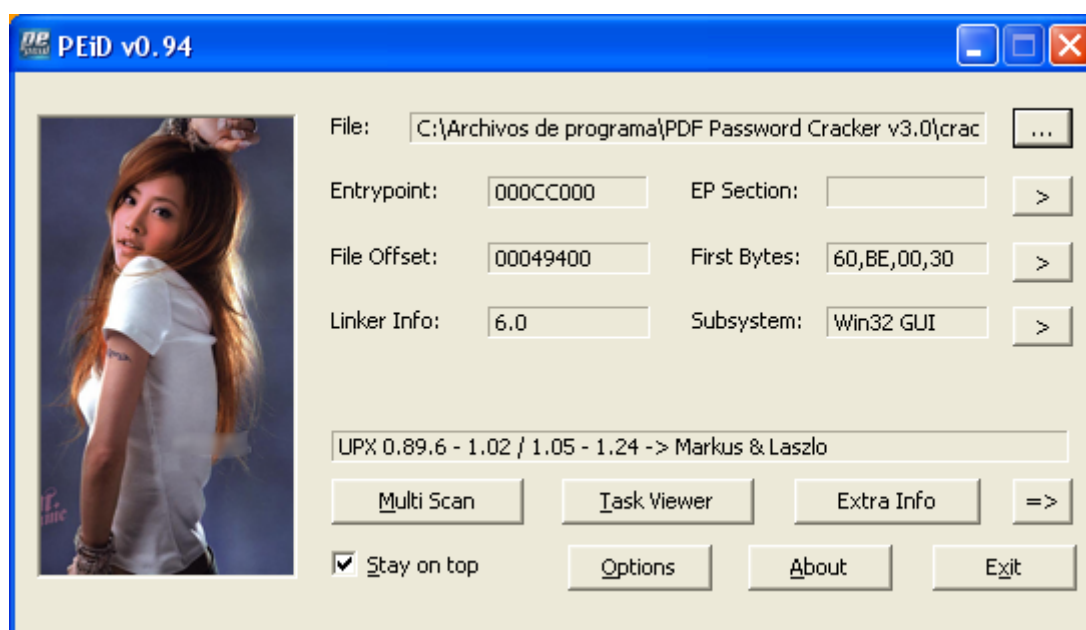
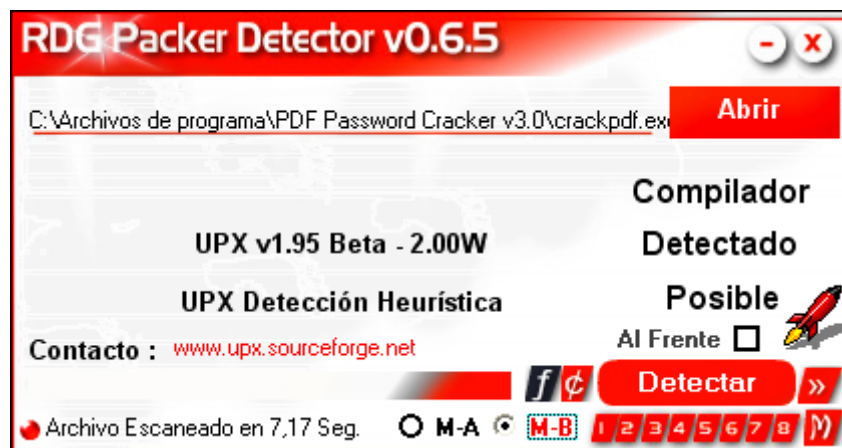
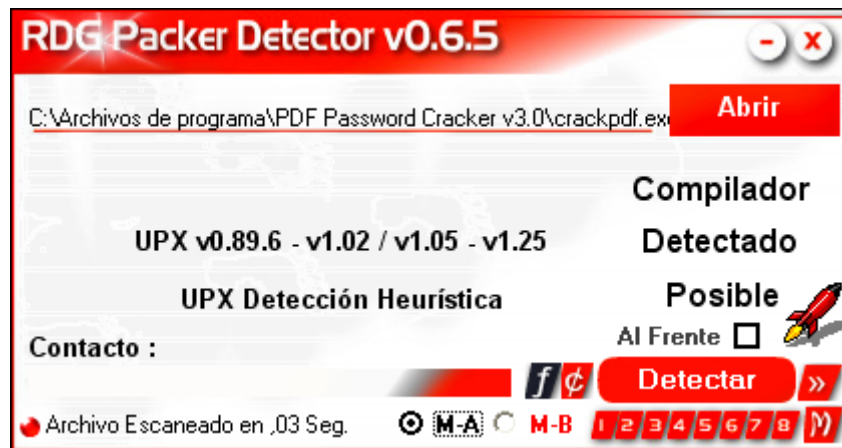


Autor	Aguml
Fecha	01-11-2008
Victima	PDF Password Cracker v3.0
Protección	UPX + Redirección IAT + Serial
Herramientas	Ollydbg + Import Rec
Objetivo	Desempacar y registrar
Dificultad	Muy Sencillo

Lo primero que hago es mirar con que está protegido:





Vemos que ambos coinciden aunque en el modo M-B el RDG me dice que es otra versión pero parece que está claro que es UPX.

Abrimos el programa en olly y lo arrancamos con F9 y nos saldrá una ventana para registrarnos:



Metamos un serial cualquiera y probemos a ver si acertamos:



Lo normal, si hubiera acertado sería prácticamente milagroso jajaja. Bueno, por lo menos nos muestra una ventanita de chico malo así que atacaré por ahí. Así que vayamos hasta el OEP y dumpeemos y reparemos la IAT para que corra.

Lo primero será llegar al OEP así que reiniciamos la aplicación en olly y aparecemos aquí:

Address	Hex dump	Disassembly
004CC000	60	pushad
004CC001	BE 00304800	mov esi, 483000
004CC006	8DBE 00E0F7FF	lea edi, ds:[esi+FFF7E000]
004CC00C	57	push edi
004CC00D	83CD FF	or ebp, FFFFFFFF
004CC010	EB 10	jmp short 004CC022
004CC012	90	nop
004CC013	90	nop
004CC014	90	nop
004CC015	90	nop
004CC016	90	nop
004CC017	90	nop

Pulsamos F7 para pasar el PUSHAD y ahora nos vamos a la Pila para ver donde introdujo el último valor de registro:

Address	Value	Comment
0013FFA4	7C920208	ntdll.7C920208
0013FFA8	FFFFFFFF	
0013FFAC	0013FFF0	
0013FFB0	0013FFC4	
0013FFB4	7FFDF000	
0013FFB8	7C91E4F4	ntdll.KiFastSystemCallRet
0013FFBC	004CC000	crackpdf.<ModuleEntryPoint>
0013FFC0	00400000	crackpdf.00400000
0013FFC4	7C817067	RETURN to kernel32.7C817067
0013FFC8	7C920208	ntdll.7C920208
0013FFCC	FFFFFFFF	
0013FFD0	7FFDF000	
0013FFD4	80544C7D	
0013FFD8	0013FFC8	
0013FFDC	864E3D10	
0013FFE0	FFFFFFFF	End of SEH chain
0013FFE4	7C839AC0	SE handler
0013FFE8	7C817070	kernel32.7C817070
0013FFEC	00000000	
0013FFF0	00000000	
0013FFF4	00000000	
0013FFF8	0014515A	
0013FFFC	00000000	

Vemos que el último que añadió esta en 0013FFA4 así que vayamos a esa dirección en el Dump y pongamos un HBP en él:

Address	Hex dump	Comment
004CC000	60	
004CC001	BE 003048	
004CC006	8DBE 00E0	
004CC00C	57	
004CC00D	83CD FF	
004CC010	EB 10	
004CC012	90	
004CC013	90	
004CC014	90	
004CC015	90	
004CC016	90	
004CC017	90	
004CC018	8A06	
004CC01A	46	
004CC01B	8807	

Address	Hex dump	Comment
0013FFA4	08 02 92 7C	
0013FFB4	00 F0 FD 7F	
0013FFC4	67 70 81 7C	
0013FFD4	7D 4C 54 80	
0013FFE4	C0 9A 83 7C	
0013FFF4	00 00 00 00	

Ahora damos a F9 y vemos que para en:

Address	Hex dump	Disassembly
004CC1A7	8D4424 80	lea eax, ss:[esp-80]
004CC1AB	6A 00	push 0
004CC1AD	39C4	cmp esp, eax
004CC1AF	75 FA	jnz short 004CC1AB
004CC1B1	83EC 80	sub esp, -80
004CC1B4	E9 5494F9FF	jmp 0046560D
004CC1B9	0000	add ds:[eax], al
004CC1BB	0000	add ds:[eax], al
004CC1BD	0000	add ds:[eax], al
004CC1BF	0000	add ds:[eax], al
004CC1C1	0000	add ds:[eax], al
004CC1C3	0000	add ds:[eax], al

Pues pongamos un BP en el JMP como se ve en la imagen y demos a F9 y cuando pare en el BP damos a F7 para pasarlo y llegamos aquí:

Address	Hex dump	Disassembly	Comment
0046560D	55	push ebp	
0046560E	8BEC	mov ebp, esp	
00465610	6A FF	push -1	
00465612	68 E8BC4800	push 48BCE8	
00465617	68 4C9D4600	push 469D4C	
0046561C	64:A1 00000000	mov eax, fs:[0]	SE handler installation
00465622	50	push eax	
00465623	64:8925 00000000	mov fs:[0], esp	
0046562A	83EC 58	sub esp, 58	
0046562D	53	push ebx	
0046562E	56	push esi	
0046562F	57	push edi	
00465630	8965 E8	mov ss:[ebp-18], esp	
00465633	FF15 38824800	call ds:[488238]	kernel32.GetVersion
00465639	33D2	xor edx, edx	
0046563B	8AD4	mov dl, ah	

Esto tiene toda la pinta de ser un OEP correcto así que lo doy por bueno y seguimos. Ahora lo siguiente es dumpearlo y para eso usamos por ejemplo OllyDump.

OllyDump - crackpdf.exe

Start Address: 400000 Size: CF000 Dump

Entry Point: CC000 -> Modify: 6560D Get EIP as OEP Cancel

Base of Code: 83000 Base of Data: CD000

☒ Fix Raw Size & Offset of Dump Image

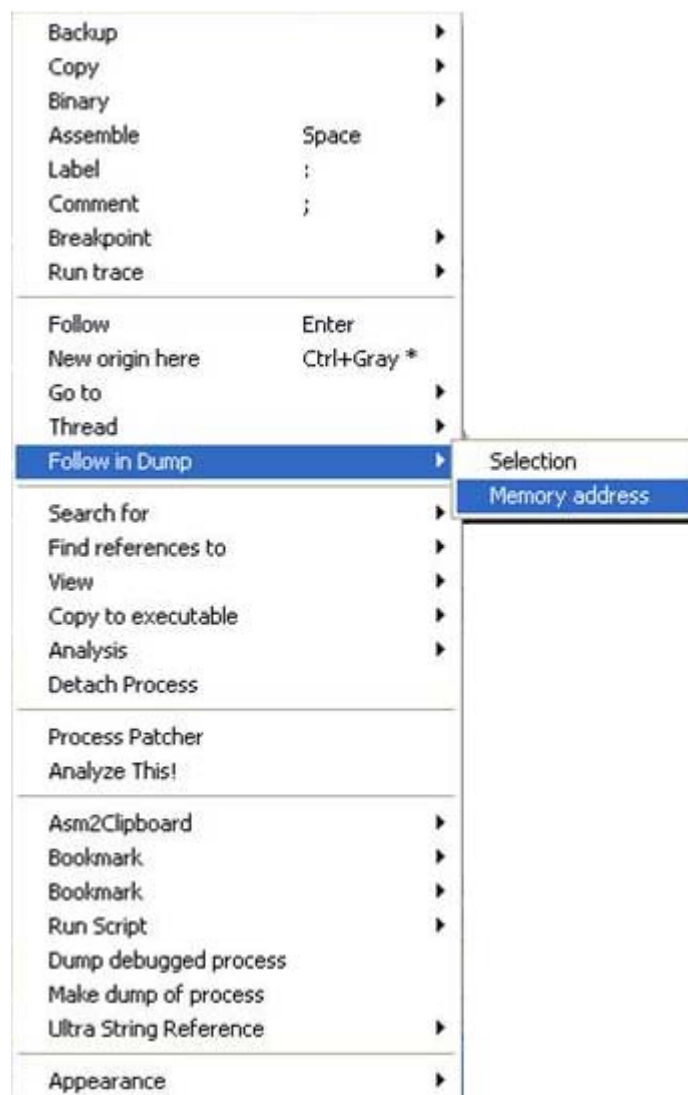
Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.rsrc	00082000	00001000	00082000	00001000	E0000080
	0004A000	00083000	0004A000	00083000	E0000040
	00002000	000CD000	00002000	000CD000	C0000040

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Ahora toca reparar la IAT y para ello necesitamos saber donde empieza y donde acaba esta. Para localizar la IAT nos colocamos encima de la siguiente CALL, ya que vemos que apunta a una API, y hacemos lo siguiente:



Aparecemos aquí:

Address	Hex dump
00488238	6A 12 81 7C B8 97 80 7C 93 C1 85 7C 31 BB 80 7C
00488248	C1 60 83 7C DB 60 83 7C AB 0B 83 7C 31 B7 80 7C
00488258	67 EE 80 7C 69 38 81 7C CC 15 81 7C 8C 39 81 7C
00488268	16 50 83 7C 00 10 91 7C E0 10 91 7C 30 AE 80 7C
00488278	5A 13 92 7C 81 9F 80 7C 6E 2B 81 7C 27 D8 81 7C
00488288	6B 23 80 7C 17 08 86 7C 12 18 80 7C 30 25 80 7C
00488298	46 24 80 7C D7 9B 80 7C 7C AC 85 7C FA CA 81 7C
004882A8	8D 1B 82 7C 7A 4F 81 7C 6E 2B 83 7C 6C 5D 83 7C
004882B8	5F B5 80 7C 59 4D 83 7C 46 BE 80 7C AD 23 86 7C

Subimos hasta encontrar el principio de la IAT y lo encontramos en 00487FFC:

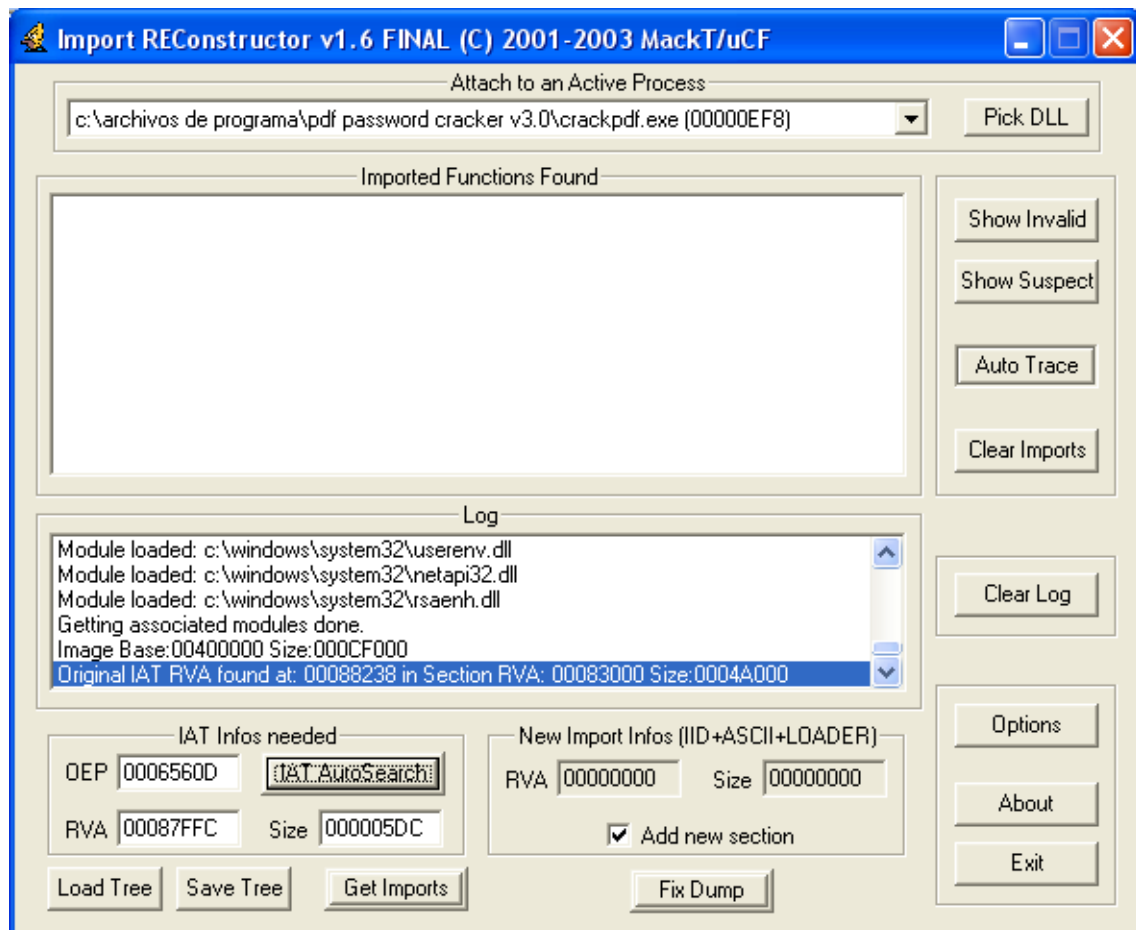
Address	Hex dump															
00487FE8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00487FF8	00	00	00	00	00	00	00	00	00	5D	BB	DC	77	42	78	DA 77
00488008	AB	7A	DA	77	E4	E9	DA	77	D7	EA	DA	77	D5	EC	DA	77
00488018	80	42	DB	77	17	6C	DA	77	00	00	00	00	F4	C7	C3	58
00488028	CF	65	C3	58	D8	03	C4	58	05	02	C4	58	00	00	00	00
00488038	AE	3A	F0	77	0E	D3	F1	77	05	3A	F0	77	EF	D3	F1	77
00488048	56	6A	EF	77	FA	6B	EF	77	06	C0	EF	77	F1	7C	EF	77
00488058	79	7C	EF	77	A5	61	EF	77	37	65	F2	77	1B	82	EF	77
00488068	3F	BA	EF	77	EA	D3	EF	77	7C	77	F0	77	9B	86	EF	77

Y ahora vamos para abajo hasta encontrar el final que lo vemos en 004885D8:

Address	Hex dump															
00488558	7A	97	3A	7E	9E	B2	3A	7E	89	C6	3B	7E	B2	DE	3A	7E
00488568	56	AF	3A	7E	00	00	00	00	40	4D	F8	72	5F	66	F9	72
00488578	57	37	F9	72	00	00	00	00	10	7C	37	76	9F	30	36	76
00488588	63	25	36	76	00	00	00	00	AC	00	50	77	EA	F6	4C	77
00488598	60	D0	4C	77	44	D0	4C	77	2A	56	51	77	5A	57	51	77
004885A8	1D	C8	5A	77	C5	56	4E	77	B6	FC	4F	77	F2	87	4E	77
004885B8	1F	5F	52	77	F3	A2	4F	77	71	AB	54	77	C1	A9	54	77
004885C8	E7	31	50	77	00	00	00	00	6A	09	1F	7E	00	00	00	00
004885D8	00	00	00	00	00	00	00	00	08	97	48	00	E8	85	48	00

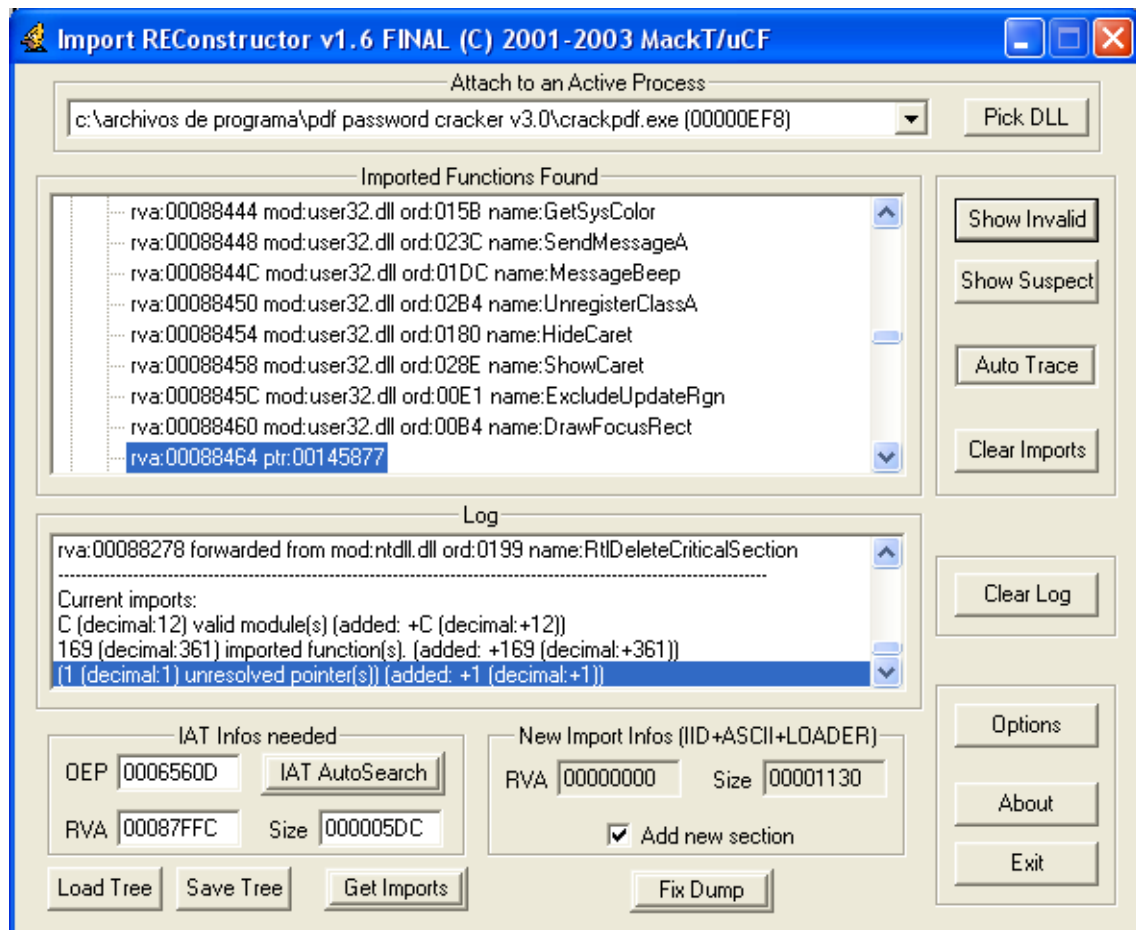
Final IAT – Inicio IAT = Tamaño IAT
Tamaño IAT = 004885D8 - 00487FFC = 5DC

Bueno pues con estos valores y el OEP ya podemos intentar reparar la IAT así que abrimos Import Rec y seleccionamos a la víctima en la lista de procesos:



Como veis ya tiene puesto los valores necesarios que son el OEP (menos la ImageBase, el RVA (es el inicio de la IAT menos la ImageBase), y el Size (es el tamaño de la IAT).

Ahora hacemos clic en Get Imports y a continuación en Show Invalid, ya que en el log me está avisando de que existe una entrada mala en la IAT:



En la imagen vemos que la entrada mala está en 00088464 así que sumamos la ImageBase y nos da 00488464. Pues vayamos a esa dirección en el Dump para poner un HBP:

Address	Hex dump	Appearance	ASCII
0046560D	55		
0046560E	8BEC		
00465610	6A FF		
00465612	68 E8BC48		
00465617	68 4C9D46		
0046561C	64:A1 000		
00465622	50		
00465623	64:8925 0		
0046562A	83EC 58		
0046562D	53		
0046562E	56		
0046562F	57		
00465630	8965 E8		
00465633	FF15 3882		
00465639	33D2		
0046563B	8AD4		
0046563D	8915 58E2		
00465643	8BC8		
00465645	81E1 FF00		
0046564B	890D 54E2		
00465651	C1E1 08		
ebp=0013FFFF			
00488464	77 58 14 00	B8 96 39 7E FF 97 3A 7E 9C 8F 39 7E	wX . , - 9 ~ y - : ~ æ □ 9 ~
00488474	6C D0 3B 7E	22 B2 3A 7E 0E 1B 3B 7E F6 E8 3A 7E	1D ; ~ " ^ : ~ □ □ ; ~ ö è : ~
00488484	49 98 3A 7E	3E D3 3A 7E 72 DE 39 7E B4 90 3A 7E	I ~ : ~ > 0 : ~ r P 9 ~ ^ □ : ~
00488494	0F 91 3A 7E	C5 77 3A 7E 28 8E 39 7E 2B 8D 39 7E	□ ` : ~ Å w : ~ (Ž 9 ~ + □ 9 ~

Damos a F9 y nos sale la ventana para registrarnos y no para así que tendremos que registrarlo para ver cuando usa esa dirección. Pues nada, reiniciamos la aplicación de nuevo y volvemos a llegar hasta el OEP verdadero. Lo primero que se me ocurre siempre es usar un BP MessageBoxA así que probemos a ponerlo desde la CommandBar y voy al lugar donde se puso y lo quito y lo coloco en el RETN de la API. Doy a F9 y nos sale la ventanita que nos dice que el serial ese no vale así que aceptamos y...

Address	Hex dump	Assembly	Comment
7E3D07ED	8BEC	mov ebp, esp	
7E3D07EF	83D BC143F7E	cmp dword ptr ds:[7E3F14BC], 0	
7E3D07F6	74 24	je short 7E3D081C	USER32.7E3D081C
7E3D07F8	64:A1 18000000	mov eax, fs:[18]	
7E3D07FE	6A 00	push 0	
7E3D0800	FF70 24	push dword ptr ds:[eax+24]	
7E3D0803	68 241B3F7E	push 7E3F1B24	
7E3D0808	FF15 C412397E	call ds:[7E3912C4]	kernel32.InterlockedCompare
7E3D080E	85C0	test eax, eax	
7E3D0810	75 0A	jnz short 7E3D081C	USER32.7E3D081C
7E3D0812	C705 201B3F7E	mov dword ptr ds:[7E3F1B20], 1	
7E3D081C	6A 00	push 0	
7E3D081E	FF75 14	push dword ptr ss:[ebp+14]	
7E3D0821	FF75 10	push dword ptr ss:[ebp+10]	
7E3D0824	FF75 0C	push dword ptr ss:[ebp+C]	
7E3D0827	FF75 08	push dword ptr ss:[ebp+8]	
7E3D082A	E8 2D000000	call 7E3D085C	USER32.MessageBoxExA
7E3D082F	5D	pop ebp	
7E3D0830	C2 1000	retn 10	
7E3D0833	90	nop	
7E3D0834	90	nop	

Vemos como ha parado donde queríamos así que demos a F7 para salir de la API y llegamos aquí:

00405260	B9 32000000	mov	ecx, 32	
00405265	33C0	xor	eax, eax	
00405267	BF F0B44B00	mov	edi, 4BB4F0	ASCII "98989898"
0040526C	68 F0B44B00	push	4BB4F0	ASCII "98989898"
00405271	68 FB030000	push	3FB	
00405276	56	push	esi	
00405277	F3:AB	rep	stos dword ptr es:[edi]	
00405279	FF15 F8834800	call	ds:[4883F8]	USER32.GetDlgItemTextA
0040527F	68 F0B44B00	push	4BB4F0	ASCII "98989898"
00405284	E8 77F9FFFF	call	00404C00	crackpdf.00404C00
00405289	83C4 04	add	esp, 4	
0040528C	85C0	test	eax, eax	
0040528E	74 44	je	short 004052D4	crackpdf.004052D4
00405290	6A 40	push	40	
00405292	68 246A4900	push	496A24	ASCII "Thank you."
00405297	68 EC694900	push	4969EC	ASCII "Thanks for purchasing"
0040529C	56	push	esi	
0040529D	FF15 00844800	call	ds:[488400]	USER32.MessageBoxA
004052A3	51	push	ecx	
004052A4	8BCC	mov	ecx, esp	
004052A6	896424 0C	mov	ss:[esp+C], esp	
004052AA	68 F0B44B00	push	4BB4F0	ASCII "98989898"
004052AF	E8 5A6C0700	call	0047BF0E	crackpdf.0047BF0E
004052B4	E8 F7FBFFFF	call	00404EB0	crackpdf.00404EB0
004052B9	83C4 04	add	esp, 4	
004052BC	C705 B8B54B00	mov	dword ptr ds:[4BB5B8], 1	
004052C6	6A 01	push	1	
004052C8	56	push	esi	
004052C9	FF15 FC834800	call	ds:[4883FC]	USER32.EndDialog
004052CF	E9 8F020000	jmp	00405563	crackpdf.00405563
004052D4	6A 10	push	10	
004052D6	6A 00	push	0	
004052D8	68 A4694900	push	4969A4	ASCII "Your registration key"
004052DD	56	push	esi	
004052DE	FF15 00844800	call	ds:[488400]	USER32.MessageBoxA
004052E4	68 FB030000	push	3FB	
004052E9	56	push	esi	
004052EA	FF15 04844800	call	ds:[488404]	USER32.GetDlgItem

Jojojo, vemos el MessageBox del chico malo y también el del chico bueno y justo encima del mensaje de chico bueno hay un salto condicional que si se cumple nos tira a la zona del chico malo y un poquito mas arriba hay una CALL que es la encargada de verificar el serial casi seguro así que pongamos un BP en esa CALL y demos a F9 para continuar y acto seguido intentamos registrarnos de nuevo para que pare en dicha CALL:

0040526C	68 F0B44B00	push	4BB4F0	ASCII "98989898"
00405271	68 FB030000	push	3FB	
00405276	56	push	esi	
00405277	F3:AB	rep	stos dword ptr es:[edi]	
00405279	FF15 F8834800	call	ds:[4883F8]	USER32.GetDlgItemTextA
0040527F	68 F0B44B00	push	4BB4F0	ASCII "98989898"
00405284	E8 77F9FFFF	call	00404C00	crackpdf.00404C00
00405289	83C4 04	add	esp, 4	
0040528C	85C0	test	eax, eax	
0040528E	74 44	je	short 004052D4	crackpdf.004052D4
00405290	6A 40	push	40	
00405292	68 246A4900	push	496A24	ASCII "Thank you."
00405297	68 EC694900	push	4969EC	ASCII "Thanks for purchasing"
0040529C	56	push	esi	

Una vez que pare en la CALL, damos a F7 y llegamos aquí:

Address	Hex dump	Disassembly
00404C00	83EC 18	sub esp, 18
00404C03	83C9 FF	or ecx, FFFFFFFF
00404C06	33C0	xor eax, eax
00404C08	53	push ebx
00404C09	56	push esi
00404C0A	8B7424 24	mov esi, ss:[esp+24]
00404C0E	57	push edi
00404C0F	8BFE	mov edi, esi
00404C11	F2:AE	repne scas byte ptr es:[edi]
00404C13	F7D1	not ecx
00404C15	49	dec ecx
00404C16	83F9 14	cmp ecx, 14
00404C19	74 07	je short 00404C22
00404C1B	5F	pop edi
00404C1C	5E	pop esi
00404C1D	5B	pop ebx
00404C1E	83C4 18	add esp, 18
00404C21	C3	retn
00404C22	8A46 0E	mov al, ds:[esi+E]
00404C25	8A4E 0F	mov cl, ds:[esi+F]
00404C28	8D5424 0C	lea edx, ss:[esp+C]
00404C2C	32DB	xor bl, bl
00404C2E	52	push edx

Si vamos traceando veremos que lo que hace es contar el largo de nuestro serial y justo antes del salto lo compara con 14 hexadecimal y como no es igual pues no saltara y nos sacara de la CALL y EAX valdrá 0 con lo cual nos llevará a la zona de chico malo así que tendremos que hacerlo saltar siempre:

00404C0F	8BFE	mov edi, esi
00404C11	F2:AE	repne scas byte ptr es:[edi]
00404C13	F7D1	not ecx
00404C15	49	dec ecx
00404C16	83F9 14	cmp ecx, 14
00404C19	74 07	je short 00404C22
00404C1B	5F	pop edi
00404C1C	5E	pop esi
00404C1D	5B	pop ebx
00404C1E	83C4 18	add esp, 18
00404C21	C3	retn
00404C22	8A46 0E	mov al, ds:[esi+E]
00404C25	8A4E 0F	mov cl, ds:[esi+F]
00404C28	8D5424 0C	lea edx, ss:[esp+C]
00404C2C	32DB	xor bl, bl

00404C0F	8BFE	mov	edi, esi
00404C11	F2:AE	repne	scas byte ptr es:[edi]
00404C13	F7D1	not	ecx
00404C15	49	dec	ecx
00404C16	83F9 14	cmp	ecx, 14
00404C19	EB 07	jmp	short 00404C22
00404C1B	5F	pop	edi
00404C1C	5E	pop	esi
00404C1D	5B	pop	ebx
00404C1E	83C4 18	add	esp, 18
00404C21	C3	retn	
00404C22	8A46 0E	mov	al, ds:[esi+E]
00404C25	8A4E 0F	mov	cl, ds:[esi+F]
00404C28	8D5424 0C	lea	edx, ss:[esp+C]
00404C2C	32DB	xor	bl, bl

Seguimos traceando y llegamos a otro salto y si nos fijamos si no se cumple el salto xoreara a EAX contra si misma así que la pondrá a 0 y eso no nos interesa así que tiene que saltar aquí también siempre:

00404C44	8BF8	mov	edi, eax
00404C46	8D4424 1C	lea	eax, ss:[esp+1C]
00404C4A	50	push	eax
00404C4B	E8 60090600	call	004655B0
00404C50	03F8	add	edi, eax
00404C52	83C4 08	add	esp, 8
00404C55	83FF 0B	cmp	edi, 0B
00404C58	74 09	je	short 00404C63
00404C5A	5F	pop	edi
00404C5B	5E	pop	esi
00404C5C	33C0	xor	eax, eax
00404C5E	5B	pop	ebx
00404C5F	83C4 18	add	esp, 18
00404C62	C3	retn	
00404C63	8A0E	mov	cl, ds:[esi]
00404C65	8A56 01	mov	dl, ds:[esi+1]
00404C68	8D4424 0C	lea	eax, ss:[esp+C]

00404C44	8BF8	mov	edi, eax
00404C46	8D4424 1C	lea	eax, ss:[esp+1C]
00404C4A	50	push	eax
00404C4B	E8 60090600	call	004655B0
00404C50	03F8	add	edi, eax
00404C52	83C4 08	add	esp, 8
00404C55	83FF 0B	cmp	edi, 0B
00404C58	EB 09	jmp	short 00404C63
00404C5A	5F	pop	edi
00404C5B	5E	pop	esi
00404C5C	33C0	xor	eax, eax
00404C5E	5B	pop	ebx
00404C5F	83C4 18	add	esp, 18
00404C62	C3	retn	
00404C63	8A0E	mov	cl, ds:[esi]
00404C65	8A56 01	mov	dl, ds:[esi+1]
00404C68	8D4424 0C	lea	eax, ss:[esp+C]

Seguimos traceando y vemos que la historia se repite de nuevo así que aquí también tendrá que saltar siempre:

00404C86	8BF8	mov	edi, eax
00404C88	51	push	ecx
00404C89	E8 22090600	call	004655B0
00404C8E	03F8	add	edi, eax
00404C90	83C4 08	add	esp, 8
00404C93	83FF 0A	cmp	edi, 0A
00404C96	74 09	je	short 00404CA1
00404C98	5F	pop	edi
00404C99	5E	pop	esi
00404C9A	33C0	xor	eax, eax
00404C9C	5B	pop	ebx
00404C9D	83C4 18	add	esp, 18
00404CA0	C3	retn	
00404CA1	807E 05 33	cmp	byte ptr ds:[esi+5], 33
00404CA5	74 09	je	short 00404CB0
00404CA7	5F	pop	edi
00404CA8	5E	pop	esi

00404C86	8BF8	mov	edi, eax
00404C88	51	push	ecx
00404C89	E8 22090600	call	004655B0
00404C8E	03F8	add	edi, eax
00404C90	83C4 08	add	esp, 8
00404C93	83FF 0A	cmp	edi, 0A
00404C96	EB 09	jmp	short 00404CA1
00404C98	5F	pop	edi
00404C99	5E	pop	esi
00404C9A	33C0	xor	eax, eax
00404C9C	5B	pop	ebx
00404C9D	83C4 18	add	esp, 18
00404CA0	C3	retn	
00404CA1	807E 05 33	cmp	byte ptr ds:[esi+5], 33
00404CA5	74 09	je	short 00404CB0
00404CA7	5F	pop	edi
00404CA8	5E	pop	esi

Justo después tenemos otro salto y lo mismo así que hay que tratarlo igual:

00404C93	83FF 0A	cmp	edi, 0A
00404C96	EB 09	jmp	short 00404CA1
00404C98	5F	pop	edi
00404C99	5E	pop	esi
00404C9A	33C0	xor	eax, eax
00404C9C	5B	pop	ebx
00404C9D	83C4 18	add	esp, 18
00404CA0	C3	retn	
00404CA1	807E 05 33	cmp	byte ptr ds:[esi+5], 33
00404CA5	EB 09	jmp	short 00404CB0
00404CA7	5F	pop	edi
00404CA8	5E	pop	esi
00404CA9	33C0	xor	eax, eax
00404CAB	5B	pop	ebx
00404CAC	83C4 18	add	esp, 18
00404CAF	C3	retn	
00404CB0	8A56 07	mov	dl, ds:[esi+7]

Seguimos traceando y llegamos a un SETE AL y poco después un RETN. Vemos que la condición no se cumple, con lo cual EAX seguirá siendo 0 así que sustituyamos el SETE por SETNE para hacer que se cumpla:

00404CD0	8D5424 1C	lea	edx, ss:[esp+1C]
00404CD4	8BF0	mov	esi, eax
00404CD6	52	push	edx
00404CD7	E8 D4080600	call	004655B0
00404CDC	83C4 08	add	esp, 8
00404CDF	03F0	add	esi, eax
00404CE1	33C0	xor	eax, eax
00404CE3	83FE 0F	cmp	esi, 0F
00404CE6	5F	pop	edi
00404CE7	5E	pop	esi
00404CE8	0F94C0	sete	al
00404CEB	5B	pop	ebx
00404CEC	83C4 18	add	esp, 18
00404CEF	C3	retn	
00404CF0	83EC 28	sub	esp, 28
00404CF3	53	push	ebx

00404CD0	8D5424 1C	lea	edx, ss:[esp+1C]
00404CD4	8BF0	mov	esi, eax
00404CD6	52	push	edx
00404CD7	E8 D4080600	call	004655B0
00404CDC	83C4 08	add	esp, 8
00404CDF	03F0	add	esi, eax
00404CE1	33C0	xor	eax, eax
00404CE3	83FE 0F	cmp	esi, 0F
00404CE6	5F	pop	edi
00404CE7	5E	pop	esi
00404CE8	0F95C0	setne	al
00404CEB	5B	pop	ebx
00404CEC	83C4 18	add	esp, 18
00404CEF	C3	retn	
00404CF0	83EC 28	sub	esp, 28
00404CF3	53	push	ebx

Seguimos traceando y salimos de la CALL y vemos como el salto no se cumple y nos aparece la ventana de chico bueno:



Ahora es el momento de ver cuando para en la entrada mala de la IAT así que, como tenemos un HBP puesto para que pare en cuanto intente hacer algo con ese valor, pues aceptemos la ventanita de chico bueno y demos a F9:

Address	Hex dump	Disassembly	Comment
00145877	FF7424 04	push dword ptr ss:[esp+4]	crackpdf.004BB450
0014587B	E8 ABFEFFFF	call 0014572B	USER32.TranslateMessage
00145880	FF25 CC281500	jmp ds:[1528CC]	
00145886	55	push ebp	
00145887	8BEC	mov ebp, esp	
00145889	81EC 2C020000	sub esp, 22C	
0014588F	53	push ebx	
00145890	57	push edi	
00145891	6A 00	push 0	
00145893	6A 02	push 2	
00145895	32DB	xor bl, bl	
00145897	C785 D4FDFFFF	mov dword ptr ss:[ebp-22C], 22C	kernel32.CreateToolhelp32Sn
001458A1	FF15 EC2B1500	call ds:[152BEC]	
001458A7	8BF8	mov edi, eax	
001458A9	8D85 D4FDFFFF	lea eax, ss:[ebp-22C]	
001458AF	50	push eax	
001458B0	57	push edi	kernel32.Process32FirstW
001458B1	FF15 F02B1500	call ds:[152BF0]	
001458B7	85C0	test eax, eax	
001458B9	74 4E	je short 00145909	
001458BB	56	push esi	

Stack ss:[0013FCBC]=004BB450 (crackpdf.004BB450)

Address	Hex dump	ASCII	Address	Value	Comment
00488464	77 58 14 00 B8 96 39 7E FF 97 3A 7E 9C 8F 39 7E	wX.,-9~y-:~(9~	0013FCB8	00470381	RETURN to crackpdf.00470381 from 00145877
00488474	6C D0 3B 7E 22 B2 3A 7E 0E 1B 3B 7E F6 E8 3A 7E	lD;"z:~;~oe:~	0013FCBC	004BB450	crackpdf.004BB450
00488484	49 98 3A 7E 3E D3 3A 7E 72 DE 39 7E B4 90 3A 7E	I":~>0:~rB9~":~	0013FCC0	00000000	
			0013FCC4	0013FD30	
			0013FCC8	0047AF76	RETURN to crackpdf.0047AF76
			0013FCCC	00000000	
			0013FCD0	0013FD30	

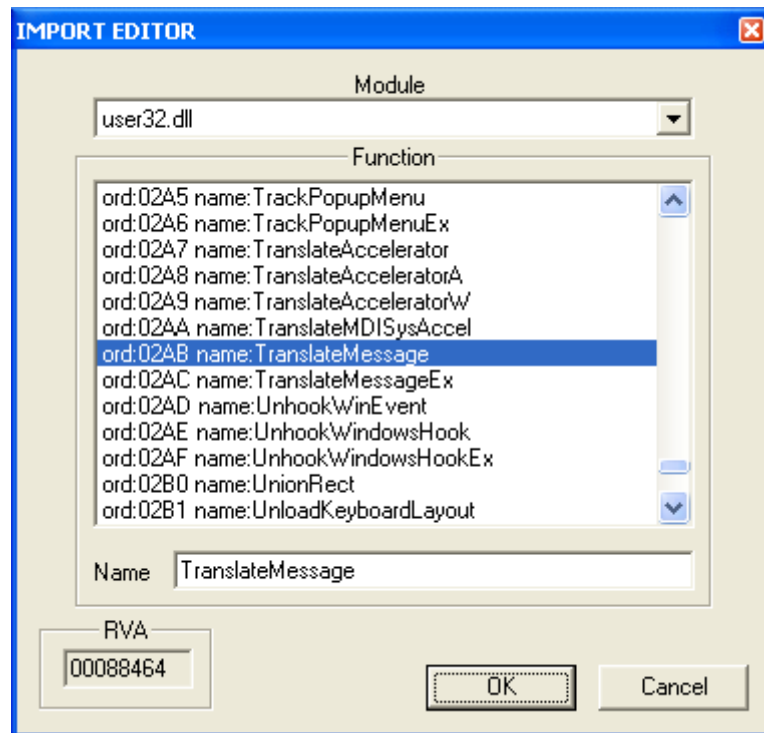
Address	Value	Comment
0013FCB8	00470381	RETURN to crackpdf.00470381 from 00145877
0013FCBC	004BB450	crackpdf.004BB450
0013FCC0	00000000	
0013FCC4	0013FD30	
0013FCC8	0047AF76	RETURN to crackpdf.0047AF76
0013FCCC	00000000	
0013FCD0	0013FD30	

Vemos que esta vez si que paro y que fue desde una CALL, la cual es indirecta ya que no apareció al hacer una búsqueda de referencias.

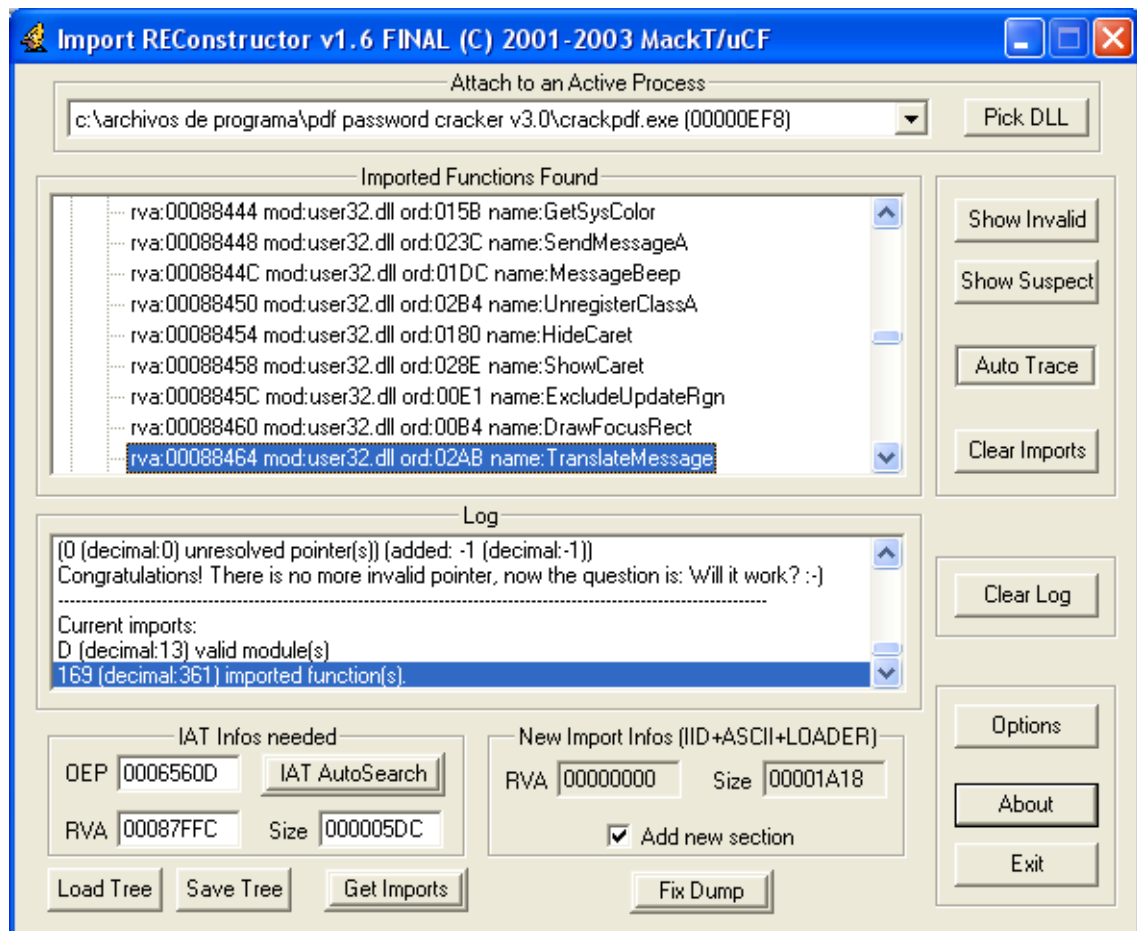
Pues yo diría que esa API es TranslateMessage así que ya tenemos todo lo que necesitamos para intentar sacar nuestro dumpeado.

Reiniciamos de nuevo la aplicación en olly y volvemos a parar en el OEP verdadero y esta vez cambiaremos todos los saltos y el SETE y lo dumpeamos.

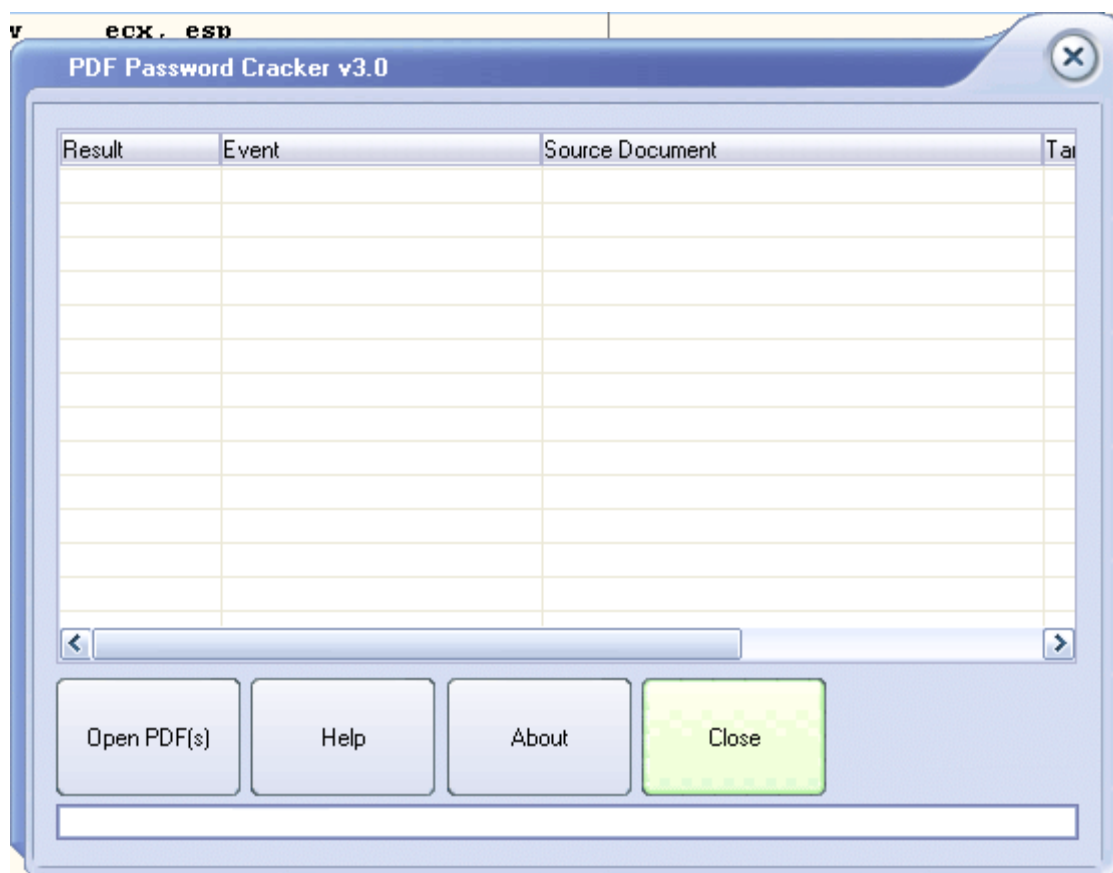
Acto seguido hacemos igual que antes con Import Rec y nos colocamos encima de la entrada mala y hacemos doble clic encima y nos mostrara una ventana como esta:



Buscamos la API TranslateMessage y damos a OK:



Como vemos quedo reparada y nos indica que ya no hay más entradas malas así que damos en Fix Dump y elegimos el dumpeado que acabamos de crear y ya tendremos nuestro ejecutable con la IAT reparada. Ahora toca ver si funciona así que hacemos doble clic en el y...



¡Funciona! Y no solo eso, sino que los mismos saltos que cambiamos son los responsables de comprobar el registro al iniciarse la aplicación, así que al cambiar los saltos conseguimos hacerle creer que estamos registrados.

Bueno, lo he probado y todo va perfecto y sin limitaciones ni nada así que objetivo cumplido jejeje.

Esto es todo y espero que os haya gustado.