

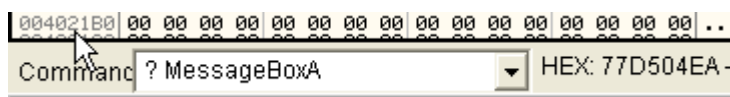
INTRODUCCION AL CRACKING CON OLLYDBG PARTE 33

Que es la IAT y como repararla

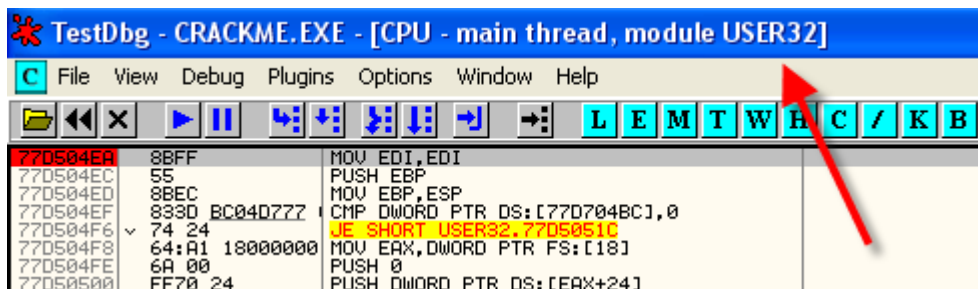
Antes de ponernos a reparar IATs como loco haremos una introduccion de que es exactamente la IAT y veremos en el CRACKME DE CRUEHEAD original donde esta ubicada y que le hace el packer a la misma, comparandolo con el crackmeUPX empacado.

Primero demos una idea generica de para que sirve todo esto que vamos a explicar:

El tema es el siguiente, como vimos las apis tienen una direccion en nuestra maquina, por ejemplo si abro el Crackme de Cruehead original en OLLYDBG y tipeo



Veo que en mi maquina la direccion donde se encuentra la api es 77D504EA, si ustedes ven la direccion de esta api en vuestras maquinas, algunos tendran la misma direccion, otros no, depende de la version de Windows que tengan, y de las actualizaciones que hayan bajado tambien, que como saben son muchisimas, y en cada una al bajar nuevas versiones de la dll que contiene la api en este caso User32.dll, generalmente cambiara la dirección.

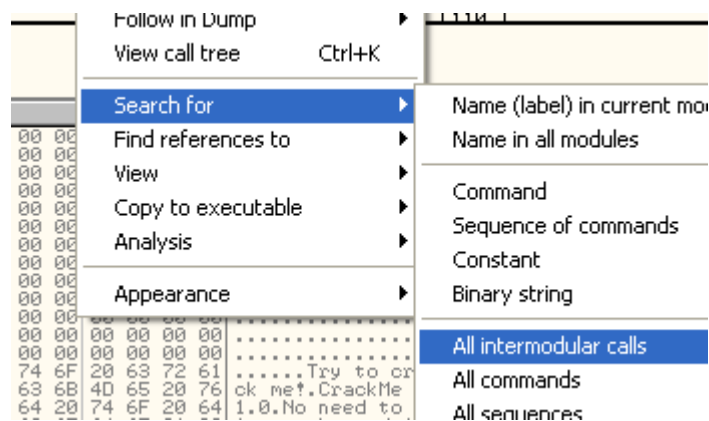


Cada vez que Microsoft saca una nueva version de esta dll, casi siempre cambiara la direccion de las apis que contiene, por lo tanto si yo programara el Crackme de Cruehead, al saltar a la api MessageBoxA, lo haria saltar a 77D504EA, y funcionaria perfectamente en mi maquina, y en las que tuvieran la suerte de tener la misma version de User32.dll que yo, en el resto de los demas Windows que o bien no sean XP, o bien no tengan la misma version de la User32.dll, saltaria a una direccion donde no estaria la api, lo cual produciria error y no correria.

El tema entonces es que el sistema operativo debe suministrar alguna forma de compatibilizar para que mi crackme funcione en otros Windows (dentro de ciertos limites) y en otras versiones de la dll debe funcionar perfectamente.

Para ello se crearon estas famosas tablas llamadas IT (Import Table), y IAT (Import Adress Table)

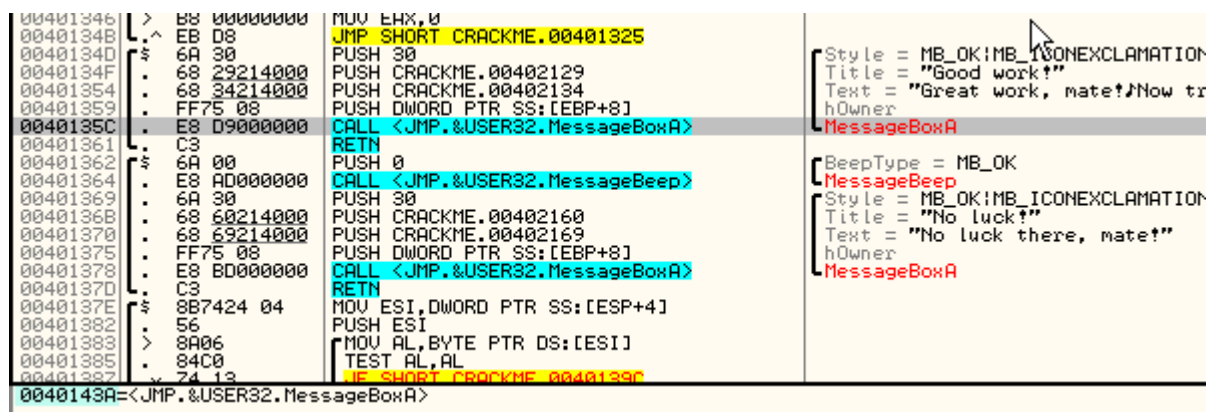
No se auyen son puro nombre una vez que uno sabe donde esta ubicada cada una en un programa desempacado y para que sirven no hay mas miedo.



Ahora hagamos SEARCH FOR-ALL INTERMODULAR CALLS y veamos los CALLS que van a otros modulos o dlls, que pueden ser calls a apis.

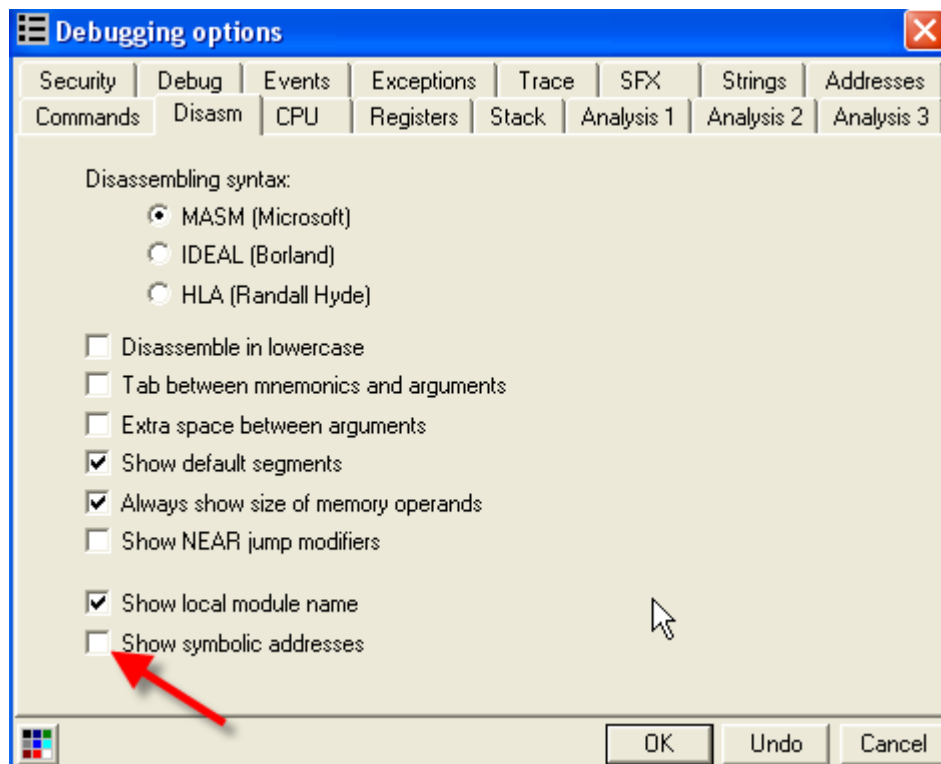
Address	Disassembly	Destination
00401000	PUSH 0	(Initial CPU selection)
00401002	CALL <JMP.&KERNEL32.GetModuleHandleA>	kernel32.GetModuleHandleA
00401013	CALL <JMP.&USER32.FindWindowA>	USER32.FindWindowA
00401052	CALL <JMP.&USER32.LoadIconA>	USER32.LoadIconA
00401063	CALL <JMP.&USER32.LoadCursorA>	USER32.LoadCursorA
00401090	CALL <JMP.&USER32.RegisterClassA>	USER32.RegisterClassA
004010C3	CALL <JMP.&USER32.CreateWindowExA>	USER32.CreateWindowExA
004010D5	CALL <JMP.&USER32.ShowWindow>	USER32.ShowWindow
004010E0	CALL <JMP.&USER32.UpdateWindow>	USER32.UpdateWindow
004010EC	CALL <JMP.&USER32.InvalidateRect>	USER32.InvalidateRect
004010FC	CALL <JMP.&USER32.GetMessageA>	USER32.GetMessageA
0040110C	CALL <JMP.&USER32.TranslateMessage>	USER32.TranslateMessage
00401116	CALL <JMP.&USER32.DispatchMessageA>	USER32.DispatchMessageA
00401123	CALL <JMP.&KERNEL32.ExitProcess>	kernel32.ExitProcess
0040118C	CALL <JMP.&USER32.DefWindowProcA>	USER32.DefWindowProcA
00401195	CALL <JMP.&USER32.PostQuitMessage>	USER32.PostQuitMessage
00401202	CALL <JMP.&USER32.ShowDialogBoxParamA>	USER32.ShowDialogBoxParamA
0040121E	CALL <JMP.&USER32.ShowDialogBoxParamA>	USER32.ShowDialogBoxParamA
00401292	CALL <JMP.&USER32.InvalidateRect>	USER32.InvalidateRect
0040129A	CALL <JMP.&USER32.SetFocus>	USER32.SetFocus
004012C4	CALL <JMP.&USER32.GetDlgItemTextA>	USER32.GetDlgItemTextA
004012E4	CALL <JMP.&USER32.GetDlgItemTextA>	USER32.GetDlgItemTextA
004012FB	CALL <JMP.&USER32.EndDialog>	USER32.EndDialog
0040133A	CALL <JMP.&USER32.EndDialog>	USER32.EndDialog
0040135C	CALL <JMP.&USER32.MessageBoxA>	USER32.MessageBoxA
00401364	CALL <JMP.&USER32.MessageBeep>	USER32.MessageBeep
00401378	CALL <JMP.&USER32.MessageBoxA>	USER32.MessageBoxA
004013BC	CALL <JMP.&USER32.MessageBoxA>	USER32.MessageBoxA

Alli vemos varios Calls a la api MessageBoxA como ejemplo, si hacemos doble click en el primero de ellos

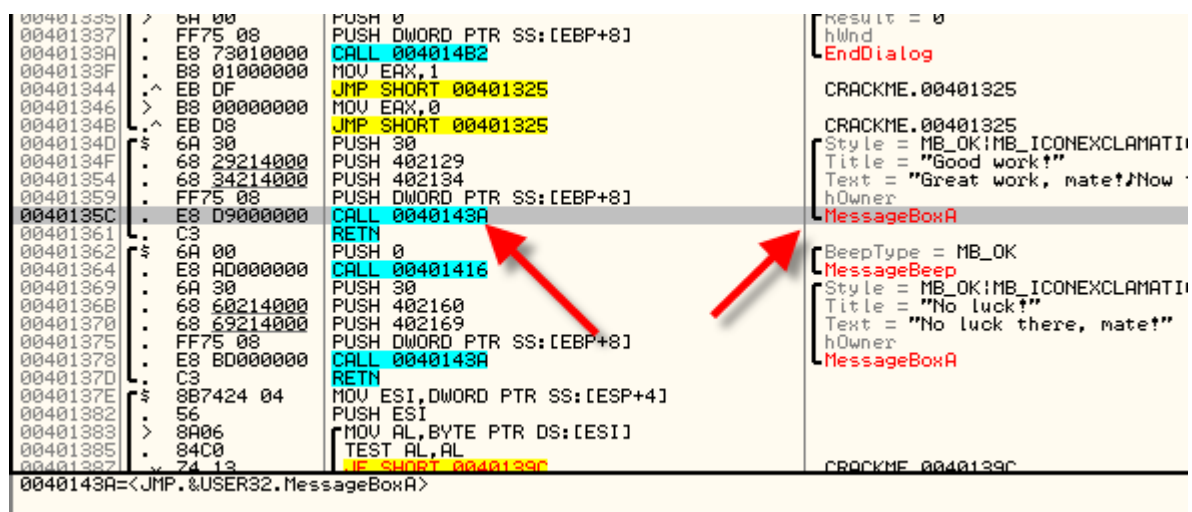


Vemos que en realidad es un CALL 40143A que OLLYDBG para aclararnos nos pone que saltara a la api MessageBoxA, cambiando la notacion por CALL <MessageBoxA> pero no es un call indirecto, es un call directo

CALL 40143A



Vemos que en las opciones del OLLYDBG en la pestaña DISASM si le quitamos la tilde a SHOW SIMBOLIC ADDRESSES el listado queda mas claro.



Vemos que igual nos aclara a la derecha los parametros y la api a la cual ira, pero ahora nos muestra el CALL realmente como call directo.

CALL 40143A

En SEARCH FOR INTERMODULARS CALLS

Address	Disassembly	Description
00401000	PUSH 0	(Initial CPU selection)
00401002	CALL 00401506	kernel32.GetModuleHandleA
00401013	CALL 004014BE	USER32.FindWindowA
00401052	CALL 00401428	USER32.LoadIconA
00401063	CALL 0040140A	USER32.LoadCursorA
00401090	CALL 00401488	USER32.RegisterClassA
004010C3	CALL 00401494	USER32.CreateWindowExA
004010D5	CALL 0040146A	USER32.ShowWindow
004010E0	CALL 00401482	USER32.UpdateWindow
004010EC	CALL 0040144C	USER32.InvalidateRect
004010FC	CALL 004014D6	USER32.GetMessageA
0040110C	CALL 0040145E	USER32.TranslateMessage
00401116	CALL 004014A6	USER32.DispatchMessageA
00401123	CALL 00401512	kernel32.ExitProcess
0040118C	CALL 0040149A	USER32.DefWindowProcA
00401195	CALL 00401440	USER32.PostQuitMessage
00401202	CALL 004014A0	USER32.MessageBoxParamA
0040121E	CALL 004014A0	USER32.MessageBoxParamA
00401292	CALL 0040144C	USER32.InvalidateRect
0040129A	CALL 00401434	USER32.SetFocus
004012C4	CALL 004014D0	USER32.GetDlgItemTextA
004012E4	CALL 004014D0	USER32.GetDlgItemTextA
004012FB	CALL 004014B2	USER32.EndDialog
0040133A	CALL 004014B2	USER32.EndDialog
0040135C	CALL 0040143A	USER32.MessageBoxA
00401364	CALL 00401416	USER32.MessageBeep
00401378	CALL 0040143A	USER32.MessageBoxA
004013BC	CALL 0040143A	USER32.MessageBoxA

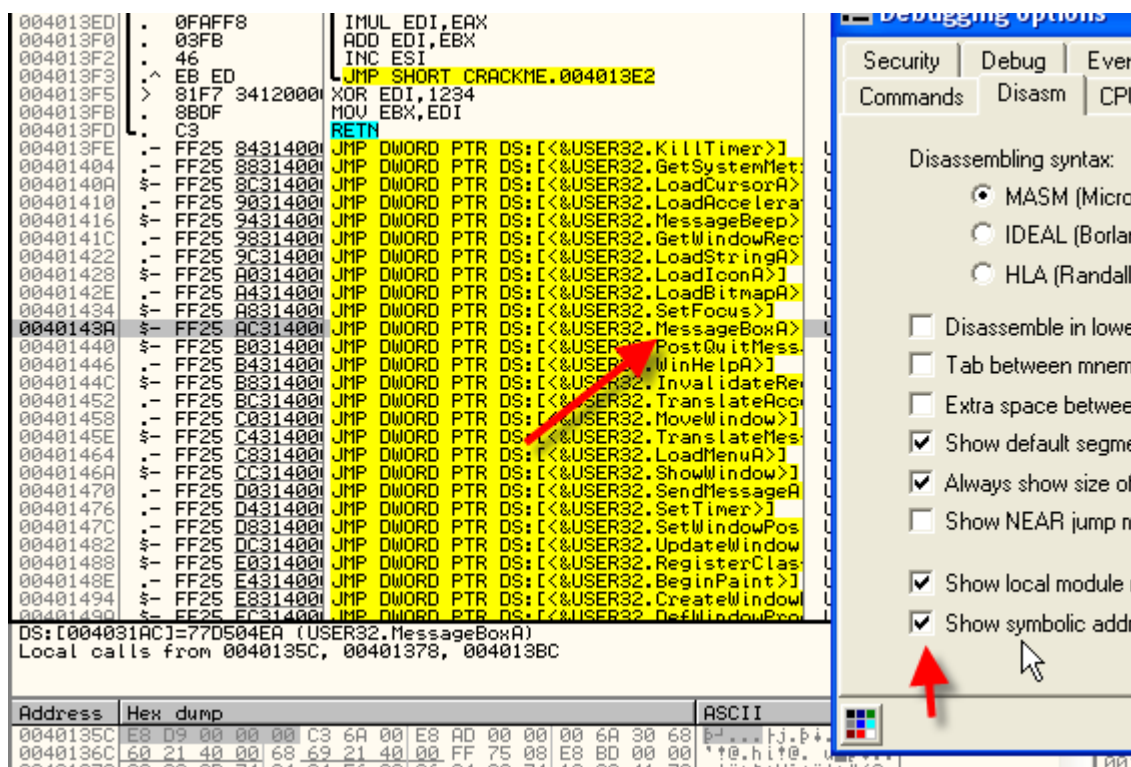
Vemos que en realidad los tres calls a MessageBoxA son calls a 40143A, pues veamos que hay alli.

00401428	FF25 A0314000	JMP DWORD PTR DS:[4031A0]	USER32.LoadIconA
0040142E	FF25 A4314000	JMP DWORD PTR DS:[4031A4]	USER32.LoadBitmapA
00401434	FF25 A8314000	JMP DWORD PTR DS:[4031A8]	USER32.SetFocus
0040143A	FF25 AC314000	JMP DWORD PTR DS:[4031AC]	USER32.MessageBoxA
00401440	FF25 B0314000	JMP DWORD PTR DS:[4031B0]	USER32.PostQuitMessage
00401448	FF25 B4314000	JMP DWORD PTR DS:[4031B4]	USER32.WinHelpA
00401450	FF25 B8314000	JMP DWORD PTR DS:[4031B8]	USER32.InvalidateRect
00401458	FF25 BC314000	JMP DWORD PTR DS:[4031BC]	USER32.TranslateAcceleratorA
00401460	FF25 C0314000	JMP DWORD PTR DS:[4031C0]	USER32.MoveWindow
00401468	FF25 C4314000	JMP DWORD PTR DS:[4031C4]	USER32.TranslateMessage
00401470	FF25 C8314000	JMP DWORD PTR DS:[4031C8]	USER32.LoadMenuA
00401478	FF25 CC314000	JMP DWORD PTR DS:[4031CC]	USER32.ShowWindow
00401480	FF25 D0314000	JMP DWORD PTR DS:[4031D0]	USER32.SendMessageA
00401488	FF25 D4314000	JMP DWORD PTR DS:[4031D4]	USER32.SetTimer
00401490	FF25 D8314000	JMP DWORD PTR DS:[4031D8]	USER32.SetWindowPos
00401498	FF25 DC314000	JMP DWORD PTR DS:[4031DC]	USER32.UpdateWindow
004014A0	FF25 E0314000	JMP DWORD PTR DS:[4031E0]	USER32.RegisterClassA
004014B0	FF25 E4314000	JMP DWORD PTR DS:[4031E4]	USER32.BeginPaint
004014C0	FF25 E8314000	JMP DWORD PTR DS:[4031E8]	USER32.CreateWindowExA
004014D0	FF25 EC314000	JMP DWORD PTR DS:[4031EC]	USER32.DefWindowProcA

DS:[004031AC]=77D504EA (USER32.MessageBoxA)
Local calls from 0040135C, 00401378, 004013BC

Aquí vemos el quid de la cuestion, para llegar a la api hace un JMP indirecto, que al tenerlo en esta forma de visualizacion se ve claro que toma el valor a donde saltara de 4031AC

JMP [4031AC]



Tambien vemos que si ponemos la tilde, se nos complica un poco entender que es un salto indirecto, por lo cual la quitaremos mientras estamos trabajando con IATs.

Y aquí esta el truco, el programa salta a la api con un JMP indirecto que toma la direccion que la api tiene en nuestra maquina de 4031AC, y vemos que para todas las otras apis, hay otros JMPS INDIRECTOS similares.

Ahora empezamos a ver un poco mas claro el tema de la compatibilidad entre maquina, cuando el programa tiene que ir a una api, hace un salto indirecto, leyendo la direccion de la api de una especie de deposito, si vemos en el dump este deposito.

0040141C	FF25	9C314000	JMP	DWORD	PTR	DS:[4003198]	USER32.GetWindowRect
00401422	FF25	9C314000	JMP	DWORD	PTR	DS:[400319C]	USER32.LoadStringA
00401428	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A0]	USER32.LoadIconA
0040142E	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A4]	USER32.LoadBitmapA
00401434	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A8]	USER32.SetFocus
0040143A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031AC]	USER32.MessageBoxA
00401440	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B0]	USER32.PostQuitMessage
00401446	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B4]	USER32.WinHelpA
0040144C	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B8]	USER32.InvalidateRect
00401452	FF25	9C314000	JMP	DWORD	PTR	DS:[40031BC]	USER32.TranslateAcceleratorA
00401458	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C0]	USER32.MoveWindow
0040145E	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C4]	USER32.TranslateMessage
00401464	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C8]	USER32.LoadMenuA
0040146A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031CC]	USER32.ShowWindow
00401470	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D0]	USER32.SendMessageA
00401476	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D4]	USER32.SetTimer
0040147C	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D8]	USER32.SetWindowPos
00401482	FF25	9C314000	JMP	DWORD	PTR	DS:[40031DC]	USER32.UpdateWindow
00401488	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E0]	USER32.RegisterClassA
0040148E	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E4]	USER32.BeginPaint
00401494	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E8]	USER32.CreateWindowExA
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031EC]	USER32.DefWindowProcA

DS:[004031AC]=77D504EA (USER32.MessageBoxA)
Local calls from 0040135C, 00401378, 004013BC

Address	Hex dump	ASCII
004031AC	EA 04 05 77 11 12 02 77	U♦!w♦#ew
004031B4	35 EE 03 77 F5 B5 01 77	S~ew\$~0w
004031BC	9C FA 02 77 EC DB 01 77	£~ewy~0w
004031C4	F6 8B 01 77 83 F7 04 77	+i0w~ew
004031CC	A4 D8 01 77 9A F3 02 77	Ki0w%ew
004031D4	2E 8C 01 77 1B C0 01 77	.i0w+~0w
004031DC	F9 07 01 77 8C 14 02 77	~i0wi0ew
004031E4	09 B6 01 77 5E 02 02 77	~0w^0ew
004031EC	EE 04 01 77 1C B1 03 77	~0wL~ew
004031F4	B8 96 01 77 9C F3 04 77	@0w0%ew
004031FC	50 62 02 77 10 B6 01 77	P0ew#~0w
00403204	81 E5 02 77 C7 86 01 77	u0ew\$~0w
0040320C	16 48 02 77 1E AC 06 77	~HEW%iw
00403214	42 10 02 77 00 00 00 00	B~ew....
0040321C	C1 C9 80 7C 99 6B 82 7C	~F0k0k0k

Alli vemos realmente que 4031AC es una parte de un deposito que contiene todas las direcciones a las apis en mi maquina, ahora este DEPOSITO es la famosa IAT o IMPORT TABLE ADDRESS, aquí esta la clave de la compatibilidad, quiere decir que todos los programas tendran los mismos JMPs INDIRECTOS a las apis, lo que cambiara es el valor guardado en el deposito, para cada maquina tendra direcciones diferentes que llevaran el salto indirecto, a la direccion correcta de la api.

Pero un momento me diran algunos, 4031AC es una direccion del programa, por lo cual si tiene diferentes direcciones para cada maquina, pues entonces el programa seria diferente para cada maquina?

Jeje buena pregunta ahora veremos como funciona el sistema, pero ya sabemos el objetivo, el cual es llenar la IAT de las direcciones correctas para cada maquina.

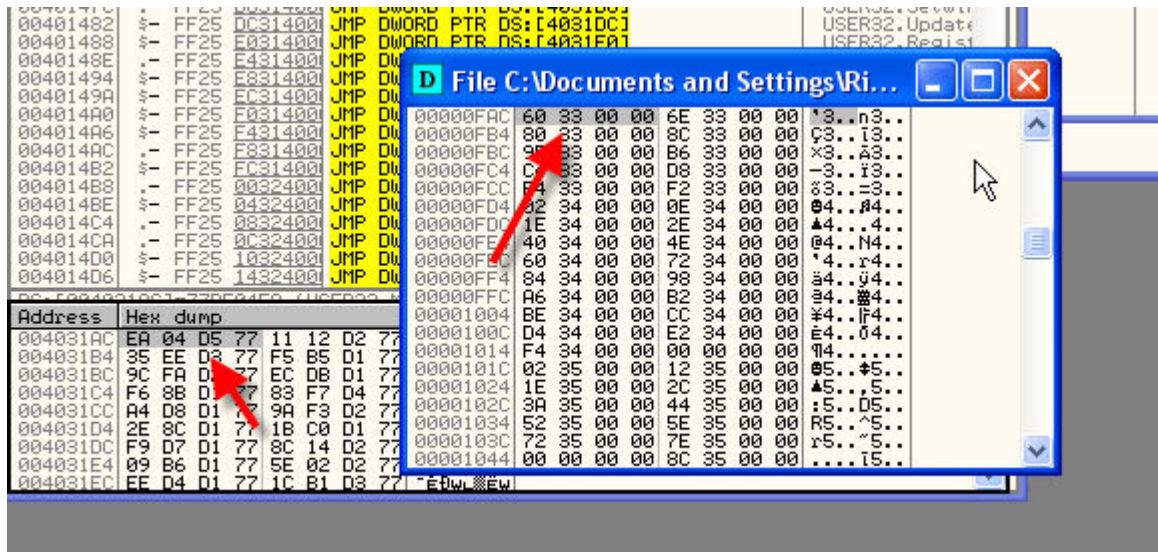
0040143A	FF25	9C314000	JMP	DWORD	PTR	DS:[4003198]	USER32.GetWindowRect
0040143A	FF25	9C314000	JMP	DWORD	PTR	DS:[400319C]	USER32.LoadStringA
00401440	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A0]	USER32.LoadIconA
00401446	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A4]	USER32.LoadBitmapA
0040144C	FF25	9C314000	JMP	DWORD	PTR	DS:[40031A8]	USER32.SetFocus
00401452	FF25	9C314000	JMP	DWORD	PTR	DS:[40031AC]	USER32.MessageBoxA
00401458	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B0]	USER32.PostQuitMessage
0040145E	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B4]	USER32.WinHelpA
00401464	FF25	9C314000	JMP	DWORD	PTR	DS:[40031B8]	USER32.InvalidateRect
0040146A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031BC]	USER32.TranslateAcceleratorA
00401470	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C0]	USER32.MoveWindow
00401476	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C4]	USER32.TranslateMessage
0040147C	FF25	9C314000	JMP	DWORD	PTR	DS:[40031C8]	USER32.LoadMenuA
00401482	FF25	9C314000	JMP	DWORD	PTR	DS:[40031CC]	USER32.ShowWindow
00401488	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D0]	USER32.SendMessageA
0040148E	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D4]	USER32.SetTimer
00401494	FF25	9C314000	JMP	DWORD	PTR	DS:[40031D8]	USER32.SetWindowPos
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031DC]	USER32.UpdateWindow
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E0]	USER32.RegisterClassA
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E4]	USER32.BeginPaint
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031E8]	USER32.CreateWindowExA
0040149A	FF25	9C314000	JMP	DWORD	PTR	DS:[40031EC]	USER32.DefWindowProcA

DS:[004031AC]=77D504EA (USER32.MessageBoxA)
Local calls from 0040135C, 00401378, 004013BC

Address	Hex dump	ASCII
004031AC	EA 04 05 77 11 12 02 77	U♦!w♦#ew
004031B4	35 EE 03 77 F5 B5 01 77	S~ew\$~0w
004031BC	9C FA 02 77 EC DB 01 77	£~ewy~0w
004031C4	F6 8B 01 77 83 F7 04 77	+i0w~ew
004031CC	A4 D8 01 77 9A F3 02 77	Ki0w%ew

Sabemos que si hacemos VIEW- EJECUTABLE FILE podemos ver lo que hay en la direccion

4031AC, realmente en el archivo ejecutable guardado en nuestro disco rigido.



Vemos que el valor en el exe, es 60 33, y que nosotros parados en el entry point ya tenemos en la memoria en la misma posicion la direccion de la api EA 04 D5 77 guardada alli, quiere decir que el sistema al arrancar el programa, tomo el 3360 que habia alli en el OFFSET 0FAC que corresponde a la direccion 4031ac y lo machaco con el valor de la direccion real de la api en mi maquina.

Magia?

No no asi trabaja Windows, cada archivo que arranca, el sistema operativo llena la IAT de las direcciones correctas de las apis en mi maquina, en este caso lleno 4031AC con el valor de la api MessageBoxA y los valores alrededor, con las direcciones de otras apis, para completar la tabla IAT o deposito de direcciones de las apis en mi maquina.

Ahora el sistema no es mago como hace esto, vemos que en 0FAC tiene un valor 3360, le indicara algo al sistema este puntero, para que sepa que api es la que debe llenar alli?

Pues si 3360 es el valor al cual si le sumamos la imagebase queda 403360 y si miramos en el sump esa direccion que vemos?

Address	Hex dump	ASCII
00403360	00 00 40 65 73 73 61 67	..Message
00403368	65 42 6F 78 41 00 00 00	eBoxA..
00403370	50 6F 73 74 51 75 69 74	PostQui
00403378	40 65 73 73 61 67 65 00	Message
00403380	00 00 57 69 6E 48 65 6C	..WinHel
00403388	70 41 00 00 00 00 49 6E	pA....In
00403390	76 61 6C 69 64 61 74 65	validate
00403398	52 65 63 74 00 00 00 00	Rect....
004033A0	54 72 61 6E 73 6C 61 74	Translat
004033A8	65 41 63 68 65 6C 65 72	eAcceler
004033B0	61 74 6F 72 41 00 00 00	atorA...
004033B8	40 6F 76 65 57 69 6E 64	MoveWind
004033C0	6F 77 00 00 00 00 54 72	ow....Tr
004033C8	61 6E 73 6C 61 74 65 40	anslateM
004033D0	65 73 73 61 67 65 00 00	essage..
004033D8	00 00 4C 6F 61 64 40 65	..LoadMe
004033E0	6E 75 41 00 00 00 53 68	nuA...Sh
004033E8	6F 77 57 69 6E 64 6F 77	owWindow
004033F0	00 00 00 00 53 65 6E 64Send
004033F8	40 65 73 73 61 67 65 41	MessageA
00403400	00 00 00 00 53 65 74 54SetT
00403408	69 6D 65 72 00 00 00 00	imer....

Pues aquí esta el truco es un puntero a la string MessageBoxA, eso quiere decir que el sistema mira ese puntero, busca que api es por el nombre y con GetProcAddress, saca la direccion de dicha api en

nuestra maquina, y lo completa en la IAT, machacando el 3360, con la direccion real de la api en nuestra maquina, y de esta forma se asegura que el programa funcione para cualquier version de la dll, porque halla la direccion antes de llegar al Entry Point del programa y una vez que llego alli, queda la iat completita con las direcciones de las apis en mi maquina, si miro la iat de otra maquina diferente vere que el contenido de 4031AC es diferente pues la direccion de la api, es diferente, pero cuando haga.

JMP [4031AC]

ira a su MessageBoxA al igual que en mi maquina, el programador no se tiene que preocupar, siempre que haga calls o jmps indirectos al valor que se encuentra en el deposito de la IAT, este estara correcto, al arrancar el sistema coloco en tiempo real la direccion correcta, apuntando a MessageBoxA en todas las maquina para asegurar compatibilidad.

O sea que para que el sistema me complete correctamente la direccion de la api en la iat, debe el archivo tener varios punteros como vimos.

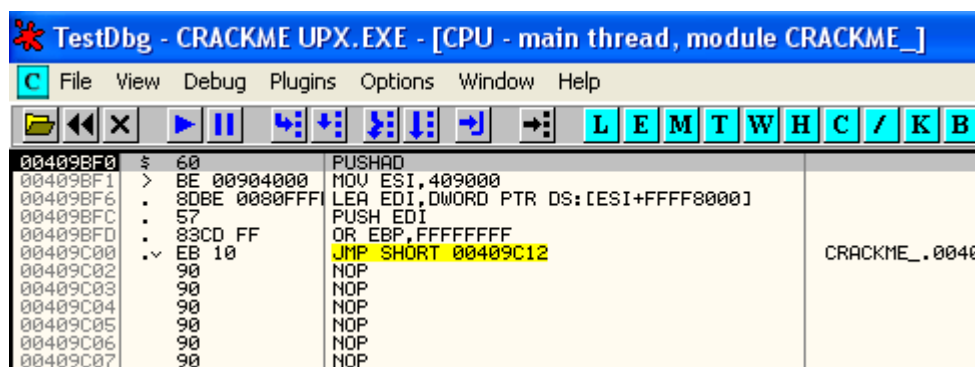
- 1)en la entrada de la iat en el ejecutable en mi rigido, debe tener un puntero a una cadena de texto que le identifique al sistema que api debe colocar en esa entrada.
- 2)Por supuesto debe tener la cadena de texto con el nombre de la api

Pues con estos dos puntos, podemos concluir que el programa arrancara y el sistema me llenara la IAT con los valores correctos. (mas adelante veremos el trabajo completo de este sistema)

Ahora ya vimos como llegamos al OEP de un programa empackado, porque se dice que cuando llegamos alli y dumpeamos un programa debemos reconstruir la IAT, los packers la rompen? Que hacen con ella, pues si la rompen algunos mas otros menos, el truco es que el packer no necesita la iat del programa en si porque arranca desde el mismo desempacador, y puede leer las apis que deberian ir en la IAT, mientras va desempacando el programa, y una vez que leyo cada api, y lleno la iat con las direcciones correctas de cada api, pues las strings que le indicaron cual era cada api, las borrara, ya no las necesita, pues el programa ya arranco y tiene ya en la IAT las direcciones correctas.

Es mas en los packers ni siquiera existen esas strings ya que el programa o bien las guarda encriptadas o bien las guarda en alguna otra direccion que no sean facil de hallar por nosotros los crackers.

Veamos el ejemplo con el empackado UPX para ver la diferencia y como nos queda al dumpearlo, y porque hay que repararlo.



Aquí tenemos el Crackme de Cruehead que esta empackado con UPX que habiamos hecho en partes

anteriores.

Miremos la direccion 4031AC donde estaba el deposito de apis en el original

Address	Hex dump	ASCII
004031AC	00 00 00 00 00 00 00 00
004031B4	00 00 00 00 00 00 00 00
004031BC	00 00 00 00 00 00 00 00
004031C4	00 00 00 00 00 00 00 00
004031CC	00 00 00 00 00 00 00 00
004031D4	00 00 00 00 00 00 00 00
004031DC	00 00 00 00 00 00 00 00
004031E4	00 00 00 00 00 00 00 00
004031EC	00 00 00 00 00 00 00 00
004031F4	00 00 00 00 00 00 00 00
004031FC	00 00 00 00 00 00 00 00
00403204	00 00 00 00 00 00 00 00
0040320C	00 00 00 00 00 00 00 00
00403214	00 00 00 00 00 00 00 00
0040321C	00 00 00 00 00 00 00 00
00403224	00 00 00 00 00 00 00 00
0040322C	00 00 00 00 00 00 00 00
00403234	00 00 00 00 00 00 00 00
0040323C	00 00 00 00 00 00 00 00
00403244	00 00 00 00 00 00 00 00
0040324C	00 00 00 00 00 00 00 00
00403254	00 00 00 00 00 00 00 00
0040325C	00 00 00 00 00 00 00 00
00403264	00 00 00 00 00 00 00 00
0040326C	00 00 00 00 00 00 00 00
00403274	00 00 00 00 00 00 00 00
0040327C	00 00 00 00 00 00 00 00
00403284	00 00 00 00 00 00 00 00

Command

nada borro todo, y las strings de las apis estaban en 403360 que haya alli?

Address	Hex dump	ASCII
00403360	00 00 00 00 00 00 00 00
00403368	00 00 00 00 00 00 00 00
00403370	00 00 00 00 00 00 00 00
00403378	00 00 00 00 00 00 00 00
00403380	00 00 00 00 00 00 00 00
00403388	00 00 00 00 00 00 00 00
00403390	00 00 00 00 00 00 00 00
00403398	00 00 00 00 00 00 00 00
004033A0	00 00 00 00 00 00 00 00
004033A8	00 00 00 00 00 00 00 00
004033B0	00 00 00 00 00 00 00 00
004033B8	00 00 00 00 00 00 00 00
004033C0	00 00 00 00 00 00 00 00
004033C8	00 00 00 00 00 00 00 00
004033D0	00 00 00 00 00 00 00 00
004033D8	00 00 00 00 00 00 00 00
004033E0	00 00 00 00 00 00 00 00
004033E8	00 00 00 00 00 00 00 00
004033F0	00 00 00 00 00 00 00 00
004033F8	00 00 00 00 00 00 00 00
00403400	00 00 00 00 00 00 00 00

Nada, de nada, lleguemos al oep y veamos que hay en estas mismas direcciones, ya que para que el programa corra en la iat debe haber algo si no al saltar a la api, y hacer

JMP [4031ac]

y estar vacio dara error, o sea que el packer hara el trabajo que hace normalmente el sistema y llenara la iat, lleguemos al OEP.

00409D2D	09C0	OR EAX,EAX	
00409D2F	74 07	JE SHORT 00409D38	CRACKME_.00409D38
00409D31	8903	MOV DWORD PTR DS:[EBX],EAX	
00409D33	83C3 04	ADD EBX,4	
00409D36	EB E1	JMP SHORT 00409D19	CRACKME_.00409D19
00409D38	FF96 60950000	CALL DWORD PTR DS:[ESI+9560]	
00409D3E	61	POPAD	
00409D3F	E9 BC72FFFF	JMP 00401000	CRACKME_.00401000
00409D44	00	DB 00	
00409D45	00	DB 00	
00409D46	00	DB 00	
00409D47	00	DB 00	
00409D48	00	DB 00	
00409D49	00	DB 00	
00409D4A	00	DB 00	

Ponemos un BP en el salto al OEP y damos RUN y llegamos alli

Veamos que hay en el lugar de la iat

0040142E	FF25 44314000	JMP DWORD PTR DS:[403144]	USER32.LoadBitmapH
00401434	FF25 A8314000	JMP DWORD PTR DS:[4031A8]	USER32.SetFocus
0040143A	FF25 AC314000	JMP DWORD PTR DS:[4031AC]	USER32.MessageBoxA
00401440	FF25 B0314000	JMP DWORD PTR DS:[4031B0]	USER32.PostQuitMessage
00401446	FF25 B4314000	JMP DWORD PTR DS:[4031B4]	USER32.WinHelpA
0040144C	FF25 B8314000	JMP DWORD PTR DS:[4031B8]	USER32.InvalidRect
00401452	FF25 BC314000	JMP DWORD PTR DS:[4031BC]	USER32.TranslateAcceler
00401458	FF25 C0314000	JMP DWORD PTR DS:[4031C0]	USER32.MoveWindow
0040145E	FF25 C4314000	JMP DWORD PTR DS:[4031C4]	USER32.TranslateMessage
00401464	FF25 C8314000	JMP DWORD PTR DS:[4031C8]	USER32.LoadMenuA
0040146A	FF25 CC314000	JMP DWORD PTR DS:[4031CC]	USER32.ShowWindow
00401470	FF25 D0314000	JMP DWORD PTR DS:[4031D0]	USER32.SendMessageA
00401476	FF25 D4314000	JMP DWORD PTR DS:[4031D4]	USER32.SetTimer
0040147C	FF25 D8314000	JMP DWORD PTR DS:[4031D8]	USER32.SetWindowPos
00401482	FF25 DC314000	JMP DWORD PTR DS:[4031DC]	USER32.UpdateWindow
00401488	FF25 E0314000	JMP DWORD PTR DS:[4031E0]	USER32.RegisterClassA
0040148E	FF25 E4314000	JMP DWORD PTR DS:[4031E4]	USER32.BeginPaint
00401494	FF25 E8314000	JMP DWORD PTR DS:[4031E8]	USER32.CreateWindowExA
0040149A	FF25 EC314000	JMP DWORD PTR DS:[4031EC]	USER32.DefWindowProcA
004014A0	FF25 F0314000	JMP DWORD PTR DS:[4031F0]	USER32.DialogBoxParamA
004014A6	FF25 F4314000	JMP DWORD PTR DS:[4031F4]	USER32.DispatchMessageA
004014AC	FF25 F8314000	JMP DWORD PTR DS:[4031F8]	USER32.DrawMenuBar
004014B2	FF25 FC314000	JMP DWORD PTR DS:[4031FC]	USER32.EndDialog
004014B8	FF25 00324000	JMP DWORD PTR DS:[403200]	USER32.EndPaint
004014BE	FF25 04324000	JMP DWORD PTR DS:[403204]	USER32.FindWindowA

DS:[004031AC]=77D504EA (USER32.MessageBoxA)

Address	Hex dump	ASCII
004031AC	EA 04 D5 77 11 12 D2 77	04'w4\$Ew
004031B4	35 EE D3 77 F5 B5 D1 77	5-Ew3A0w
004031BC	9C FA D2 77 EC DB D1 77	6-Ew3A0w
004031C4	F6 8B D1 77 83 F7 D4 77	÷i0w3-Ew
004031CC	A4 D8 D1 77 9A F3 D2 77	xi0w0%Ew
004031D4	2E 8C D1 77 1B C0 D1 77	.i0w+L0w
004031DC	F9 D7 D1 77 8C 14 D2 77	..i0wi0Ew

Vemos que para que el programa funcione el desempacador lleno la iat con las direcciones correctas de la api en mi maquina, pero si nosotros dumpeamos aquí habra un problema, el ejecutable que salga del dumpeado le faltan datos para arrancar.

Ya de por si las strings que identificaban cada api y estaban en 403360 no estan

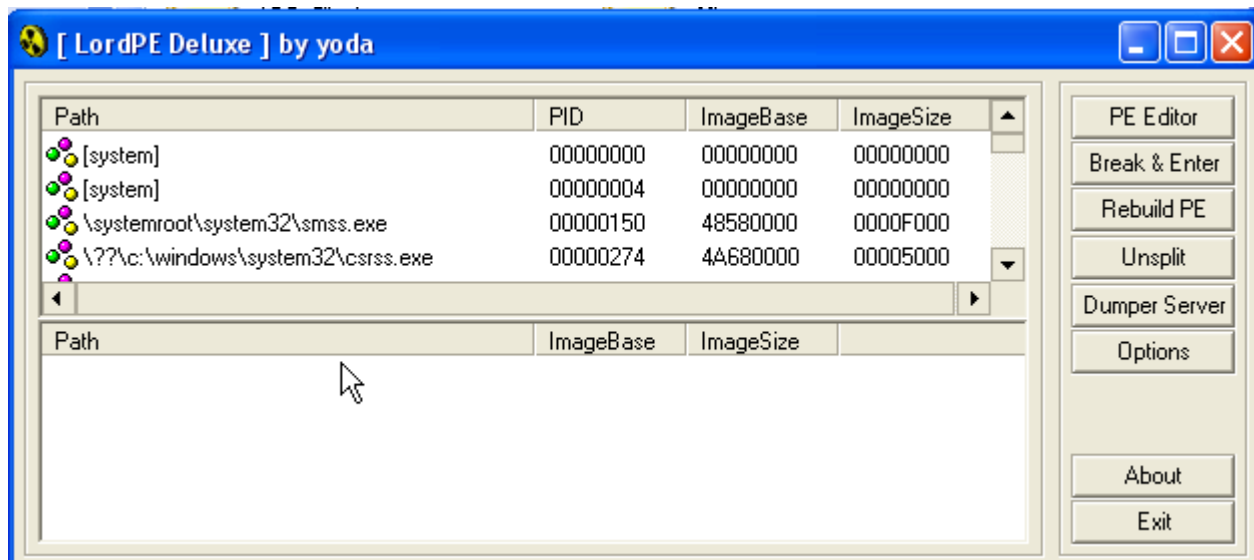
Address	Hex dump	ASCII
00403360	00 00 00 00 00 00 00 00
00403368	00 00 00 00 00 00 00 00
00403370	00 00 00 00 00 00 00 00
00403378	00 00 00 00 00 00 00 00
00403380	00 00 00 00 00 00 00 00
00403388	00 00 00 00 00 00 00 00
00403390	00 00 00 00 00 00 00 00
00403398	00 00 00 00 00 00 00 00
004033A0	00 00 00 00 00 00 00 00
004033A8	00 00 00 00 00 00 00 00
004033B0	00 00 00 00 00 00 00 00
004033B8	00 00 00 00 00 00 00 00
004033C0	00 00 00 00 00 00 00 00
004033C8	00 00 00 00 00 00 00 00
004033D0	00 00 00 00 00 00 00 00
004033D8	00 00 00 00 00 00 00 00
004033E0	00 00 00 00 00 00 00 00
004033E8	00 00 00 00 00 00 00 00

Veamos hagamos un dumpeado a ver que queda, pero ya nos damos cuenta que tendremos el

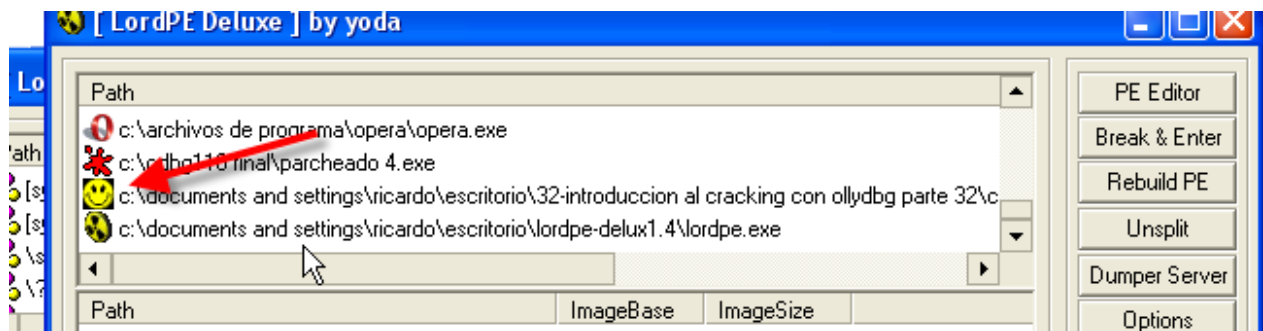
codigo del programa correcto, pero el dumpeado no correra al no poder el sistema llenar la iat por falta de datos.

Por ahora utilizaremos un programa externo al OLLYDBG para dumpear este es el LORDPE DELUXE que se puede bajar desde mi HTTP.

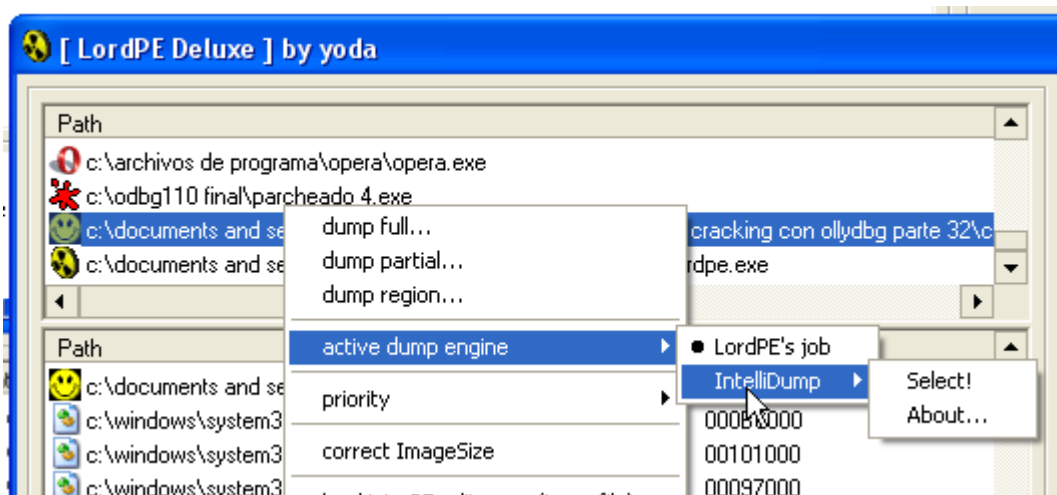
<http://www.ricnar456.dyndns.org/HERRAMIENTAS/L-M-N-O-P/lordpe-delux1.4.zip>



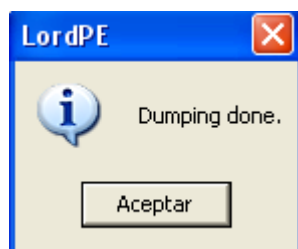
Alli tenemos al LORD PE DELUXE busquemos el proceso a dumpear que es el crackmeUPX que tenemos detenido en el OEP.



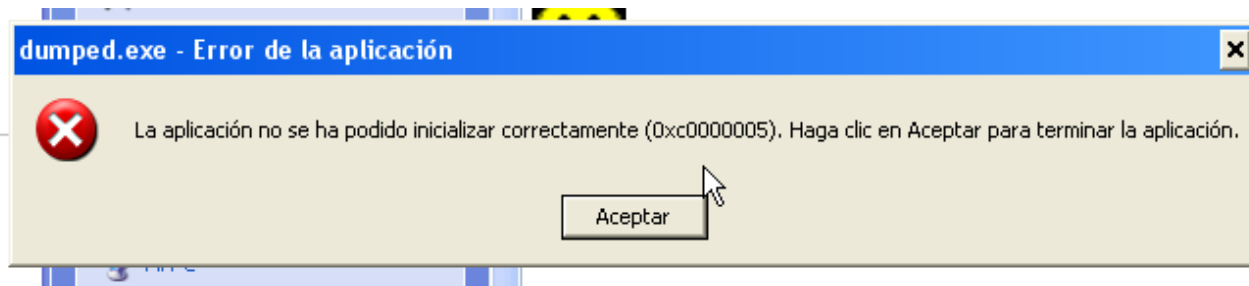
Alli esta el crackmeUPX, marquemoslo



Haciendo click derecho cambiemos a INTELLIDUMP que dicen que es mejor dumpeando, jeje y luego hagamos click en DUMP FULL.

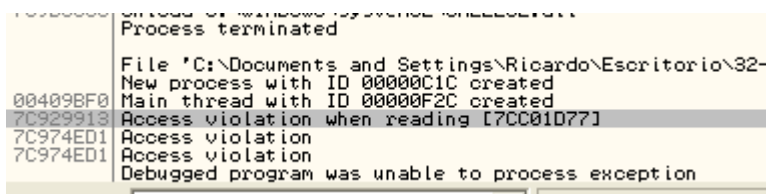


Lo hemos guardado con el nombre DUMPED.exe



Si corremos el dumpeado vemos que no arranca tratemos de abrirlo en OLLYDBG

Y nos da el mismo error pero si vemos el LOG del OLLYDBG los errores



El error se produjo en 7c929913 en mi maquina vayamos a esta direccion, ustedes en su maquina a la direccion que figure en el LOG.

Pues no hay que ser un genio para darse cuenta que para que arranque el programa hay que restaurar los nombres de las apis, y los punteros a ellas, si quisiera hacerlo a mano es un trabajo terrible deberia cambiar el contenido de 4031AC por un puntero a la string de MessageBoxA y con ciertos arreglos de algunos punteros, pues arrancaria.

Aquí tenemos planteado el desafio, esto es lo que debemos reparar para que nuestro dumpeado ademas de tener el codigo correcto, que por supuesto lo tiene, corra perfectamente.

De que tiene el codigo correcto no hay duda vayamos a 401000 y miremos que hay alli

Address	Disassembly	Comment
00401000	6A 00	PUSH 0
00401002	E8 FF040000	CALL 00401506
00401007	A3 CA204000	MOV DWORD PTR DS:[4020CA],EAX
0040100C	6A 00	PUSH 0
0040100E	68 F4204000	PUSH 4020F4
00401013	E8 A6040000	CALL 004014BE
00401018	0BC0	OR EAX,EAX
0040101A	74 01	JE SHORT 0040101D
0040101C	C3	RETN
0040101D	C705 64204000	MOV DWORD PTR DS:[402064],4003
00401027	C705 68204000	MOV DWORD PTR DS:[402068],401128
00401031	C705 6C204000	MOV DWORD PTR DS:[40206C],0
0040103B	C705 70204000	MOV DWORD PTR DS:[402070],0
00401045	A1 CA204000	MOV EAX,DWORD PTR DS:[4020CA]
0040104A	A3 74204000	MOV DWORD PTR DS:[402074],EAX
0040104F	6A 64	PUSH 64
00401051	50	PUSH EAX
00401052	E8 D1030000	CALL 00401428
00401057	A3 78204000	MOV DWORD PTR DS:[402078],EAX
0040105C	68 007F0000	PUSH 7F00
00401061	6A 00	PUSH 0
00401063	E8 A2030000	CALL 0040140A
00401068	A3 7C204000	MOV DWORD PTR DS:[40207C],EAX
0040106D	C705 80204000	MOV DWORD PTR DS:[402080],5
00401077	C705 84204000	MOV DWORD PTR DS:[402084],402110
00401081	C705 88204000	MOV DWORD PTR DS:[402088],4020F4
0040108B	68 64204000	PUSH 402064
00401090	E8 F3030000	CALL 00401488
00401095	6A 00	PUSH 0

ASCII "No need to disasm the code!"

ASCII "MENU"

ASCII "No need to disasm the code!"

Si vamos a la zona de los saltos a las apis

Alli vemos el call a la MessageBoxA que habiamos visto anteriormente

Address	Disassembly	Comment
00401337	FF75 08	JMP SHORT 0040133F
0040133A	E8 73010000	CALL 004014B2
0040133F	B8 01000000	MOV EAX,1
00401344	EB DF	JMP SHORT 00401325
00401346	B8 00000000	MOV EAX,0
0040134B	EB D8	JMP SHORT 00401325
0040134D	6A 30	PUSH 30
0040134F	68 29214000	PUSH 402129
00401354	68 34214000	PUSH 402134
00401359	FF75 08	JMP SHORT 0040133F
0040135C	E8 D9000000	CALL 0040143A
00401361	C3	RETN
00401362	6A 00	PUSH 0
00401364	E8 AD000000	CALL 00401416
00401369	6A 30	PUSH 30
0040136B	68 60214000	PUSH 402160
00401370	68 69214000	PUSH 402169
00401375	FF75 08	JMP SHORT 0040133F
00401378	E8 BD000000	CALL 0040143A
0040137D	C3	RETN
0040137F	8B7424 04	MOV ESI,DWORD PTR SS:[ESP+4]

ASCII "Good work!"

ASCII "Great work, mate! Now try the next CrackMe!"

ASCII "No luck!"

ASCII "No luck there, mate!"

En el CALL el codigo esta correcto, nos lleva al mismo JUMP que antes

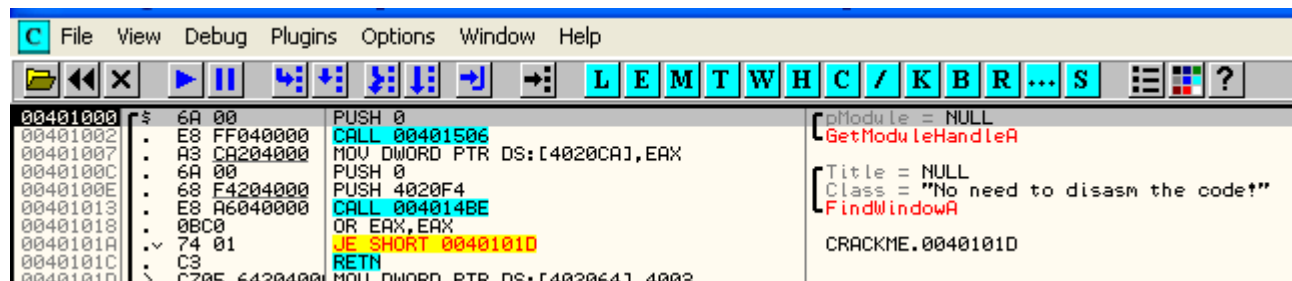
004013FB	8BDF	MOV EBX,EDI
004013FD	C3	RETN
004013FE	- FF25 84314000	JMP DWORD PTR DS:[403184]
00401404	- FF25 88314000	JMP DWORD PTR DS:[403188]
0040140A	- FF25 8C314000	JMP DWORD PTR DS:[40318C]
00401410	- FF25 90314000	JMP DWORD PTR DS:[403190]
00401416	- FF25 94314000	JMP DWORD PTR DS:[403194]
0040141C	- FF25 98314000	JMP DWORD PTR DS:[403198]
00401422	- FF25 9C314000	JMP DWORD PTR DS:[40319C]
00401428	- FF25 A0314000	JMP DWORD PTR DS:[4031A0]
0040142E	- FF25 A4314000	JMP DWORD PTR DS:[4031A4]
00401434	- FF25 A8314000	JMP DWORD PTR DS:[4031A8]
0040143A	- FF25 AC314000	JMP DWORD PTR DS:[4031AC]
00401440	- FF25 B0314000	JMP DWORD PTR DS:[4031B0]
00401446	- FF25 B4314000	JMP DWORD PTR DS:[4031B4]
0040144C	- FF25 B8314000	JMP DWORD PTR DS:[4031B8]
00401452	- FF25 BC314000	JMP DWORD PTR DS:[4031BC]
00401458	- FF25 C0314000	JMP DWORD PTR DS:[4031C0]
0040145E	- FF25 C4314000	JMP DWORD PTR DS:[4031C4]
00401464	- FF25 C8314000	JMP DWORD PTR DS:[4031C8]
0040146A	- FF25 CC314000	JMP DWORD PTR DS:[4031CC]
00401470	- FF25 D0314000	JMP DWORD PTR DS:[4031D0]
00401476	- FF25 D4314000	JMP DWORD PTR DS:[4031D4]
0040147C	- FF25 D8314000	JMP DWORD PTR DS:[4031D8]
00401482	- FF25 DC314000	JMP DWORD PTR DS:[4031DC]
00401488	- FF25 E0314000	JMP DWORD PTR DS:[4031E0]
0040148E	- FF25 E4314000	JMP DWORD PTR DS:[4031E4]
00401494	- FF25 E8314000	JMP DWORD PTR DS:[4031E8]
0040149A	- FF25 EC314000	JMP DWORD PTR DS:[4031EC]
004014A0	- FF25 F0314000	JMP DWORD PTR DS:[4031F0]
004014A6	- FF25 F4314000	JMP DWORD PTR DS:[4031F4]
004014AC	- FF25 F8314000	JMP DWORD PTR DS:[4031F8]
004014B2	- FF25 FC314000	JMP DWORD PTR DS:[4031FC]

DS:[004031AC]=77D504EA

Los JUMPS estan lo unico que falta es hacer funcionar el sistema de que llene la IAT con el contenido correcto, para que tenga en la misma antes de correr el programa, los punteros a las strings y que el sistema lo llene con las direcciones correctas sin error.

Antes de acometer esta tarea veremos las definiciones y ubicación correcta en el programa original sin empacar de la IAT y la IT.

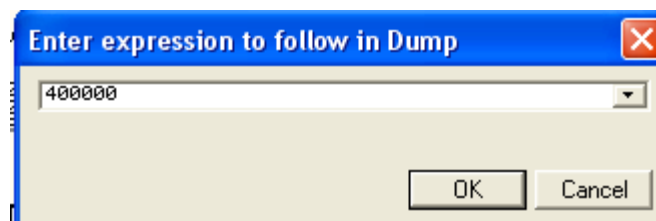
Abramos el CC original en OLLYDBG.



Bueno ubicaremos las partes que hacen que este sistema funcione bien en el original.

En el header, tenemos algunos punteros importantes.

Vayamos al mismo



CRACKME.<ModuleEntryPoint>			
Address	Hex dump	ASCII	
00400000	4D 5A 50 00 02 00 00 00	MZP.0...	
00400008	04 00 0F 00 FF FF 00 00	♦.*. ..	
00400010	B8 00 00 00 00 00 00 00	@.....	
00400018	40 00 1A 00 00 00 00 00	@.+.....	
00400020	00 00 00 00 00 00 00 00	
00400028	00 00 00 00 00 00 00 00	
00400030	00 00 00 00 00 00 00 00	
00400038	00 00 00 00 00 01 00 000..	
00400040	BA 10 00 0E 1F B4 09 CD	.A?+. =	
00400048	21 B8 01 4C CD 21 90 90	*00L=*EE	
00400050	54 68 69 73 20 70 72 6F	This pro	
00400058	67 72 61 6D 20 6D 75 73	gram mus	
00400060	74 20 62 65 20 72 75 6E	t be run	
00400068	20 75 6E 64 65 72 20 57	under W	
00400070	69 6E 33 32 0D 0A 24 37	in32...\$?	
00400078	00 00 00 00 00 00 00 00	
00400080	00 00 00 00 00 00 00 00	
00400088	00 00 00 00 00 00 00 00	
00400090	00 00 00 00 00 00 00 00	

Ahora cambiemoslo a modo SPECIAL-PE HEADER

CRACKME.<ModuleEntryPoint>			
Address	Hex dump	Data	Comment
00400000	4D 5A	ASCII "MZ"	DOS EXE Signature
00400002	5000	DW 0050	DOS_PartPag = 50 (80.)
00400004	0200	DW 0002	DOS_PageCnt = 2
00400006	0000	DW 0000	DOS_ReloCnt = 0
00400008	0400	DW 0004	DOS_HdrSize = 4
0040000A	0F00	DW 000F	DOS_MinMem = F (15.)
0040000C	FFFF	DW FFFF	DOS_MaxMem = FFFF (65535.)
0040000E	0000	DW 0000	DOS_ReloSS = 0
00400010	B800	DW 00B8	DOS_ExeSP = B8
00400012	0000	DW 0000	DOS_ChkSum = 0
00400014	0000	DW 0000	DOS_ExeIP = 0
00400016	0000	DW 0000	DOS_ReloCS = 0
00400018	4000	DW 0040	DOS_TableOff = 40
0040001A	1A00	DW 001A	DOS_Overlay = 1A
0040001C	00	DB 00	
0040001D	00	DB 00	
0040001E	00	DB 00	
0040001F	00	DB 00	

Bajemos

Address	Hex dump	Data	Comment
00400035	00	DB 00	
00400036	00	DB 00	
00400037	00	DB 00	
00400038	00	DB 00	
00400039	00	DB 00	
0040003A	00	DB 00	
0040003B	00	DB 00	
0040003C	00010000	DD 00000100	Offset to PE signature
00400040	BA	DB BA	
00400041	10	DB 10	
00400042	00	DB 00	
00400043	0E	DB 0E	
00400044	1F	DB 1F	
00400045	B4	DB B4	
00400046	09	DB 09	
00400047	00	DB 00	

Los datos realmente empiezan en 100 donde comienza la PE SIGNATURE

Address	Hex dump	Data	Comment
004000F6	00	DB 00	
004000F7	00	DB 00	
004000F8	00	DB 00	
004000F9	00	DB 00	
004000FA	00	DB 00	
004000FB	00	DB 00	
004000FC	00	DB 00	
004000FD	00	DB 00	
004000FE	00	DB 00	
004000FF	00	DB 00	
00400100	50 45 00 00	ASCII "PE"	PE signature (PE)
00400104	4C01	DW 014C	Machine = IMAGE_FILE_MACHINE_I386
00400106	0600	DW 0006	NumberOfSections = 6
00400108	2924D90A	DD 0AD92429	TimeDateStamp = AD92429
0040010C	00000000	DD 00000000	PointerToSymbolTable = 0
00400110	00000000	DD 00000000	NumberOfSymbols = 0
00400114	E000	DW 00E0	SizeOfOptionalHeader = E0 (224.)
00400116	8E81	DW 818E	Characteristics = EXECUTABLE_IMAGE 32BIT_
00400118	0B01	DW 010B	MagicNumber = PE32
0040011A	02	DB 02	MajorLinkerVersion = 2
0040011B	19	DB 19	MinorLinkerVersion = 19 (25.)
0040011C	00060000	DD 00000600	SizeOfCode = 600 (1536.)
00400120	00220000	DD 00002200	SizeOfInitializedData = 2200 (8704.)
00400124	00000000	DD 00000000	SizeOfUninitializedData = 0
00400128	00100000	DD 00001000	AddressOfEntryPoint = 1000
0040012C	00100000	DD 00001000	BaseOfCode = 1000
00400130	00200000	DD 00002000	BaseOfData = 2000
00400134	00040000	DD 00000400	ImageBase = 400000

El chico no miente alli empieza jeje en 400100 o sea en offset 100.

0040017C	46000000	DD 00000046	Export Table size = 46 (70.)
00400180	00300000	DD 00003000	Import Table address = 3000
00400184	70060000	DD 00000670	Import Table size = 670 (1648.)
00400188	00600000	DD 00006000	Resource Table address = 6000
0040018C	00140000	DD 00001400	Resource Table size = 1400 (5120.)

alli tenemos los punteros a la IT o sea la Import Table que no debemos confundirla con la IAT a pesar de que tengan nombres parecidos.

IT= IMPORT TABLE

IAT=IMPORT ADDRESS TABLE

Como ya vimos la IAT es el deposito donde el sistema guarda las direcciones correctas de mi maquina al arrancar, y que es la IT, pues vayamos alli, como vemos dice que empieza en 3000 (403000) y su largo es 670 o sea que termina en 403670, vayamos a mirar.

Como esta fuera del header quitamos el MODO SPECIAL-PE HEADER

CRACKME.<ModuleEntryPoint>												
Address	Hex dump										ASCII	
00403000	78	30	00	00	00	00	00	00	00	90	32	00 00
00403010	84	31	00	00	10	31	00	00	00	00	00	00
00403020	9B	32	00	00	1C	32	00	00	3C	31	00	00
00403030	00	00	00	00	A8	32	00	00	48	32	00	00
00403040	00	00	00	00	00	00	00	00	B5	32	00	00
00403050	74	31	00	00	00	00	00	00	00	00	00	00
00403060	80	32	00	00	00	00	00	00	00	00	00	00
00403070	00	00	00	00	00	00	00	00	CC	32	00	00
00403080	EC	32	00	00	FA	32	00	00	0E	33	00	00
00403090	2C	33	00	00	3A	33	00	00	46	33	00	00
004030A0	60	33	00	00	6E	33	00	00	80	33	00	00
004030B0	9E	33	00	00	B6	33	00	00	C4	33	00	00
004030C0	E4	33	00	00	F2	33	00	00	02	34	00	00
004030D0	1E	34	00	00	2E	34	00	00	40	34	00	00
004030E0	60	34	00	00	72	34	00	00	84	34	00	00
004030F0	A6	34	00	00	B2	34	00	00	BE	34	00	00
00403100	D4	34	00	00	E2	34	00	00	F4	34	00	00
00403110	02	35	00	00	12	35	00	00	1E	35	00	00
00403120	3A	35	00	00	44	35	00	00	52	35	00	00
00403130	72	35	00	00	7E	35	00	00	00	00	00	00
00403140	A2	35	00	00	B4	35	00	00	00	00	00	00
00403150	D0	35	00	00	DC	35	00	00	E8	35	00	00
00403160	0C	36	00	00	16	36	00	00	20	36	00	00
00403170	00	00	00	00	3C	36	00	00	50	36	00	00
00403180	00	00	00	00	42	8C	D1	77	9D	8F	D1	77
00403190	24	15	D3	77	4C	1F	D3	77	D4	B6	D1	77
004031A0	24	13	D2	77	DA	5E	D2	77	60	DA	D1	77
004031B0	11	12	02	77	35	EE	D3	77	F5	B5	D1	77
004031C0	EC	D8	D1	77	F6	88	D1	77	83	F7	D4	77
004031D0	9A	F3	D2	77	2E	8C	D1	77	1B	C0	D1	77
004031E0	8C	14	D2	77	09	B6	D1	77	5E	02	D2	77
004031F0	1C	B1	D3	77	B8	96	D1	77	9C	F3	D4	77
00403200	1D	B6	D1	77	81	E5	D2	77	C7	86	D1	77
00403210	1E	AC	D6	77	42	10	D2	77	00	00	00	00
00403220	99	68	82	7C	2F	FE	80	7C	2D	FF	80	7C
00403230	77	9B	80	7C	9F	0F	81	7C	29	B5	80	7C
00403240	A2	CA	81	7C	00	00	00	00	DD	15	C5	58
00403250	3B	8B	C6	58	00	00	00	00	0C	BC	EF	77
00403260	E9	49	F2	77	68	E0	EF	77	E1	61	EF	77
00403270	D1	E0	F0	77	2D	6C	EF	77	98	6E	EF	77
00403280	58	7C	37	76	1E	31	36	76	CD	46	38	76
00403290	55	53	45	52	33	32	2E	64	6C	6C	00	4B
004032A0	4C	33	32	2E	64	6C	6C	00	43	4F	4D	43
004032B0	2E	44	4C	4C	00	47	44	49	33	32	2E	64
004032C0	4F	4D	44	4C	47	33	32	2E	64	6C	6C	00
004032D0	6C	6C	54	69	6D	65	72	00	00	00	47	65
004032E0	74	65	6D	4D	65	74	72	69	63	73	00	00
004032F0	61	64	43	75	72	73	6F	72	41	00	00	00
00403300	41	63	63	65	6C	65	72	61	74	6F	72	73
00403310	4D	65	73	73	61	67	65	42	65	65	70	00
00403320	70	57	69	6E	64	6F	77	52	65	63	74	00
00403330	61	64	53	74	72	69	6E	67	41	00	00	00
00403340	49	63	6F	6E	41	00	00	00	4C	6F	61	64
00403350	61	70	41	00	00	00	53	65	74	46	6F	63
00403360	00	00	4D	65	73	73	61	67	65	42	6F	78
00403370	50	6F	73	74	51	75	69	74	4D	65	73	73
00403380	00	00	57	69	6E	48	65	6C	70	41	00	00

Esto que parece un chorizo sin interpretacion posible, es facil de interpretar, si sabemos que es cada cosa, lo detallaremos y veran que esto es muy importante.

Esto es la famosa IT, aquí la tenemos si comprendemos esto, y aunque no sepamos los nombres de cada puntero, sepamos que es cada cosa y donde apunta, pues estamos salvados.

Lo malo que tenemos para explicar esto en OLLYDBG es que es mucho mas comodo y visible hacerlo en una visualizacion de 5 columnas asi queda ordenado, ya veremos por que.

La famosa IT esta compuesta por los denominados IMAGE_IMPORT_DESCRIPTORs que son varias lineas de 5 valores dword, de la cual hay una por cada dll que cargara el programa.

Veamos la IT, por eso les digo que esto se ve mejor con 5 columnas para ordenar cada IID uno en cada linea.

Pero bueno no nos queda otra aquí vemos el primero

CRACKME.<ModuleEntryPoint>																	
Address	Hex dump												ASCII				
00403000	78	30	00	00	00	00	00	00	00	00	90	32	00	00	x0.....e2..		
00403010	84	31	00	00	10	31	00	00	00	00	00	00	00	00	a1..1.....		
00403020	9B	32	00	00	1C	32	00	00	3C	31	00	00	00	00	2...2...<1....		
00403030	00	00	00	00	A8	32	00	00	48	32	00	00	4C	31	00	...2...H2...L1..	
00403040	00	00	00	00	00	00	00	00	B5	32	00	00	58	32	00A2...X2..	
00403050	74	31	00	00	00	00	00	00	00	00	00	00	BF	32	00	t1.....t2..	
00403060	80	32	00	00	00	00	00	00	00	00	00	00	00	00	00	2.....	
00403070	00	00	00	00	00	00	00	00	CC	32	00	00	08	32	00t2...i2..	
00403080	EC	32	00	00	FA	32	00	00	0E	33	00	00	1C	33	00	y2...2...#3...L3..	
00403090	2C	33	00	00	3A	33	00	00	46	33	00	00	54	33	00	3...3...F3...T3..	
004030A0	60	33	00	00	6E	33	00	00	80	33	00	00	8C	33	00	3...n3...C3...i3..	
004030B0	9E	33	00	00	B6	33	00	00	C4	33	00	00	D8	33	00	x3...3...-3...i3..	
004030C0	E4	33	00	00	F2	33	00	00	02	34	00	00	0E	34	00	33...=3...04...#4..	

Ese es el primer IID de nuestra IT y tiene 5 DWORDS cuyos nombres son estos

- OriginalFirtsThunks.
- TimeDateStamp.
- ForwarderChain.
- Puntero al nombre de la dll.
- FirtsThunk (Apunta a donde debe buscar en la IAT la primera entrada de dicha dll).

Los tres primeros no son importantes para el cracking hasta siendo cero, normalmente arranca igual el programa, son para usos muy determinados, los importantes son el 4 y el 5to puntero de nuestro primer IID.

Address	Hex dump	ASCII
00403000	78 30 00 00 00 00 00 00 00 00 00 00 00 90 32 00 00	x0.....
00403010	84 31 00 00 10 31 00 00 00 00 00 00 00 00 00 00	31...1.....
00403020	9B 32 00 00 1C 32 00 00 3C 31 00 00 00 00 00 00	2...2...<1....
00403030	00 00 00 00 A8 32 00 00 48 32 00 00 4C 31 00 00	...2...H2...L1..
00403040	00 00 00 00 00 00 00 00 B5 32 00 00 58 32 00 00A2...X2..
00403050	74 31 00 00 00 00 00 00 00 00 00 00 BF 32 00 00	t1.....t2..

Alli esta como vemos el 4to apunta al nombre de la dll a la cual pertenece este IID, veamos cual es la dll en 403290.

00403250	3B 8B C6 58 00 00 00 00 0C BC EF 77 26 F1 F0 77	i&X....'w&t-u
00403260	E9 49 F2 77 68 E0 EF 77 E1 61 EF 77 C9 DD F0 77	0i=wh0'wBa'wfi-w
00403270	51 E0 F0 77 20 6C EF 77 98 6E EF 77 00 00 00 00	Q0-w-l'wyn'w....
00403280	08 7C 37 76 1E 31 36 76 CD 46 38 76 00 00 00 00	i!7v16v=F8v....
00403290	55 53 45 52 33 32 2E 64 6C 6C 00 4B 45 52 4E 45	USER32.dll KERNE
004032A0	4C 33 32 2E 64 6C 6C 00 43 4F 4D 43 54 4C 33 32	L32.dll.COMCTL32
004032B0	2E 44 4C 4C 00 47 44 49 33 32 2E 64 6C 6C 00 43	.DLL.GDI32.dll.C
004032C0	4F 4D 44 4C 47 33 32 2E 64 6C 6C 00 00 00 4B 69	OMDLG32.dll...Ki
004032D0	6C 6C 54 69 60 65 72 00 00 00 47 65 74 53 79 73	llTimer...GetSys
004032E0	74 65 6D 4D 65 74 72 69 63 73 00 00 00 00 4C 6F	temMetrics....Lo
004032F0	61 64 43 75 72 73 6F 72 41 00 00 00 4C 6F 61 64	adCursorA...Load
00403300	41 63 63 65 6C 65 72 61 74 6F 72 73 41 00 00 00	AcceleratorsA...

Alli esta la primera dll que el sistema buscara es USER32.dll y el 5to puntero marca donde comienza en la IAT las entradas de esta dll, en este caso es 403184.

CRACKME.<ModuleEntryPoint>																		
Address	Hex dump												ASCII					
00403184	42	8C	D1	77	9D	8F	D1	77	3E	0B	D2	77	24	15	D3	77	B10w0A0w>0Ew\$S0w	
00403194	4C	1F	D3	77	D4	B6	D1	77	E8	0F	D2	77	24	13	D2	77	L7EwE0A0w*Ew\$Ew	
004031A4	DA	5E	D2	77	60	DA	D1	77	EA	04	D5	77	11	12	D2	77	r^Ew'r0w0'w4Ew	
004031B4	35	EE	D3	77	F5	B5	D1	77	9C	FA	D2	77	EC	DB	D1	77	S^Ew\$A0wE^Ew\$0w	
004031C4	F6	8B	D1	77	83	F7	D4	77	A4	D8	D1	77	9A	F3	D2	77	÷i0w0^Ew0i0w0Ew	
004031D4	2E	8C	D1	77	1B	C0	D1	77	F9	D7	D1	77	8C	14	D2	77	.i0w+L0w-i0wi0Ew	
004031E4	09	B6	D1	77	5E	02	D2	77	EE	D4	D1	77	1C	B1	D3	77	.A0w^0Ew^E0w\$Ew	
004031F4	B8	96	D1	77	9C	F3	D4	77	50	62	D2	77	1D	B6	D1	77	000w0EwPbEw#A0w	
00403204	81	E5	D2	77	C7	86	D1	77	16	48	D2	77	1E	AC	D6	77	0E0w\$A0w.HEwA0iw	
00403214	42	10	D2	77	00	00	00	00	C1	C9	80	7C	99	68	82	7C	B0Ew...+fC!0kE!	
00403224	2F	FE	80	7C	2D	FF	80	7C	E0	C6	80	7C	77	9B	80	7C	/mC!-C!0A0!w0C!	
00403234	9F	0F	81	7C	29	B5	80	7C	0E	18	80	7C	A2	CA	81	7C	f#u!A0!0A0!0A0!	
00403244	00	00	00	00	DD	15	C5	58	21	9B	C4	58	3B	8B	C6	58!S+X!-X!i&X	
00403254	00	00	00	00	0C	BC	EF	77	26	F1	F0	77	E9	49	F2	77'w&t-w0I=w	
00403264	68	E0	EF	77	E1	61	EF	77	C9	DD	F0	77	51	E0	F0	77	h0'wBa'wfi-w00-w	

Alli esta es la IAT, y su primera entrada, todo esto esta dentro de la IT, ya que terminaba en 403670, asi que tenemos que la famosa IT, tiene una linea para cada dll, estas lineas son llamadas IID o IMAGE IMPORT DESCRIPTOR y dentro de la IT esta la IAT o deposito de las direcciones de las apis, todo compacto y para

el uso del sistema.

Muchos crackres experimentados diran que no siempre la IAT esta dentro de la IT, ya que el 5to puntero de cada IID puede apuntar a cualquier lado y tranquilamente puede estar fuera, ubicada en cualquier lugar del programa que tenga permiso de escritura, para que cuando arranque el exe, alli el sistema vaya guardando las direcciones correctas de las apis, la cuestion es que ya vemos como trabaja mejor el sistema para llenar la IAT,

- 1) Busca la IT
- 2) Busca la primera IID y se fija en el 4to puntero a que dll pertenece
- 3) Luego mira en el 5to puntero donde esta la primera entrada de la IAT
- 4) Alli hay un puntero a la string con el nombre de la api
- 5) Con GetProcAddress busca la direccion y la guarda en la misma entrada.
- 6) Cuando encuentra en la IAT una entrada con ceros, eso le indica que termino la primera dll y que debe pasar a mirar el segundo IID para buscar cual es la segunda, y repetir el proceso.

O sea que si el sistema hara esto en imagenes y es importante que lo entiendan bien porque si al sistema le falta algo o encuentra punteros incorrectos dara error y es importante entenderlo a fondo.

1)Busca la direccion de la IT

0040017C	46000000	DD 00000046	Export Table size = 46 (70.)
00400180	00300000	DD 00003000	Import Table address = 3000
00400184	70060000	DD 00000670	Import Table size = 670 (1648.)
00400188	00600000	DD 00006000	Resource Table address = 6000

2)Va a esa direccion

CRACKME.<ModuleEntryPoint>			
Address	Hex dump	ASCII	
00403000	78 30 00 00 00 00 00 00 00 00 00 00 90 32 00 00	x0.....E2..	
00403010	84 31 00 00 10 31 00 00 00 00 00 00 00 00 00 00	ä1..1.....	
00403020	98 32 00 00 1C 32 00 00 3C 31 00 00 00 00 00 00	2..L2.<1.....	
00403030	00 00 00 00 A8 32 00 00 48 32 00 00 4C 31 00 00	...ä2..H2..L1..	
00403040	00 00 00 00 00 00 00 00 B5 32 00 00 58 32 00 00A2..X2..	
00403050	74 31 00 00 00 00 00 00 00 00 00 00 BF 32 00 00	t1.....72..	
00403060	80 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00	2.....f2..i2..	
00403070	00 00 00 00 00 00 00 00 CC 32 00 00 D8 32 00 002...#3..L3..	
00403080	EC 32 00 00 FA 32 00 00 0E 33 00 00 1C 33 00 00	2...2...F3..T3..	
00403090	2C 33 00 00 3A 33 00 00 46 33 00 00 54 33 00 00	3...3...C3..I3..	
004030A0	60 33 00 00 6E 33 00 00 80 33 00 00 8C 33 00 00	3...3...C3..I3..	
004030B0	00 33 00 00 00 33 00 00 00 33 00 00 00 33 00 00	3...3...C3..I3..	

3)En el primer IID busca el 4to DWORD que le apunta al nombre de la dll donde empezara a trabajar en este caso sera USER32.dll.

CRACKME.<ModuleEntryPoint>			
Address	Hex dump	ASCII	
00403290	55 53 45 52 33 32 2E 64 6C 6C 00 4B 45 52 4E 45	USER32.dll.KERNE	
004032A0	4C 33 32 2E 64 6C 6C 00 43 4F 4D 43 54 4C 33 32	L32.dll.COMCTL32	
004032B0	2E 44 4C 4C 00 47 44 49 33 32 2E 64 6C 6C 00 43	.DLL.GDI32.dll.C	
004032C0	4F 4D 44 4C 47 33 32 2E 64 6C 6C 00 00 00 4B 69	OMDLG32.dll...Ki	
004032D0	6C 6C 54 69 6D 65 72 00 00 00 47 65 74 53 79 73	llTimer...GetSys	
004032E0	74 65 6D 4D 65 74 72 69 63 73 00 00 00 00 4C 6F	temMetrics...Lo	
004032F0	61 64 43 75 72 73 6F 72 41 00 00 00 4C 6F 61 64	adCursorA...Load	
00403300	41 63 63 65 6C 65 72 61 74 6F 72 73 41 00 00 00	AcceleratorsA...	
00403310	4D 65 73 73 61 67 65 42 65 65 70 00 00 00 47 65	MessageBeep...Ge	
00403320	74 57 69 6E 64 6F 77 52 65 63 74 00 00 00 4C 6F	tWindowRect...Lo	
00403330	61 64 53 74 72 69 6E 67 41 00 00 00 4C 6F 61 64	adStringA...Load	
00403340	49 63 6F 6E 41 00 00 00 4C 6F 61 64 42 69 74 6D	IconA...LoadBitm	
00403350	61 70 41 00 00 00 53 65 74 46 6F 63 75 73 00 00	apA...SetFocus...	
00403360	00 00 4D 65 73 73 61 67 65 42 6F 78 41 00 00 00	..MessageBoxA...	
00403370	50 6F 73 74 51 75 69 74 4D 65 73 73 61 67 65 00	PostQuitMessage.	

Luego mira el 5to puntero para buscar la primera entrada de la IAT que es 403184 y va alli.

CRACKME.<ModuleEntryPoint>											
Address	Hex dump										ASCII
00403000	78	30	00	00	00	00	00	00	00	90	32 00 00
00403010	84	31	00	00	10	31	00	00	00	00	00 00 00
00403020	9B	32	00	00	10	32	00	00	3C	31 00 00	00 00 00 00

Address	Hex dump										ASCII
00403184	42	8C	D1	77	9D	8F	D1	77	3E	0B	D2 77 24 15 03 77
00403194	4C	1F	03	77	04	B6	D1	77	E8	0F	D2 77 24 13 02 77
004031A4	DA	5E	02	77	60	DA	D1	77	EA	04	05 77 11 12 02 77
004031B4	35	EE	03	77	F5	B5	D1	77	9C	FA	D2 77 EC 0B 01 77
004031C4	F6	8B	D1	77	83	F7	04	77	A4	08	D1 77 9A F3 D2 77
004031D4	2E	8C	D1	77	1B	C0	D1	77	F9	07	D1 77 8C 14 D2 77
004031E4	09	B6	D1	77	5E	02	D2	77	EE	D4	D1 77 1C B1 03 77
004031F4	B8	96	D1	77	9C	F3	04	77	50	62	D2 77 1D B6 D1 77
00403204	81	E5	D2	77	C7	86	D1	77	16	48	D2 77 1E AC D6 77
00403214	42	10	D2	77	00	00	00	00	C1	C9	80 7C 99 6B 82 7C

Alli lee el valor que no es ese pues alli ya esta machacado, sino el que esta en el ejecutable, veamos

File C:\Documents and Settings\Ricardo\Escritorio\32-IN...											
00000F84	CC	32	00	00	D8	32	00	00	EC	32	00 00 FA 32 00 00
00000F94	0E	33	00	00	1C	33	00	00	2C	33	00 00 3A 33 00 00
00000FA4	46	33	00	00	54	33	00	00	60	33	00 00 6E 33 00 00
00000FB4	80	33	00	00	8C	33	00	00	9E	33	00 00 B6 33 00 00
00000FC4	C4	33	00	00	D8	33	00	00	E4	33	00 00 F2 33 00 00

alli ve que el nombre de la primera api esta en 4032CC, va alli

CRACKME.<ModuleEntryPoint>											
Address	Hex dump										ASCII
004032CC	00	00	4B	69	6C	6C	54	69	6D	65	72 00 00 00 47 65
004032DC	74	53	79	73	74	65	6D	40	65	74	72 69 63 73 00 00
004032EC	00	00	4C	6F	61	64	43	75	72	73	6F 72 41 00 00 00
004032FC	4C	6F	61	64	41	63	63	65	6C	65	72 61 74 6F 72 73
0040330C	41	00	00	00	40	65	73	73	61	67	65 42 65 65 70 00
0040331C	00	00	47	65	74	57	69	6E	64	6F	77 52 65 63 74 00
0040332C	00	00	4C	6F	61	64	53	74	72	69	6E 67 41 00 00 00
0040333C	4C	6F	61	64	49	63	6F	6E	41	00	00 00 4C 6F 61 64
0040334C	42	69	74	6D	61	70	41	00	00	00	53 65 74 46 6F 63
0040335C	75	73	00	00	00	00	4D	65	73	73	61 67 65 42 6F 78

Ve que la primera api es KillTimer con GetProcAddress halla su direccion que nosotros hallamos con el OLLYDBG.

0040351C	00	00	00	00	47	6C	6F	62	61	6C	46	72	65	65	00	00	...	G1
Command ? KillTimer										HEX: 77D18C42 - DE								
Program entry point																		

Y ese valor que en mi maquina es 77d18c42 lo guarda en la misma entrada de la IAT machacando el existente.

CRACKME.<ModuleEntryPoint>											
Address	Hex dump										ASCII
00403184	42	8C	D1	77	9D	8F	D1	77	3E	0B	D2 77 24 15 03 77
00403194	4C	1F	03	77	04	B6	D1	77	E8	0F	D2 77 24 13 02 77
004031A4	DA	5E	02	77	60	DA	D1	77	EA	04	05 77 11 12 02 77
004031B4	35	EE	03	77	F5	B5	D1	77	9C	FA	D2 77 EC 0B 01 77
004031C4	F6	8B	D1	77	83	F7	04	77	A4	08	D1 77 9A F3 D2 77
004031D4	2E	8C	D1	77	1B	C0	D1	77	F9	07	D1 77 8C 14 D2 77
004031E4	09	B6	D1	77	5E	02	D2	77	EE	D4	D1 77 1C B1 03 77

Alli lo vemos es la primera entrada de la IAT ya cuando llego al entry point y muestra el valor que tiene esa api en mi maquina.

Asi saltara a la siguiente entrada de la IAT, que halla facilmente sumandole 4 a esta, y busca el siguiente puntero al nombre de la proxima api.

CRACKME.<ModuleEntryPoint>															
Address	Hex dump												ASCII		
00403184	42	8C	D1	77	9D	8F	D1	77	3E	0B	D2	77	24	15	D3 77
00403194	4C	1F	D3	77	04	B6	D1	77	E8	0F	D2	77	24	13	D2 77
004031A4	DA	5E	D2	77	60	DA	D1	77	EA	04	D5	77	11	12	D2 77
004031B4	35	EE	D3	77	F5	B5	D1	77	9C	FA	D2	77	EC	0B	D1 77
004031C4	F6	8B	D1	77	83	F7	D4	77	A4	D8	D1	77	9A	F3	D2 77
004031D4	2E	8C	D1	77	1B	C0	D1	77	F9	D7	D1	77	8C	14	D2 77
004031E4	09	86	D1	77	5F	82	D2	77	FF	04	D1	77	1C	R1	D3 77

Por supuesto mira en el ejecutable y este es

D File C:\Documents and Settings\Ricardo\Escritorio\32-IN...															
Address	Hex dump												ASCII		
00000F88	08	32	00	00	EC	32	00	00	FA	32	00	00	0E	33	00 00
00000F98	1C	33	00	00	2C	33	00	00	3A	33	00	00	46	33	00 00
00000FA8	54	33	00	00	60	33	00	00	6E	33	00	00	80	33	00 00
00000FB8	8C	33	00	00	9E	33	00	00	B6	33	00	00	C4	33	00 00
00000FC8	D8	33	00	00	E4	33	00	00	F2	33	00	00	02	34	00 00
00000FD8	0E	34	00	00	1E	34	00	00	2E	34	00	00	40	34	00 00
00000FE8	4E	34	00	00	60	34	00	00	72	34	00	00	84	34	00 00
00000FF8	98	34	00	00	A6	34	00	00	B2	34	00	00	BE	34	00 00
00001008	CC	34	00	00	D4	34	00	00	E2	34	00	00	F4	34	00 00
00001018	00	00	00	00	02	35	00	00	12	35	00	00	1F	35	00 00

32d8 o sea 4032d8, mira el nombre de la api siguiente alli y como no hallo un cero en la entrada, sabe que continua con las apis de user32.dll.

Address	Hex dump												ASCII		
004032D8	00	00	47	65	74	53	79	73	74	65	6D	4D	65	74	72 69
004032E8	63	73	00	00	00	00	4C	6F	61	64	43	75	72	73	6F 72
004032F8	41	00	00	00	4C	6F	61	64	41	63	63	65	6C	65	72 61
00403308	74	6F	72	73	41	00	00	00	4D	65	73	73	61	67	65 42
00403318	65	65	70	00	00	00	47	65	74	57	69	6E	64	6F	77 52
00403328	65	63	74	00	00	00	4C	6F	61	64	53	74	72	69	6E 67
00403338	41	00	00	00	4C	6F	61	64	49	63	6F	6E	41	00	00 00

La segunda api es GetSystemMetrics hallara la direccion y la guardara en la segunda entrada de la iat y asi, seguira buscando apis en user32.dll hasta que llegue a una entrada con todos ceros si vemos la iat en el ejecutable.

Address	Hex dump												ASCII		
00000F88	08	32	00	00	EC	32	00	00	FA	32	00	00	0E	33	00 00
00000F98	1C	33	00	00	2C	33	00	00	3A	33	00	00	46	33	00 00
00000FA8	54	33	00	00	60	33	00	00	6E	33	00	00	80	33	00 00
00000FB8	8C	33	00	00	9E	33	00	00	B6	33	00	00	C4	33	00 00
00000FC8	D8	33	00	00	E4	33	00	00	F2	33	00	00	02	34	00 00
00000FD8	0E	34	00	00	1E	34	00	00	2E	34	00	00	40	34	00 00
00000FE8	4E	34	00	00	60	34	00	00	72	34	00	00	84	34	00 00
00000FF8	98	34	00	00	A6	34	00	00	B2	34	00	00	BE	34	00 00
00001008	CC	34	00	00	D4	34	00	00	E2	34	00	00	F4	34	00 00
00001018	00	00	00	00	02	35	00	00	12	35	00	00	1E	35	00 00
00001028	2C	35	00	00	3A	35	00	00	44	35	00	00	52	35	00 00
00001038	5E	35	00	00	72	35	00	00	7E	35	00	00	8C	35	00 00
00001048	8C	35	00	00	A2	35	00	00	B4	35	00	00	00	00	00 00
00001058	C4	35	00	00	D0	35	00	00	DC	35	00	00	E8	35	00 00
00001068	FA	35	00	00	0C	36	00	00	16	36	00	00	20	36	00 00
00001078	30	36	00	00	00	00	00	00	3C	36	00	00	50	36	00 00
00001088	64	36	00	00	00	00	00	00	55	53	45	52	33	32	2E 64
00001098	6C	6C	00	4B	45	52	4C	45	4C	33	32	2E	64	6C	6C 00
000010A8	43	4F	4D	43	54	4C	33	32	2E	44	4C	4C	00	47	44 49

Vemos que seguira buscando en user32.dll hasta que halle ese cero donde retorna al siguiente IID

Address	Hex dump												ASCII		
00403000	78	30	00	00	00	00	00	00	00	00	00	00	90	32	00 00
00403010	84	31	00	00	10	31	00	00	00	00	00	00	00	00	00 00
00403020	9B	32	00	00	1C	32	00	00	3C	31	00	00	00	00	00 00
00403030	00	00	00	00	A8	32	00	00	48	32	00	00	4C	31	00 00
00403040	00	00	00	00	00	00	00	00	B5	32	00	00	58	32	00 00
00403050	74	33	00	00	00	00	00	00	00	00	00	00	BF	32	00 00
00403060	80	32	00	00	00	00	00	00	00	00	00	00	00	00	00 00
00403070	00	00	00	00	00	00	00	00	CC	32	00	00	00	32	00 00

Cuyos 4 y 5to puntero son esos, el 4to le dice que en 40329b esta el nombre de la 2 dll donde buscara nombres.

CRACKME.<ModuleEntryPoint>												
Address	Hex dump											
	ASCII											
0040329B	4B	45	52	4E	45	4C	33	32	2E	64	6C	6C
0040329C	00	43	4F	4D	K	E	R	N	E	L	3	2
0040329D	43	54	4C	33	32	2E	44	4C	4C	00	47	44
0040329E	49	33	32	2E	C	T	L	3	2	D	L	L
0040329F	64	6C	6C	00	43	4F	4D	44	4C	47	33	32
004032A0	2E	64	6C	6C	d	l	l	.	C	O	M	D
004032A1	00	00	00	4B	69	6C	6C	54	69	6D	65	72
004032A2	00	00	00	47	...	K	i	l	l	T	i	m
004032A3	65	74	53	79	73	74	65	6D	4D	65	74	72
004032A4	69	63	73	00	e	t	S	y	s	t	e	m
004032A5	63	00	00	4C	7F	74	74	43	7F	73	73	7F
004032A6	73	74	43	43	7F	73	73	7F	73	74	63	63

Que es Kernel32.dll y la primera entrada de la iat donde buscara sera la que esta a continuacion del cero o sea 40321c.

Address	Hex dump											
	ASCII											
0040320C	16	48	D2	77	1E	AC	D6	77	42	10	D2	77
0040320D	00	00	00	00	..	H	e	w	...			
0040320E	C1	C9	80	7C	99	6B	82	7C	2F	FE	80	7C
0040320F	2D	FF	80	7C	+	f	c	i	o	k	e	!
00403210	E0	C6	80	7C	77	9B	80	7C	9F	0F	81	7C
00403211	29	B5	80	7C	0	g	c	!	w	o	!	f
00403212	0E	18	80	7C	A2	CA	81	7C	00	00	00	00
00403213	00	00	00	00	00	00	00	00	00	00	00	00
00403214	21	9B	C4	58	3B	8B	C6	58	00	00	00	00
00403215	26	F1	F0	77	E9	49	F2	77	68	E0	EF	77
00403216	C9	0D	F0	77	51	E0	F0	77	2D	6C	EF	77
00403217	00	00	00	00	08	7C	37	76	1E	31	36	76
00403218	00	00	00	00	55	53	45	52	33	32	2E	64
00403219	6C	6C	6C	00	4B	...						

Alli vemos el cero que determino que no hay mas apis de user32.dll y en 40321c empiezan las de Kernel32.dll donde buscara en el ejecutable el primer puntero a la string y llenara aquí con el valor de la api correcta.

Creo que es muy importante entender bien esto, trate de explicarlo y repetirlo varias veces para que quede pero creo que con solo una leida eso no queda en la mente traten de grabarselo con fuego pues esto es la base de todo el tema de la reconstruccion de las IATs y aunque ya veremos en la parte siguiente que hay tools que nos ayudan a hacer el trabajo pesado sin tener que revisar todo esto, es muy importante saber como trabaja todo, asi cualquier problema o caso raro, no nos quedaremos a un costado del camino y siempre podremos entender que esta pasando.

En la parte siguiente arreglaremos IATs no lloren jeje ya tienen suficiente con comprender como trabajan por ahora jejejejeje.

Hasta la 34
 Riardo Narvaja
 27/02/06