

REVERSING WITH IDA PRO FROM SCRATCH

PART 15

In the previous part, we saw a pair of many methods to detect and get the OEP in a packed file. Now, we'll continue with the two missing steps: The dumped and the IAT rebuilding explained here.

The screenshot shows the IDA Pro interface. The assembly pane displays the following code:

```
00401000
00401000
00401000
00401000 sub_401000 proc near
00401000 push 0
00401002 call sub_401506
00401007 mov dword_4020CA, eax
0040100C push 0
0040100E push offset aNoNeedToDisasm ; "No need to disasm the code!"
00401013 call sub_4014BE
00401018 or eax, eax
0040101A jz short loc_40101D
```

The memory dump pane below shows the raw bytes of the code:

: 20 4D 33 30 08 0E 65 53 32 D8 80 40 4F 16	'..-M30..eS2+C@0.
CE 5E 8D 2E 3F DE 27 50 30 7B 99 92 18 50	@A+^..? 'P@{0@.P
63 AC 1B 68 65 97 43 1A 27 89 90 32 00 00	src%heuC.'ë.2..
0D 79 EA 24 00 00 00 FF 00 00 00 00 00 00	p}.y0\$.....
00 90 40 00 8D BE 00 80 FF FF 57 EB 0B 90	+..@...+.ç--Wd..

We go to the OEP and Reanalyze the program to have everything ready to Dump.

Let's use an IDC script not Python.

```
static main()
{
auto fp, ea;
fp = fopen("pepe.bin", "wb");
for ( ea=0x400000; ea < 0x40b200; ea++ )
    fputc(Byte(ea), fp);
}
```

In the script, we enter the ImageBase 0x400000 and the greater address we see in IDA Segments tab of the last section to be dumped, in this case, the Overlay section that ends in 0x40b200.

	00400000	00400000	R W . D .	byte	0000	p	EDX 0040981
TIB[000049A0]	003F0000	003AC000	R W . D .	byte	0000	p	ESI 0040981
HEADER	00400000	00401000	? ? ? . L	paqe	0004	p	EDI 0040981
UPX0	00401000	00409000	R W X . L	para	0001	p	EBP 0019FF1
UPX1	00409000	0040A000	R W X . L	para	0002	p	ESP 0019FF1
.rsrc	0040A000	0040B000	R W . L	para	0003	p	EIP 0040101
OVERLAY	0040B000	0040B200	R W . L	byte	0000	p	EFL 0000021
debug033	00445000	00448000	R W . D .	byte	0000	p v	

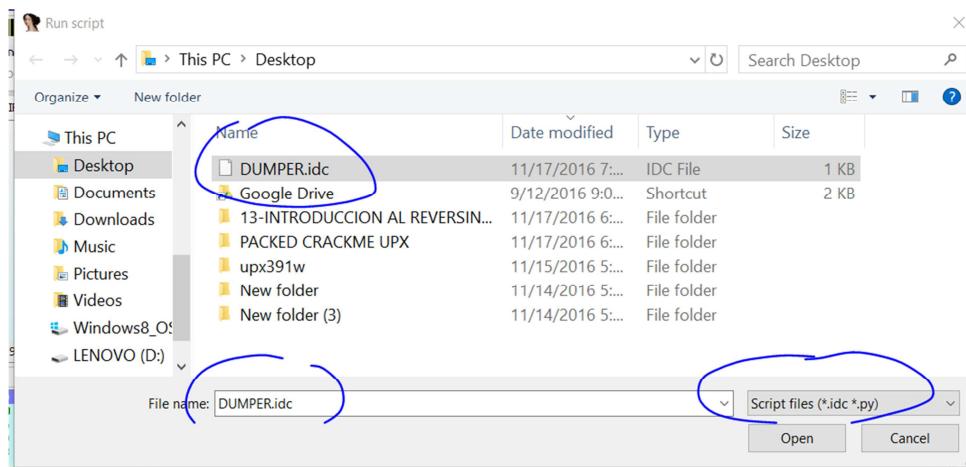
Line 3 of 191

Untitled - Notepad

File Edit Format View Help

```
static main()
{
auto fp, ea;
fp = fopen("pepe.bin", "wb");
for ( ea=0x400000; ea < 0x40b200; ea++ )
    fputc(Byte(ea), fp);
}
```

Let's save it as DUMPER.idc



And run it in FILE-SCRIPT FILE. It accepts Python and IDC scripts. So, there's no problem.

PACKED CRACKME UPX

Name	Date modified	Type	Size
<input checked="" type="checkbox"/> 14-INTRODUCCION AL REVERS...	11/15/2016 7:...	7z Archive	986 KB
<input type="checkbox"/> 14-INTRODUCCION AL REVERS...	11/15/2016 7:...	Microsoft Wor...	1,057 KB
<input checked="" type="checkbox"/> PACKED_CRACKME.EXE	11/15/2016 7:...	Application	7 KB
<input type="checkbox"/> PACKED_CRACKME.id0	11/17/2016 6:...	ID0 File	168 KB
<input type="checkbox"/> PACKED_CRACKME.id1	11/17/2016 6:...	ID1 File	192 KB
<input checked="" type="checkbox"/> PACKED_CRACKME.idb	11/15/2016 7:...	IDA Database	377 KB
<input type="checkbox"/> PACKED_CRACKME.nam	11/17/2016 6:...	NAM File	16 KB
<input type="checkbox"/> PACKED_CRACKME.til	11/17/2016 6:...	TIL File	1 KB
<input type="checkbox"/> pepe.bin	11/17/2016 7:...	BIN File	45 KB

I make a backup and rename it with the .exe extension. (If you don't see the extensions, modify that option in Folder Options in Windows)

PACKED CRACKME UPX

File Home Share Application Tools Manage

Navigation pane Panes Details pane Layout

Current view Show/hide

Folder Options

General View Search

Folder views You can apply this view (such as Details or Icons) to all folders of this type.

Apply to Folders Reset Folders

Advanced settings:

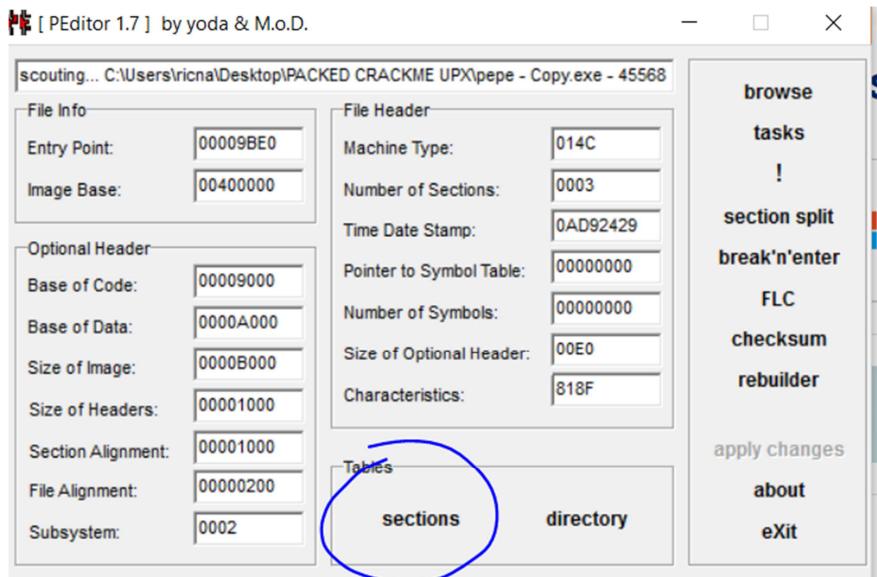
- Files and Folders
 - Always show icons, never thumbnails-OFF
 - Always show menus-OFF
 - Display file icon on thumbnails-ON
 - Display file name in folder tips-ON
 - Display full path in the title bar-OFF
 - Hidden files and folders
 - Don't show hidden files, folders, or drives-ON
 - Show hidden files, folders, and drives-OFF
 - Hide empty drives-ON
 - Hide extensions for known file types-OFF
 - Hide folder merge conflicts-ON
 - Hide non-inherited inheritance stream files (Recommended)-ON

Name	Date modified	Type	Size
Scylla v0.9.7c	11/17/2016		
<input checked="" type="checkbox"/> 14-INTRODUCCION AL REVERS...	11/15/2016		
<input checked="" type="checkbox"/> 14-INTRODUCCION AL REVERS...	11/15/2016		
logfile_scylla_plugin.txt	11/17/2016		
<input checked="" type="checkbox"/> PACKED_CRACKME.EXE	11/15/2016		
<input type="checkbox"/> PACKED_CRACKME.id0	11/17/2016		
<input type="checkbox"/> PACKED_CRACKME.id1	11/17/2016		
<input checked="" type="checkbox"/> PACKED_CRACKME.idb	11/15/2016		
<input type="checkbox"/> PACKED_CRACKME.nam	11/17/2016		
<input type="checkbox"/> PACKED_CRACKME.til	11/17/2016		
<input checked="" type="checkbox"/> pepe - Copy.exe	11/17/2016		
<input checked="" type="checkbox"/> pepe - Copy.idb	11/17/2016		
<input checked="" type="checkbox"/> pepe - Copy.SCY.exe	11/17/2016		
<input type="checkbox"/> pepe - Copy.id0	11/17/2016		

<input type="checkbox"/> PACKED_CRACKME.nam	11/17/2016 6:...	NAM File	16 KB
<input type="checkbox"/> PACKED_CRACKME.til	11/17/2016 6:...	TIL File	1 KB
<input checked="" type="checkbox"/> pepe - Copy.exe	11/17/2016 7:...	Application	45 KB
<input type="checkbox"/> pepe.bin	11/17/2016 7:...	BIN File	45 KB

We see it doesn't have any icon. So, some steps are missing.

Let's decompress PE Editor 1.7 and run it.



In SECTIONS.

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
UPX0	00008000	00001000	00000000	00000000	60000080
UPX1	00001000	00009000	00000E00	00000400	60000040
.rsrc	00001000	0000A000	00000800	00001200	C0000040

Right click in each section and select DumpFixer.

The screenshot shows the 'Section Table Viewer' window with the 'UPX0' section selected. A context menu is open over the 'edit section' option, listing various actions: 'edit section', 'add a section', 'delete the section', 'copy the section to HD', 'move the section to HD', 'copy a section from HD to EOF', 'dumpfixer (RS=VS & RO=VO)', and 'set the characteristics to E0000020'. The 'dumpfixer (RS=VS & RO=VO)' option is circled in blue.

<input type="checkbox"/> PACKED_CRACKME.nam	11/17/2016 6...	NAM File	16 KB
<input type="checkbox"/> PACKED_CRACKME.til	11/17/2016 6...	TIL File	1 KB
<input checked="" type="checkbox"/> pepe - Copy.exe	11/17/2016 7...	Application	45 KB
<input type="checkbox"/> pepe.bin	11/17/2016 7...	BIN File	45 KB

We are closer. At least, we see the icon, but it doesn't run yet because we need to fix the IAT.

What is the IAT?

The IAT or IMPORT ADDRESS TABLE is a table located in the executable and it is used when the program runs. It saves the imported function addresses there used to run the program in any machine.

If the IAT is correct and we give another person the executable, the IAT will be filled with the correspondent values of that machine independently of the OS and it will run normally.

The IAT will be in a determined position in each executable and it will have fix positions for each function to fill it.

Let's see the difference between the top image that belongs to the packed file when it was at the OEP before unpacking and the original one.

```
UPX0:00403228 off_403228 dd offset kernel32_GlobalAlloc ; DATA XREF: UPX0:004014EETr
UPX0:0040322C off_40322C dd offset kernel32_lstrlen ; DATA XREF: UPX0:004014F4Tr
UPX0:00403230 off_403230 dd offset kernel32_CloseHandle ; DATA XREF: UPX0:004014FATr
UPX0:00403234 off_403234 dd offset kernel32_WriteFile ; DATA XREF: UPX0:00401500Tr
UPX0:00403238 off_403238 dd offset kernel32_GetModuleHandleA ; DATA XREF: sub_401506Tr
UPX0:0040323C off_40323C dd offset kernel32_ReadFile ; DATA XREF: UPX0:0040150CTr
UPX0:00403240 off_403240 dd offset kernel32_ExitProcess ; DATA XREF: sub_401512Tr
UPX0:00403244 align 8
UPX0:00403248 off_403248 dd offset comctl32_InitCommonControls ; DATA XREF: UPX0:00401518Tr
UPX0:0040324C off_40324C dd offset comctl32_CreateToolbarEx ; DATA XREF: UPX0:0040151ETr
```

The content of 0x403028 is an offset (off_) or the GetmoduleHandleA API address and in the original exe, the same address is in the idata section and it has the address of the same API..

```
.idata:00403234 : 800L __stdcall WriteFile(HMODULE hFile, LPVOID lpBuffer, DWORD nNt
.idata:00403234      extrn WriteFile:dword          DATA XREF: CODE:00401500Tr
.idata:00403238 : HMODULE __stdcall GetModuleHandleA(LPCSTR lpModuleName)
.idata:00403238      extrn _imp_GetModuleHandleA:dword
.idata:00403238      ; DATA XREF: GetModuleHandleA
.idata:0040323C : 800L __stdcall ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNum
.idata:0040323C      extrn ReadFile:dword          ; DATA XREF: CODE:0040150CTr
```

Both show the 0x403238 address and it seems they have the same content. Let's open the original program too.

```

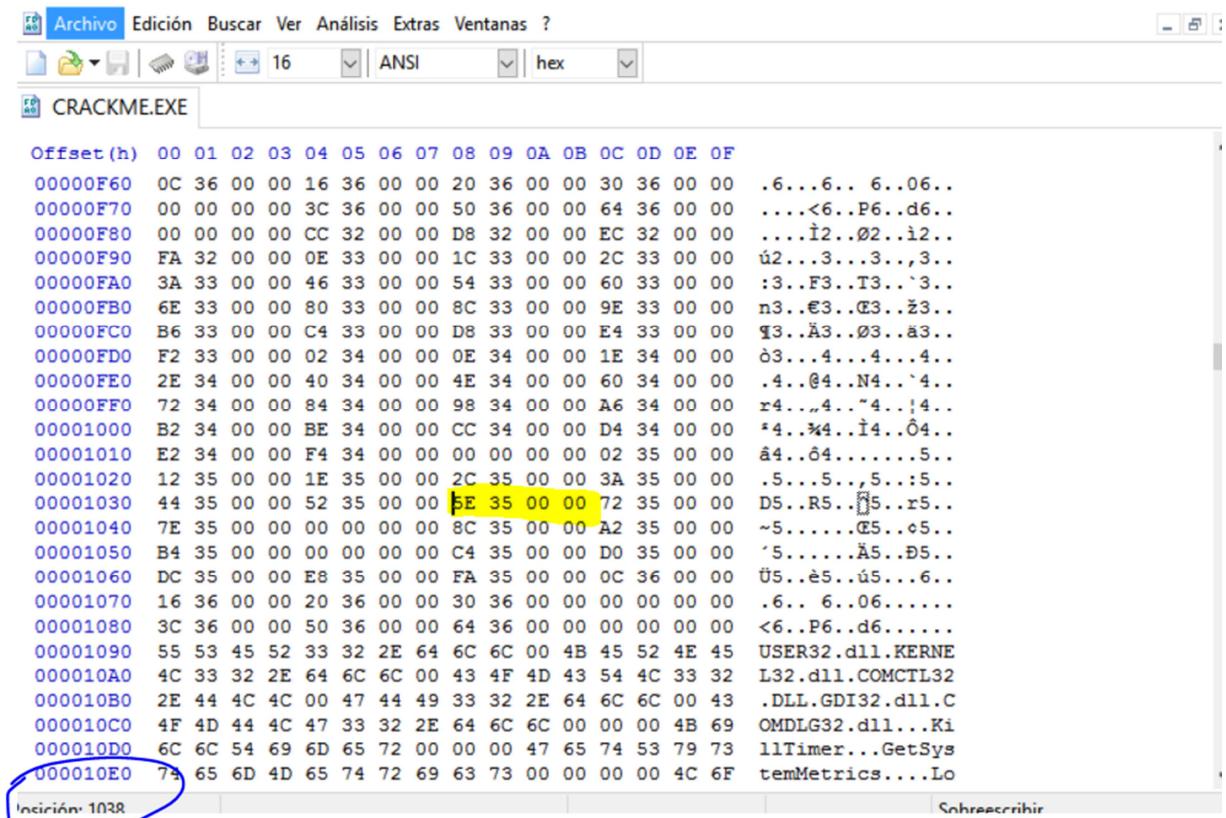
    .idata:00403234 ; BOOL __stdcall WriteFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToWrite, LPVOID lpOverlapped, LPOVERLAPPED lpCompletionRoutine, DWORD dwEventThread)
    .idata:00403234     extrn WriteFile:dword ; DATA XREF: CODE:00401500Tr
    .idata:00403238 ; HMODULE __stdcall GetModuleHandleA(LPCSTR lpModuleName)
    .idata:00403238     extrn __imp__GetModuleHandleA:dword
    .idata:00403238     ; DATA XREF: GetModuleHandleAtr
    .idata:0040323C ; BOOL __stdcall ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead, LPVOID lpOverlapped, LPOVERLAPPED lpCompletionRoutine, DWORD dwEventThread)
    .idata:0040323C     extrn ReadFile:dword ; DATA XREF: CODE:00401500Tr
    .idata:00403240 ; void __stdcall ExitProcess(UINT uExitCode)
    .idata:00403240     extrn __imp__ExitProcess:dword ; DATA XREF: ExitProcessTr
    .idata:00403244
    .idata:00403248 ; Imports from COMCTL32.DLL
    .idata:00403248
    .idata:00403248 ; void __stdcall InitCommonControls()
    .idata:00403248     extrn InitCommonControls:dword ; DATA XREF: CODE:00401518Tr
    .idata:0040324C ; HWND __stdcall CreateToolbarEx(HWND hwnd, DWORD ws, UINT wID, int nBitmapIndex)
    .idata:0040324C     extrn CreateToolbarEx:dword ; DATA XREF: CODE:0040151ETr
    .idata:00403250 ; HBITMAP __stdcall CreateToolbar(HWND hwnd, DWORD ws, UINT wID, int nBitmapIndex)
    .idata:00403250     extrn CreateToolbar:dword ; DATA XREF: CODE:00401524Tr
    .idata:00403254
    .idata:00403258 ; Imports from GDI32.dll
    .idata:00403258

```

00001038 00403238: .idata:_imp_GetModuleHandleA (Synchronized with Hex View-1)

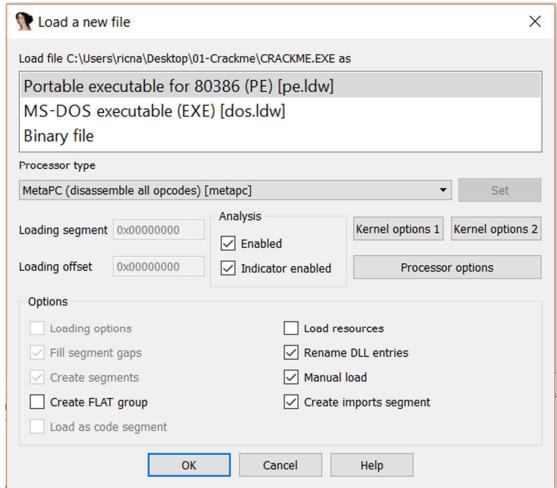
There, we see the 0x1038 File Offset to able to search in the HXD of the original one.

Let's open the original one in HXD.



The content of the 0x1038 offset is 0x355E.

If I add 0x355E to the 0x400000 ImageBase, we have 0x40355E. What is in there? To find it out, we should load the original one with Manual Load to load all the executable sections.



Accept all the sections and when it runs, go to 0x40355E and on the right, we see the API name: GetModuleHandleA.

Assembly code snippet:

```
0040351E 00 00 47 6C 6F 62 61 6C 46 72 65 65 00 00 00 00 .GlobalFree...
0040352E 47 6C 6F 62 61 6C 41 6C 6C 6F 63 00 00 00 6C 73 GlobalAlloc...ls
0040353E 74 72 6C 65 6E 00 00 00 43 6C 6F 73 65 48 61 6E tlen...CloseHan
0040354E 64 6C 65 00 00 00 57 72 69 74 65 46 69 6C 65 00 00 die...Writefile.
0040356E 65 41 00 00 00 00 52 65 61 64 46 69 6C 65 00 00 ..GetModuleHandl
0040357E 00 00 45 78 69 74 50 72 6F 63 65 73 78 00 00 00 ea....ReadFile..
0040358E 49 6E 69 74 43 6F 60 6D 6F 6E 43 6F 6E 74 72 6F ..ExitProcess...
0040359E 6C 73 00 00 00 00 43 72 65 61 74 65 54 6F 6F 6C InitCommonContro
004035AE 62 61 72 45 78 00 00 00 43 72 65 61 74 65 54 6F ls....CreateTool
004035BE 6F 6C 62 61 72 00 00 00 54 65 78 74 4F 75 74 41 barEx...CreateI
004035CE 00 00 00 00 53 74 61 72 74 50 61 67 65 00 00 00 oibar...TextOutA
004035DE 53 74 61 72 74 44 6F 63 41 00 00 00 47 65 74 54 ...StartPage...
004035EE 65 78 74 40 65 74 72 69 63 73 41 00 00 00 47 65 StartDocA...GetI
004035FE 74 53 74 6F 63 6B 4F 62 68 65 63 74 00 00 00 00 extMetricsA...Ge
0040360E 45 6E 64 50 61 67 65 00 00 00 45 6E 64 44 6F 63 tStockObject...
0040361E 00 00 00 00 44 65 6C 65 74 65 4F 62 6A 65 63 74 EndPage...EndDoc
0040362E 00 00 00 00 44 65 6C 65 74 65 4F 43 00 00 00 00 00 ....DeleteObject
0040363E 47 65 74 53 61 76 65 46 69 6C 65 4E 61 60 65 41 ....DeleteDC...
0040364E 00 00 00 00 47 65 74 4F 70 65 6E 46 69 6C 65 4E GetSaveFileNameA
0040365E 61 60 65 41 00 00 00 00 50 72 69 6E 74 44 6C 67 ....GetOpenFileNameA
0040366E 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 PrintDig
0040367E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A.....
0040368E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040369E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004036AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

So, when the system runs, if we see the below image, it works on those little boxes.

VIRSET(n)	U0	U1	U2	U3	U4	U5	U6	U7	U8	U9	UA	UB	UC	UD	UE	UF	
00000F60	0C	36	00	00	16	36	00	00	20	36	00	00	30	36	00	00	.6...6.. 6..06..
00000F70	00	00	00	00	3C	36	00	00	50	36	00	00	64	36	00	00<6..P6..d6..
00000F80	00	00	00	00	CC	32	00	00	D8	32	00	00	EC	32	00	00í2..02..i2..
00000F90	FA	32	00	00	0E	33	00	00	1C	33	00	00	2C	33	00	00	ú2...3...3...3..
00000FA0	3A	33	00	00	46	33	00	00	54	33	00	00	60	33	00	00	:3..F3..T3..`3..
00000FB0	6E	33	00	00	80	33	00	00	8C	33	00	00	9E	33	00	00	n3..€3..€3..ž3..
00000FC0	B6	33	00	00	C4	33	00	00	D8	33	00	00	E4	33	00	00	¶3..Ä3..Ø3..ä3..
00000FD0	F2	33	00	00	02	34	00	00	0E	34	00	00	1E	34	00	00	ð3...4...4...4..
00000FE0	2E	34	00	00	40	34	00	00	4E	34	00	00	60	34	00	00	.4..@4..N4..`4..
00000FF0	72	34	00	00	84	34	00	00	98	34	00	00	A6	34	00	00	r4...4..~4..!4..
00001000	B2	34	00	00	BE	34	00	00	CC	34	00	00	D4	34	00	00	*4..¾4..í4..Ø4..
00001010	E2	34	00	00	F4	34	00	00	00	00	00	00	02	35	00	00	å4..ö4.....5..
00001020	12	35	00	00	1E	35	00	00	2C	35	00	00	3A	35	00	00	.5...5...5...:5..
00001030	44	35	00	00	52	35	00	00	5E	35	00	00	72	35	00	00	D5..R5..ñ5..r5..
00001040	7E	35	00	00	00	00	00	00	8C	35	00	00	A2	35	00	00	~5.....€5..+5..
00001050	B4	35	00	00	00	00	00	00	C4	35	00	00	D0	35	00	00	'5.....Ä5..B5..
00001060	DC	35	00	00	E8	35	00	00	FA	35	00	00	OC	36	00	00	Ü5..è5..ú5...6..
00001070	16	36	00	00	20	36	00	00	30	36	00	00	00	00	00	00	.6... 6..06.....
00001080	3C	36	00	00	50	36	00	00	64	36	00	00	00	00	00	00	<6..P6..d6.....
00001090	55	53	45	52	33	32	2E	64	6C	6C	00	4B	45	52	4E	45	USER32.dll.KERNE

And it add each other the ImageBase content and finds the correspondent function string. And from that string, it solves the address in our machine in runtime. As in this case, it finds the GetModuleHandleA address and modifies **5E 35** with the API address.

That's why an executable works on any machine because it will always find the name of the correspondent API and solve it to run. So, the address that it always has will be valid independently of the machine. But the address of the GetModuleHandleA little box will be the same in all machines and executables without randomization. Just the content will change.

That's why I use:

CALL [0x403238]

It will always work because 0x403238 is the IAT address for GetModuleHandleA, what changes is the content that the system will save modifying the 5E 35 original value that pointed to the string if we add it the base.

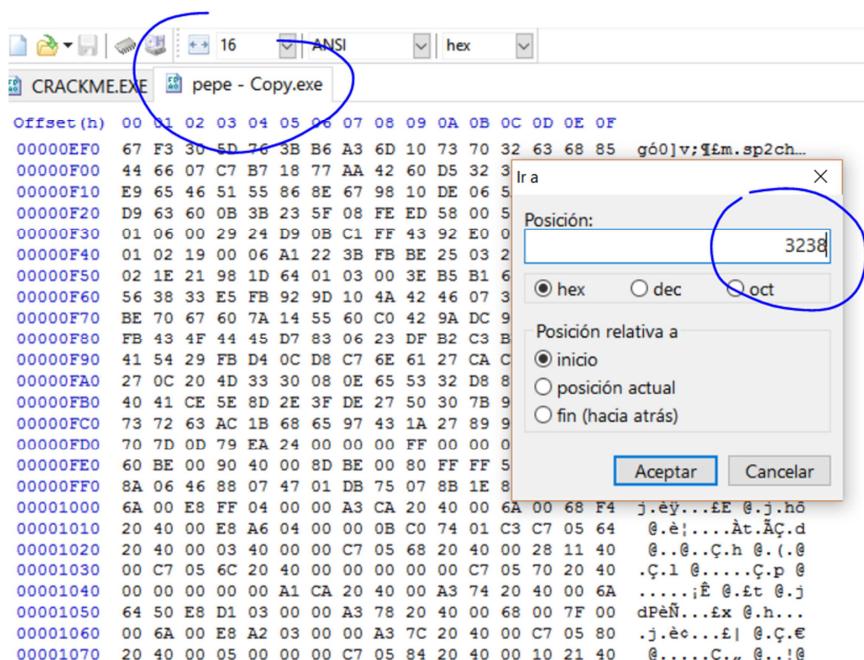
Without closing the two IDAs and with the packed and original files at the OEP, in a third IDA, load the Dumped we have just done: **pepe - Copy.exe** and in the dump, go to 0x403238 in this.

•	UPX0:00403228	dword_403228	dd 74E8CCF0h	; DATA XREF: UPX0:004014EE1r
•	UPX0:0040322C	dword_40322C	dd 74E8C4A0h	; DATA XREF: UPX0:004014F41r
•	UPX0:00403230	dword_403230	dd 74E99660h	; DATA XREF: UPX0:004014FA1r
•	UPX0:00403234	dword_403234	dd 74E99D30h	; DATA XREF: UPX0:004015001r
•	UPX0:00403238	dword_403238	dd 74E8CD90h	; DATA XREF: sub_4015061r
•	UPX0:0040323C	dword_40323C	dd 74E9C40h	; DATA XREF: UPX0:0040150C1r
•	UPX0:00403240	dword_403240	dd 74E9ADB0h	; DATA XREF: sub_4015121r
•	UPX0:00403244			
•	UPX0:00403248	dword_403248	dd 7231FC30h	; DATA XREF: UPX0:004015181r
•	UPX0:0040324C	dword_40324C	dd 72337340h	; DATA XREF: UPX0:0040151E1r
•	UPX0:00403250	dword_403250	dd 72337300h	; DATA XREF: UPX0:004015241r
•	UPX0:00403254			
•	UPX0:00403258	dword_403258	dd 771CFFA0h	; DATA XREF: UPX0:0040152A1r
•	UPX0:0040325C	dword_40325C	dd 771C6990h	; DATA XREF: UPX0:004015301r
•	UPX0:00403260	dword_403260	dd 771CFF30h	; DATA XREF: UPX0:004015361r
•	UPX0:00403264	dword_403264	dd 771C59D0h	; DATA XREF: UPX0:0040153C1r
•	UPX0:00403268	dword_403268	dd 771C4CA0h	; DATA XREF: UPX0:004015421r
•	UPX0:0040326C	dword_40326C	dd 771C69C0h	; DATA XREF: UPX0:004015481r
•	UPX0:00403270	dword_403270	dd 771C6E40h	; DATA XREF: UPX0:0040154E1r
•	UPX0:00403274	dword_403274	dd 771C3940h	; DATA XREF: UPX0:004015541r
•	UPX0:00403278	dword_403278	dd 771C3ED0h	; DATA XREF: UPX0:0040155A1r
•	UPX0:0040327C			
•	UPX0:00403280	dword_403280	dd 7516A1B0h	; DATA XREF: UPX0:004015601r
•	UPX0:00403284	dword_403284	dd 7516A0C0h	; DATA XREF: UPX0:004015661r
•	UPX0:00403288	dword_403288	dd 75173E80h	; DATA XREF: UPX0:0040156C1r
•	UPX0:004032C8			
00003238 00403238: UPX0:dword_403238 (Synchronized with Hex View-1)				

been propagated

Here, file offset is 0x3238 and it doesn't match the original because of the DumpFixer that made RawSize and VirtualSize the same changing the offsets and the GMHA API address.

Let's open it in HXD and go to 0x3238.



There, we see.

ARCHIVO EDICIÓN BUSCAR VER ANÁLISIS EXTRAS VERIFICACIÓN :

CRACKME.EXE pepe - Copy.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00003140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003180	00 00 00 00 30 E0 85 75 00 A6 83 75 20 B4 85 750à...u.;fu'...u
00003190	80 3E 85 75 60 74 8C 75 F0 96 83 75 F0 41 85 75 €>...t@uð-fuð...u
000031A0	80 87 85 75 30 88 85 75 E0 E3 85 75 60 88 8B 75 €‡...u0^...uâ...u`^<u
000031B0	70 84 85 75 C0 69 8A 75 A0 DF 85 75 50 27 85 75 p...uâ...u ß...uP^...u
000031C0	B0 E0 85 75 40 91 84 75 20 5C 88 75 B0 E5 85 75 °à...u@...u \^u°â...u
000031D0	60 B8 84 75 D0 A0 84 75 60 E5 85 75 70 FE 84 75 `...uÐ...u \^...upb...u
000031E0	90 B5 85 75 40 D7 85 75 20 BC 85 75 C0 19 27 77 .p...u@x...u 4...uÀ.'w
000031F0	D0 6F 88 75 00 BC 85 75 C0 8E 85 75 E0 BE 84 75 Ðo^u...uâ...uâ...u
00003200	10 DA 85 75 D0 5F 85 75 00 DB 85 75 B0 BB 84 75 .Ú...uÐ...u.Ú...u^...u
00003210	00 D6 8C 75 80 24 85 75 00 00 00 00 00 CF E8 74 .Ö@uë...u....éítet
00003220	90 B8 EC 74 80 00 E9 74 F0 CC E8 74 A0 C4 E8 74 ,.ite.étoëiet Äét
00003230	60 96 E9 74 30 9D E9 74 B0 CD E8 74 40 9C E9 74 '-ét0.étjítet@ét
00003240	B0 AD E9 74 00 00 00 30 FC 31 72 40 73 33 72 °.ét....0uir@3r
00003250	00 73 33 72 00 00 00 A0 FF 1C 77 90 69 1C 77 .s3r.... ý.w.i.w
00003260	30 FF 1C 77 D0 59 1C 77 A0 4C 1C 77 C0 69 1C 77 0ý.wDÝ.w L.wÀi.w
00003270	40 6E 1C 77 40 39 1C 77 D0 3E 1C 77 00 00 00 00 @n.w@9.wD>.w....
00003280	B0 A1 16 75 C0 A0 16 75 80 3E 17 75 00 00 00 00 00 °i.uÀ...u€>.u....
00003290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000032A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000032B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000032C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

We have an API address there and not an offset to add it the ImageBase to find the API name.

When the system ran it solved the API address and saved it there and when dumping, the packed file GetModuleHandleA API address stays there.

nts, Program Segmentation, General registers, Hex View-1, S

Breakpoints Program

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
00003140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003180	00 00 00 00 30 E0 85 75 00 A6 83 75 20 B4
00003190	80 3E 85 75 60 74 8C 75 F0 96 83 75 F0 41
000031A0	80 87 85 75 30 88 85 75 E0 E3 85 75 60 88
000031B0	70 84 85 75 C0 69 8A 75 A0 DF 85 75 50 27
000031C0	B0 E0 85 75 40 91 84 75 20 5C 88 75 B0 E5
000031D0	60 B8 84 75 D0 A0 84 75 60 E5 85 75 70 FE
000031E0	90 B5 85 75 40 D7 85 75 20 BC 85 75 C0 19
000031F0	D0 6F 88 75 00 BC 85 75 C0 8E 85 75 E0 BE
00003200	10 DA 85 75 D0 5F 85 75 00 DB 85 75 B0 BB
00003210	00 D6 8C 75 80 24 85 75 00 00 00 00 80 CF
00003220	90 B8 EC 74 80 00 E9 74 F0 CC E8 74 A0 C4
00003230	60 96 E9 74 30 9D E9 74 B0 CD E8 74 40 9C
00003240	B0 AD E9 74 00 00 00 00 30 FC 31 72 40 73
00003250	00 73 33 72 00 00 00 00 A0 FF 1C 77 90 69
00003260	30 FF 1C 77 D0 59 1C 77 A0 4C 1C 77 C0 69

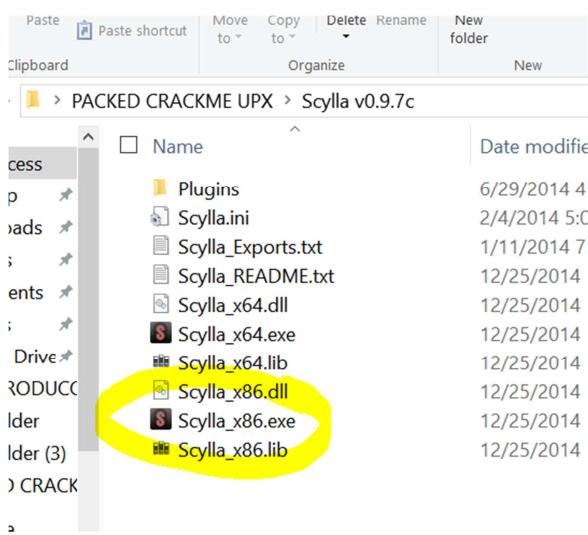
And what's the problem?

When the system runs, it looks for the value in the API address to add it to the ImageBase and find a string to solve it, but it is broken because when dumping, the final address was saved and the program will crash not being able to fill the IAT correctly.

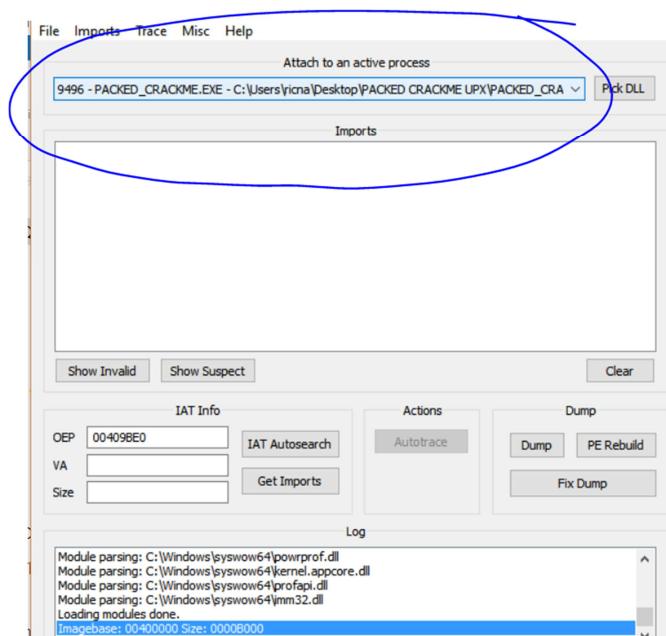
The idea is to fix the IAT and restore all those offsets, which pointed to the strings, with the API names. To do this, we need a tool called Scylla.

<https://tuts4you.tuts4you.com/download.php?view.3503>

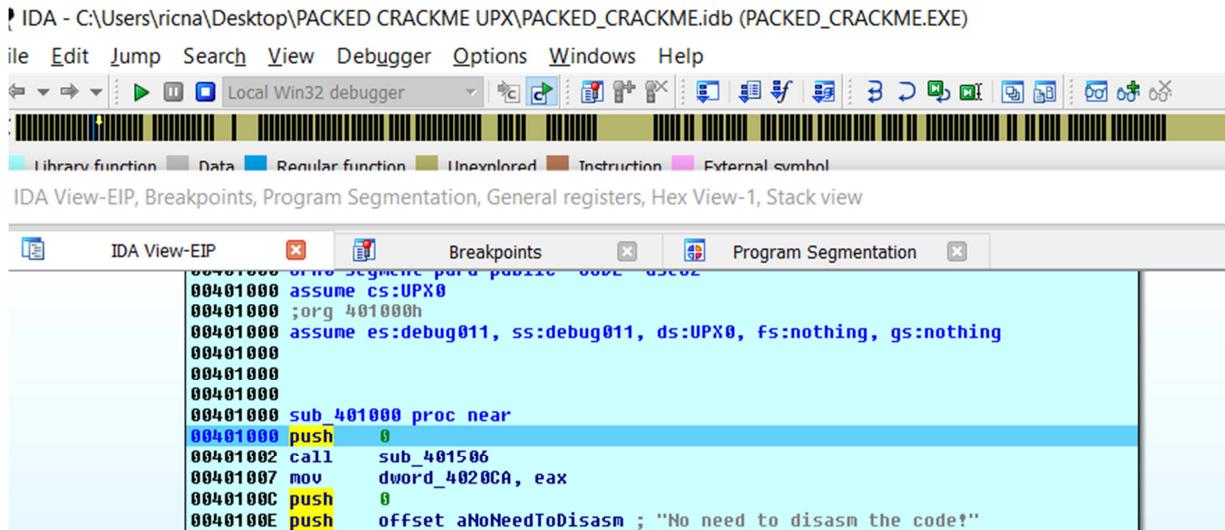
rar password = **tuts4you**



Run it.



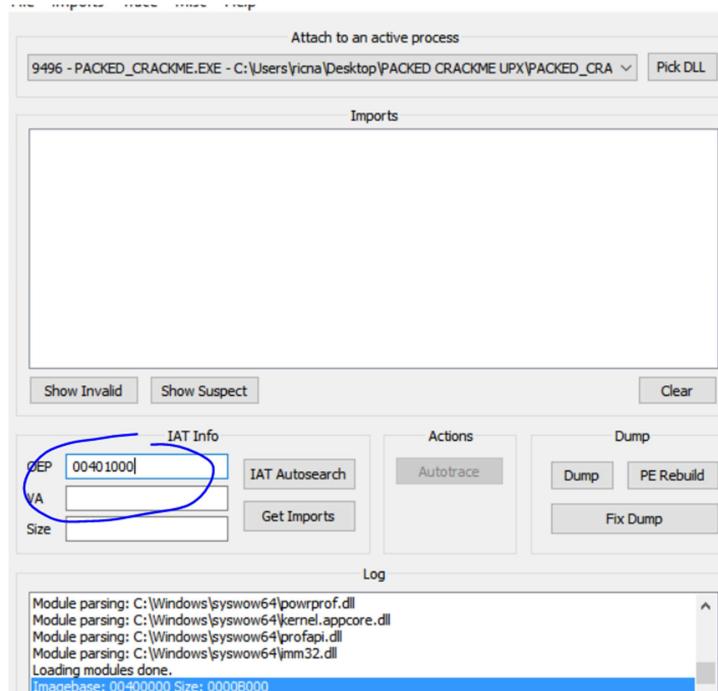
In ATTACH TO AN ACTIVE PROCESS, select the packed file process that is at the OEP in IDA yet.



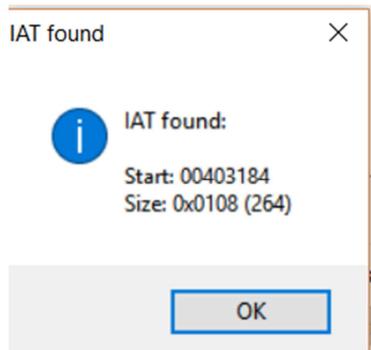
The screenshot shows the IDA Pro interface with the title bar "IDA - C:\Users\ricna\Desktop\PACKED CRACKME UPX\PACKED_CRACKME.idb (PACKED_CRACKME.EXE)". The menu bar includes File, Edit, Jump, Search, View, Debugger, Options, Windows, and Help. Below the menu is a toolbar with various icons. The status bar at the bottom displays "IDA View-EIP, Breakpoints, Program Segmentation, General registers, Hex View-1, Stack view". The main window shows assembly code starting at address 00401000:

```
00401000 assume cs:UPX0
00401000 ;org 401000h
00401000 assume es:debug011, ss:debug011, ds:UPX0, fs:nothing, gs:nothing
00401000
00401000
00401000 sub_401000 proc near
00401000 push 0
00401002 call sub_401506
00401007 mov dword_4020CA, eax
0040100C push 0
0040100E push offset aNoNeedToDisasm ; "No need to disasm the code!"
```

Change the OEP to 401000.



Press IAT AUTOSEARCH.



There, it says that the IAT starts at 00x403184 and its size is 0x108.

Let's press GET IMPORTS.

A screenshot of the Immunity Debugger's 'Imports' window. The window title is 'Imports'. It lists imports from several DLLs: user32.dll (37 FThunks), kernel32.dll (10 FThunks), gdi32.dll (9 FThunks), and comdlg32.dll (3 FThunks). Under kernel32.dll, three entries are marked as invalid (red X) and highlighted with a blue border: rva: 00003248 ptr: 7231FC30, rva: 0000324C ptr: 72337340, and rva: 00003250 ptr: 72337300. At the bottom of the window are two buttons: 'Show Invalid' and 'Show Suspect'.

It solved everything except 3 APIs shown when pressing SHOW INVALID.

Imports					
kernel32.dll (10) FThunk: 0000321C					
✓	rva: 0000321C	mod: kernel32.dll	ord: 025D	name: GetLocalTime	
✓	rva: 00003220	mod: kernel32.dll	ord: 03F5	name: OpenFile	
✓	rva: 00003224	mod: kernel32.dll	ord: 032D	name: GlobalFree	
✓	rva: 00003228	mod: kernel32.dll	ord: 0326	name: GlobalAlloc	
⚠	rva: 0000322C	mod: kernel32.dll	ord: 0627	name: lstrlenA	
✓	rva: 00003230	mod: kernel32.dll	ord: 0087	name: CloseHandle	
✓	rva: 00003234	mod: kernel32.dll	ord: 05FF	name: WriteFile	
✓	rva: 00003238	mod: kernel32.dll	ord: 0270	name: GetModuleHandleA	
✓	rva: 0000323C	mod: kernel32.dll	ord: 0465	name: ReadFile	
✓	rva: 00003240	mod: kernel32.dll	ord: 015C	name: ExitProcess	
✗	?	(3) FThunk	00003248		
Show Invalid		Show Suspect			

In the packed file, we see that the 3238 offset belonged to GetModuleHandleA and that is well solved. Let's see, in the packed file, the invalid API addresses from 0x403248 on to see what they are.

```
UPX0:00403240 off_403240 dd offset kernel32_ExitProcess ; DATA XREF: sub_401512tr
UPX0:00403244 align 8
UPX0:00403248 off_403248 dd offset comctl32_InitCommonControls ; DATA XREF: UPX0:00401518tr
UPX0:0040324C off_40324C dd offset comctl32_CreateToolBarEx ; DATA XREF: UPX0:0040151Etr
UPX0:00403250 off_403250 dd offset comctl32_CreateToolbar ; DATA XREF: UPX0:00401524tr
UPX0:00403254 align 8
UPX0:00403258 off_403258 dd offset gdi32_TextOutA ; DATA XREF: UPX0:0040152Atr
```

Those APIs belong to comctl32. If I right click on the invalid APIs in Scylla and select PLUGIN-PE COMPACT, it fixes them well.

Imports	
[+]	[+] user32.dll (37) FThunk: 00003184
[+]	[+] kernel32.dll (10) FThunk: 0000321C
[+]	[+] comctl32.dll (3) FThunk: 00003248
	[+] rva: 00003248 mod: comctl32.dll ord: 0011 name: InitCommonControls
	[+] rva: 0000324C mod: comctl32.dll ord: 0016 name: CreateToolBarEx
	[+] rva: 00003250 mod: comctl32.dll ord: 0007 name: CreateToolbar
[+]	[+] gdi32.dll (9) FThunk: 00003258
[+]	[+] comdlg32.dll (3) FThunk: 00003280

They are the API's we need.

Imports

- gdi32.dll (9) FThunk: 00003258
 - rva: 00003258 mod: gdi32.dll ord: 09E1 name: TextOutA
 - rva: 0000325C mod: gdi32.dll ord: 09D8 name: StartPage
 - rva: 00003260 mod: gdi32.dll ord: 09D4 name: StartDocA
 - rva: 00003264 mod: gdi32.dll ord: 06B6 name: GetTextMetricsA
 - rva: 00003268 mod: gdi32.dll ord: 069A name: GetStockObject
 - rva: 0000326C mod: gdi32.dll ord: 0560 name: EndPage
 - rva: 00003270 mod: gdi32.dll ord: 055C name: EndDoc
 - rva: 00003274 mod: gdi32.dll ord: 054F name: DeleteObject
 - rva: 00003278 mod: gdi32.dll ord: 054B name: DeleteDC
- comdlg32.dll (3) FThunk: 00003280

[Show Invalid](#) [Show Suspect](#)

If we press SHOW SUSPECT, we see if those two suspects are correct going to 0x403258 and 0x403278.

```

UPX0:00403240 off_403240 dd offset kernel32_ExitProcess ; DATA XREF: sub_401512tr
UPX0:00403244 align 8
UPX0:00403248 off_403248 dd offset comctl32_InitCommonControls ; DATA XREF: UPX0:00401518tr
UPX0:0040324C off_40324C dd offset comctl32_CreateToolbarEx ; DATA XREF: UPX0:0040151Etr
UPX0:00403250 off_403250 dd offset comctl32_CreateToolbar ; DATA XREF: UPX0:00401524tr
UPX0:00403254 align 8
UPX0:00403258 off_403258 dd offset gdi32_TextOutA ; DATA XREF: UPX0:0040152Atr
UPX0:0040325C off_40325C dd offset gdi32_StartPage ; DATA XREF: UPX0:00401530tr
UPX0:00403260 off_403260 dd offset gdi32_StartDocA ; DATA XREF: UPX0:00401536tr
UPX0:00403264 off_403264 dd offset gdi32_GetTextMetricsA ; DATA XREF: UPX0:0040153Ctr
UPX0:00403268 off_403268 dd offset gdi32_GetStockObject ; DATA XREF: UPX0:00401542tr
UPX0:0040326C off_40326C dd offset gdi32_EndPage ; DATA XREF: UPX0:00401548tr
UPX0:00403270 off_403270 dd offset gdi32_EndDoc ; DATA XREF: UPX0:0040154Etr
UPX0:00403274 off_403274 dd offset gdi32_DeleteObject ; DATA XREF: UPX0:00401554tr
UPX0:00403278 off_403278 dd offset gdi32_DeleteDC ; DATA XREF: UPX0:0040155Atr
UPX0:0040327C align 10h
UPX0:00403280 off_403280 dd offset comdlg32_GetSaveFileNameA ; DATA XREF: UPX0:00401560tr

```

Yes, they are. Now, press FIX DUMP.

9496 - PACKED_CRACKME.EXE - C:\Users\icna\Desktop\PACKED CRACKME UPX\PACKED_CRA... Pick DLL

Imports

- user32.dll (37) FThunk: 00003184
- kernel32.dll (10) FThunk: 0000321C
- comctl32.dll (3) FThunk: 00003248
- gdi32.dll (9) FThunk: 00003258
- comdlg32.dll (3) FThunk: 00003280

[Show Invalid](#) [Show Suspect](#)

IAT Info	Actions	Dump	PE Ref
OEP: 00401000	IAT Autosearch	Dump	
VA: 00403184	Get Imports		
Size: 00000108		Dump	

Log

Fix Dump

Open

... < Des... > PACKED CRACKME UPX\PACKED_CRA...

Organize ▾ New folder

Name

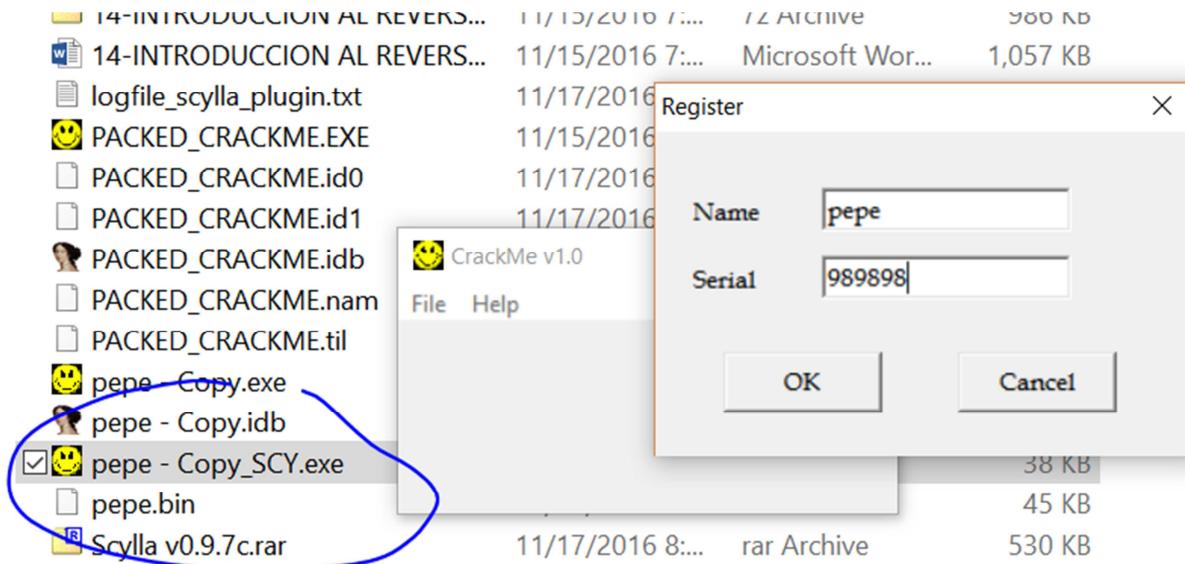
Scylla v0.9.7c

PACKED_CRACKME UPX\PACKED_CRA...

pepe - Copy.exe

File name: pepe - Copy.exe

It saves it as pepe-Copy_SCY.exe and it works.



When loading the unpacked file in IDA, we see that it runs from the 0x401000 OEP and the API names look like the original file and they are well fixed.

The screenshot shows the IDA Pro debugger interface with the assembly view open. The assembly code for the 'start' function is as follows:

```
00401000 ; Segment type: Pure code
00401000 ; Segment permissions: Read/Write/Execute
00401000 UPX0 segment para public 'CODE' use32
00401000 assume cs:UPX0
00401000 ;org 401000h
00401000 assume es:nothing, ss:nothing, ds:UPX0, fs:nothing, gs:nothing
00401000
00401000
00401000
00401000 public start
00401000 start proc near
00401000 push 0 ; lpModuleName
00401002 call GetModuleHandleA
00401007 mov hInstance, eax
0040100C push 0 ; lpWindowName
0040100E push offset ClassName ; "No need to disasm the code!"
00401013 call FindWindowA
00401018 or eax, eax
0040101A jz short loc_40101D
0040101C retn
```

The assembly code is annotated with several blue circles highlighting specific instructions: 'push 0 ; lpModuleName', 'call GetModuleHandleA', 'push offset ClassName ; "No need to disasm the code!"', and 'call FindWindowA'. A red arrow points from the assembly code to the corresponding hex dump below.

The hex dump at the bottom of the assembly view shows the following:

```
0040101D
0040101D loc_40101D:
0040101D mov WndClass.style, 4003h
```

The status bar at the bottom indicates: 100.00% (-133, 439) (939, 129) 00000400 00401000: start (Synchronized with Hex View-1)

```
.idata:00403230 ; BOOL __stdcall CloseHandle(HANDLE hObject)
idata:00403230     extrn CloseHandle:dword ; DATA XREF: UPX0:004014FA†r
idata:00403234 ; BOOL __stdcall WriteFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumber
idata:00403234     extrn WriteFile:dword ; DATA XREF: UPX0:00401500†r
idata:00403238 ; HMODULE __stdcall GetModuleHandleA(LPCSTR lpModuleName)
idata:00403238     | extrn __imp_GetModuleHandleA:dword
idata:00403238     ; DATA XREF: GetModuleHandleA†r
idata:0040323C ; BOOL __stdcall ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumber
idata:0040323C     extrn ReadFile:dword ; DATA XREF: UPX0:0040150C†r
idata:00403240 ; void __stdcall _noretturn ExitProcess(UINT uExitCode)
idata:00403240     extrn __imp_ExitProcess:dword ; DATA XREF: ExitProcess1
idata:00403244
idata:00403248 ; Imports from comct132.dll
idata:00403248
idata:00403248 ; void __stdcall InitCommonControls()
idata:00403248     extrn InitCommonControls:dword ; DATA XREF: UPX0:004015
idata:00403248     ; .SCY:0040B1441o
idata:0040324C ; HWND __stdcall CreateToolbarEx(HWND hwnd, DWORD ws, UINT wID, int nBi
idata:0040324C     extrn CreateToolbarEx:dword ; DATA XREF: UPX0:0040151E†1
idata:00403250     extrn CreateToolbar:dword ; DATA XREF: UPX0:00401524†r
idata:00403254
idata:00403258 ; Imports from gdi32.dll
00002638 00403238: .idata:_imp_GetModuleHandleA (Synchronized with Hex View-)
```

The IAT part looks original too.

We have just unpacked our first and easy packer. We'll see more complex packers later.

Ricardo Narvaja

Translated by: @IvinsonCLS