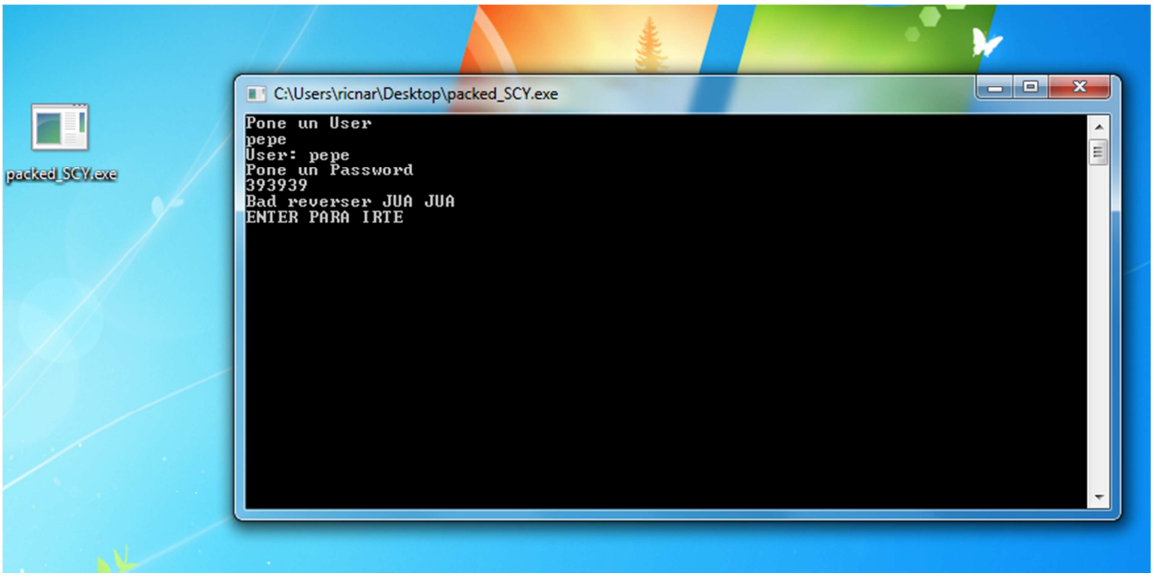

```
HEADER:00230040                 db 5Eh, 80h, 0D1h, 7Ah, 5Fh, 80h, 75h, 24h, 5Ah, 81h, 0E2h
HEADER:00230040                 db 7Ah, 5Fh, 80h, 72h, 24h, 0A0h, 80h, 0E1h, 7Ah, 5Fh
HEADER:00230040                 db 80h, 75h, 24h, 5Dh, 81h, 0E1h, 7Ah, 5Fh, 80h, 52h, 69h
HEADER:00230040                 db 63h, 68h, 0E0h, 7Ah, 5Fh, 80h, 10h dup(0)
HEADER:002300F8 ; IMAGE_NT_HEADERS
HEADER:002300F8 dword_2300F8    dd 4550h                ; DATA XREF: HEADER:off_23003C↑o
HEADER:002300F8                                         ; Signature
HEADER:002300FC ; IMAGE_FILE_HEADER
HEADER:002300FC                 dw 14Ch                 ; Machine
HEADER:002300FE                 dw 4                    ; Number of sections
HEADER:00230100                 dd 5833595Eh            ; Time stamp: Mon Nov 21 20:30:22 2016
HEADER:00230104                 dd 0                    ; Pointer to symbol table
HEADER:00230108                 dd 0                    ; Number of symbols
HEADER:0023010C                 dw 0E0h                 ; Size of optional header
HEADER:0023010E                 dw 102h                 ; Characteristics
HEADER:00230110 ; IMAGE_OPTIONAL_HEADER
HEADER:00230110                 dw 10Bh                 ; Magic number
HEADER:00230112                 db 0Eh                  ; Major linker version
HEADER:00230113                 db 0                    ; Minor linker version
HEADER:00230114                 dd 2000h                ; Size of code
HEADER:00230118                 dd 1000h                ; Size of initialized data
HEADER:0023011C                 dd 7000h                ; Size of uninitialized data
HEADER:00230120                 dd rva start            ; Address of entry point
HEADER:00230124                 dd rva dword_238000     ; Base of code
HEADER:00230128                 dd rva unk_23A000       ; Base of data
HEADER:0023012C                 dd offset __ImageBase   ; Image base
HEADER:00230130                 dd 1000h                ; Section alignment
HEADER:00230134                 dd 200h                 ; File alignment
HEADER:00230138                 dw 6                    ; Major operating system version
HEADER:0023013A                 dw 0                    ; Minor operating system version
HEADER:0023013C                 dw 0                    ; Major image version
HEADER:0023013E                 dw 0                    ; Minor image version
HEADER:00230140                 dw 6                    ; Major subsystem version
HEADER:00230142                 dw 0                    ; Minor subsystem version
HEADER:00230144                 dd 0                    ; Reserved 1
HEADER:00230148                 dd 0C000h               ; Size of image
HEADER:0023014C                 dd 400h                 ; Size of headers
HEADER:00230150                 dd 0                    ; Checksum
HEADER:00230154                 dw 3                    ; Subsystem
HEADER:00230156                 dw 0                    ; DLL characteristics
HEADER:00230158                 dd 100000h              ; Size of stack reserve
HEADER:0023015C                 dd 1000h                ; Size of stack commit
HEADER:00230160                 dd 100000h              ; Size of heap reserve
HEADER:00230164                 dd 1000h                ; Size of heap commit
HEADER:00230168                 dd 0                    ; Loader flag
HEADER:0023016C                 dd 10h                  ; Number of data directories
HEADER:00230170                 dd 2 dup(0)             ; Export Directory
HEADER:00230178 ; Import Directory
HEADER:00230178                 dd rva __IMPORT_DESCRIPTOR_kernel32 ; Virtual address
```

With the EDIT-PATCH PROGRAM-CHANGE WORD menu change it to 0.



Then, save it there with APPLY PATCHES TO INPUT FILE.

And that's all.



It is already unpacked. We will reverse it in part 18.

Ricardo Narvaja

Translated by: @IvinsonCLS