



Archivo: Croc-a-doc v. 1.02.



Herramientas: OllySND, LordPE e ImpRec.



SND



LORDPE



IMP REC

Objetivo: Desempacar.

Dificultad: La decide el lector.

Cracker: Ivinson.

Tutorial No: 3

ÍNDICE

	Pag.
▪ Introducción.....	3
▪ Instalación.....	4
▪ Comprobando posible “packer”.....	5
▪ Buscando el OEP.	7
▪ Buscando la IAT.....	9
▪ Reparando la IAT a mano.....	12
▪ Reparando el “header” (cabecera).....	21
▪ Dumpeando.....	23
▪ Reparando el Dumpeado.....	24
▪ Analizando la forma de registro.....	26
▪ Parchando.....	30
▪ Conclusión.....	33
▪ Agradecimientos.....	33
▪ Frase.....	33

INTRODUCCIÓN

Armadillo es uno de los protectores con los que no me los he llevado muy bien, pero ya veo que solo es cuestión de práctica como todo en la vida. En muchas ocasiones, traté de realizar el tutorial de Armadillo de las versiones viejas con varias protecciones y tuve problemas. Pero en este caso, trabajaremos con algo “light”; excelente para los que estamos comenzando y queremos ir poco a poco. El siguiente programa estaba en la carpeta “los que me quitaron mucho sueño y nada”; por cierto, siempre me acordaba y decía, “vas a pagar”. Hasta que logré quitarle la concha (caparazón) a este animal.

Concepto según el Diccionario Encarta 2009.

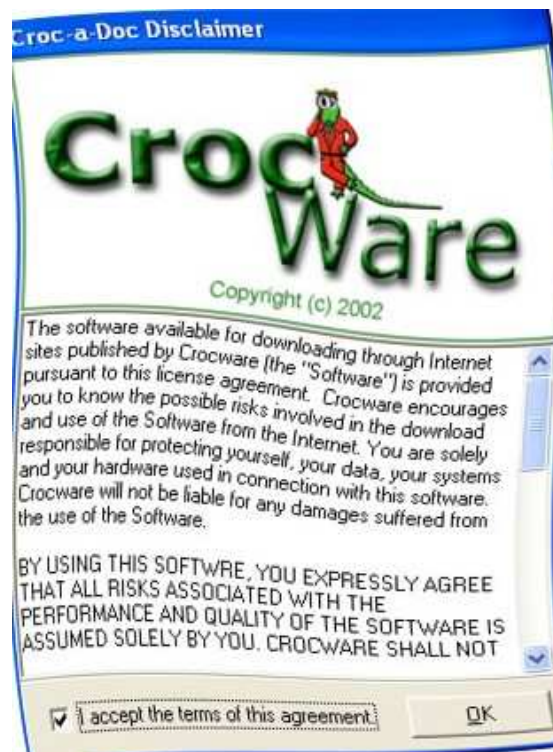
“**Armadillo.** (De *armado*). *m.* Mamífero del orden de los Desdentados, con algunos dientes laterales. El cuerpo, que mide de tres a cinco decímetros de longitud, está protegido por un caparazón formado de placas óseas cubiertas por escamas córneas, las cuales son movibles, de modo que el animal puede arrollarse sobre sí mismo. Todas las especies son propias de América Meridional.”

Microsoft® Encarta® 2009. © 1993-2008 Microsoft Corporation. Reservados todos los derechos.



INSTALACIÓN

No requiere de una instalación standard; solamente aceptan los terminos y listo como se muestra en la imagen a continuación.



Descarga Croc-a-doc v1.02: <http://www.mediafire.com/?8kkxv61g83arg15>

Usaremos el plugin HideDebugger con todas las opciones.

COMPROBANDO POSIBLE “PACKER”

El ejecutable no crea ninguna carpeta, sino que donde lo guardes ahí lo podrás ejecutar, abrámoslo con PeID.



Nos dice que está entre la versión 1.xx o la 2.xx. ¡Qué precisión!

Probemos un detector más actualizado, el RDG Packer Detector.



Parece ser la versión definitiva. Armadillo v2.52 y el compilador Visual C++ v 6.0. ¿Cuántos procesos tendrá? Comprobémoslo en PUPE.



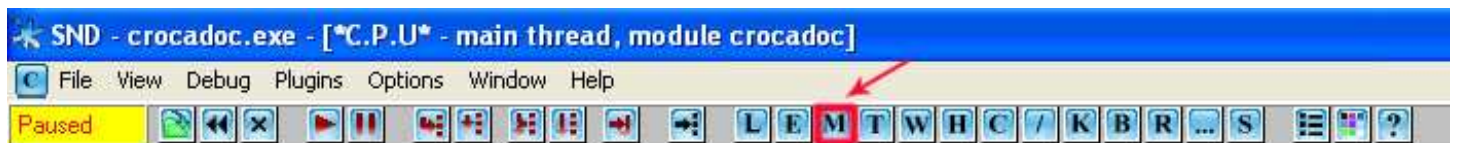
Hay un solo proceso. Esto se hace para descartar Copymen2.

BUSCANDO EL OEP

Lo cargamos en Olly y vemos el Entry Point del “Packer”.



Vayamos a “M”.



Coloquemos un Breakpoint con F2 en la sección CODE.



Ahora, presionemos F9 y para en el OEP (Original Entry Point).



Tomamos nota de los datos encontrados:

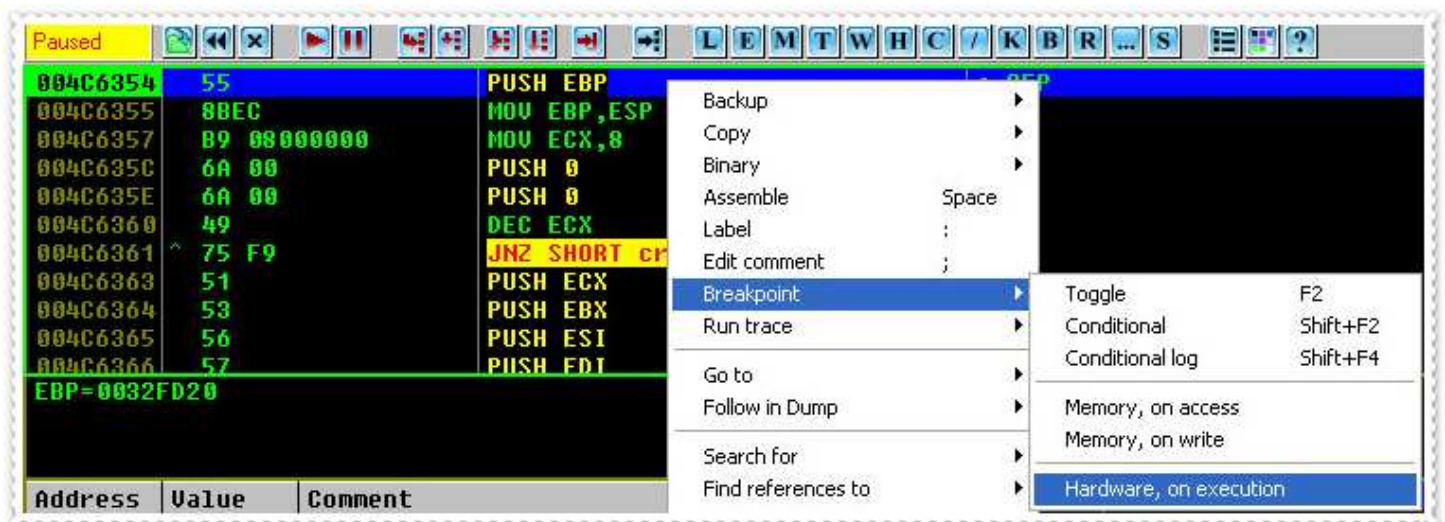
OEP = 004C6354

OEP en ImpRec = C6354 le restamos 400000 (Image Base) al OEP.

Entonces , tenemos el primer dato para ImpRec:

RVA= C6354

Aprovechemos de colocar un HE en el OEP.



BUSCANDO LA IAT

Le damos Enter a la primera CALL que vemos después del OEP.

SND - crocadoc.exe - [*C.P.U* - main thread, module crocadoc]

File View Debug Plugins Options Window Help

Paused

Address	Disassembly
004C6354	55 PUSH EBP ; OEP
004C6355	8BEC MOV EBP,ESP
004C6357	B9 08000000 MOV ECX,8
004C635C	6A 00 PUSH 0
004C635E	6A 00 PUSH 0
004C6360	49 DEC ECX
004C6361	75 F9 JNZ SHORT crocadoc.004C635C
004C6363	51 PUSH ECX
004C6364	53 PUSH EBX
004C6365	56 PUSH ESI
004C6366	57 PUSH EDI
004C6367	B8 4C604C00 MOV EAX,crocadoc.004C604C
004C636C	E8 1307F4FF CALL crocadoc.00406A84

Y después Enter de nuevo a la otra CALL. En comentarios dice que es un CALL hacia la API GetModuleHandleA.

SND - crocadoc.exe - [*C.P.U* - main thread, module crocadoc]

File View Debug Plugins Options Window Help

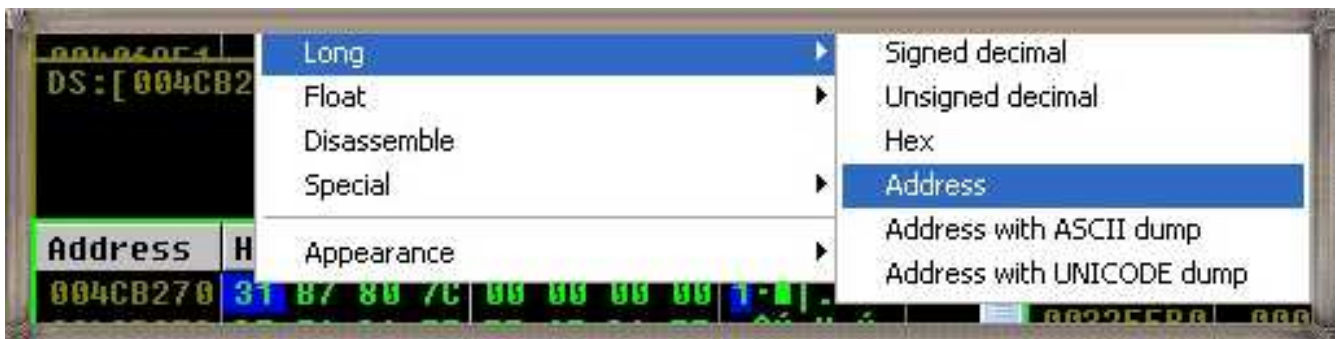
Paused

Address	Disassembly
00406A84	53 PUSH EBX
00406A85	8BD8 MOV EBX,EAX
00406A87	33C0 XOR EAX,EAX
00406A89	A3 0CA74C00 MOV DWORD PTR DS:[4CA70C],EAX
00406A8E	6A 00 PUSH 0
00406A90	E8 2BFFFFFF CALL crocadoc.004069C0
	JMP to kernel32.GetModuleHandleA

Vemos los siguientes JMP's a las API's en la IAT. Damos click derecho al primer salto y: Follow in Dump/Memory address.



Luego hacemos click derecho en el Dump para cambiar la vista a Long Address. Esta vista, Long Address, me gusta mucho; ya que se trabaja mejor por que se ven directamente los nombres de las API's.



Aquí les muestro una imagen.

Address	Value	Comment
004CB270	7C80B731	kernel32.GetModuleHandleA
004CB274	00000000	
004CB278	77DAEAD7	advapi32.RegSetValueExA
004CB27C	77DA6FEF	advapi32.RegQueryValueExW
004CB280	77DA7AAB	advapi32.RegQueryValueExA
004CB284	77DB4312	advapi32.RegQueryInfoKeyA

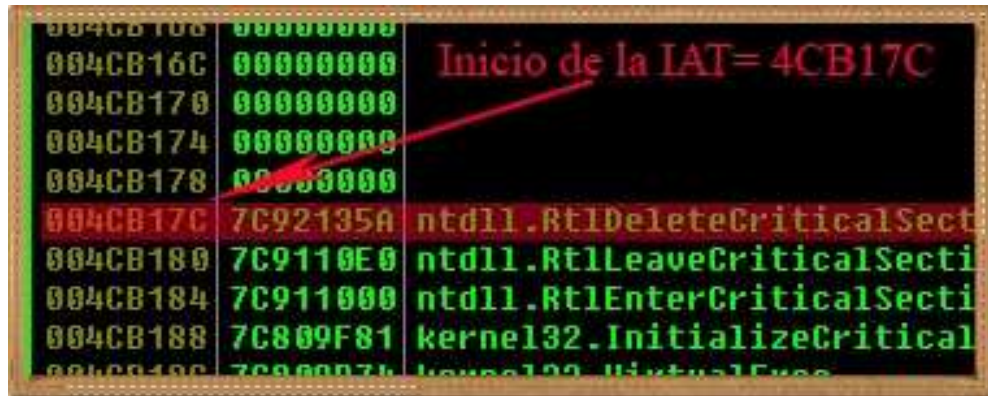
Si miramos la tabla completa, notamos que hay direcciones que no dicen nada a la derecha, los cuales son valores malos o API's encriptadas.

004CB214	7C812A99	kernel32.RaiseException
004CB218	7C812FC9	kernel32.GetStdHandle
004CB21C	00D865BF	
004CB220	00D866AE	¿Quién se robó mis API's?
004CB224	00D86059	
004CB228	00D86668	

Faltan unas cuantas API's. Ya solucionaremos esto.

REPARANDO LA IAT A MANO

Me cansé de buscar el salto mágico y no lo encontré. A lo mejor me faltan lentes o una brújula. Entonces lo hice a mano. Por una parte, es bueno porque prescindimos de lo automático y vemos como funcionan las cosas en vivo y directo. Subamos al inicio de la IAT.



Sabemos que es el inicio por no hay más nada arriba. Solamente ceros. Ya hemos encontrado dos datos para el ImpRec. Recuerden restarle 400000.

$$4CB17C - 400000 = CB17C.$$

OEP= C6354

RVA= CB17C

Bajemos para ver el final de la IAT.



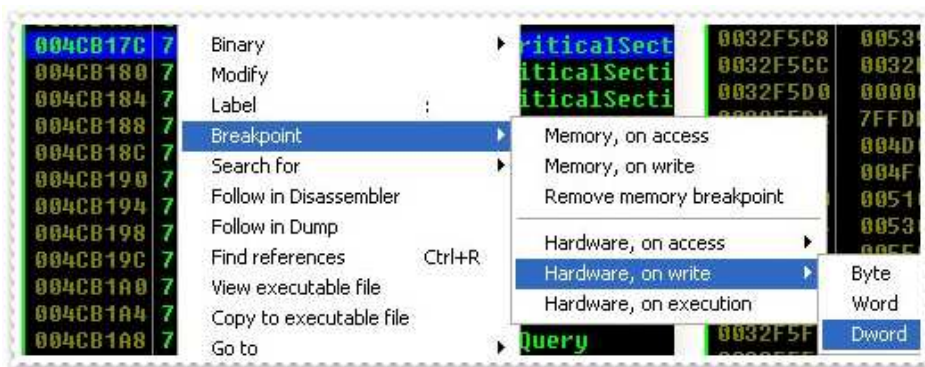
Nos falta el “Size” o largo de la IAT. La sacamos así:

Final - Inicio = Largo
4CB94C - 4CB17C = 7D0

Ya tenemos los 3 datos que nos faltaban para arreglar el dumpeado con ImpRec.

OEP = C6354
RVA = CB17C
SIZE = 7D0

La última API es MulDiv. ¿Cómo sé que lo demás es basura y no API's encriptadas? Ya lo veremos más adelante. Pongamos un Hardware Breakpoint on write Dword al inicio de la IAT en 4CB17C.



Reiniamos Olly con Ctrl+F2 y presionamos F9 (Run) y para en:

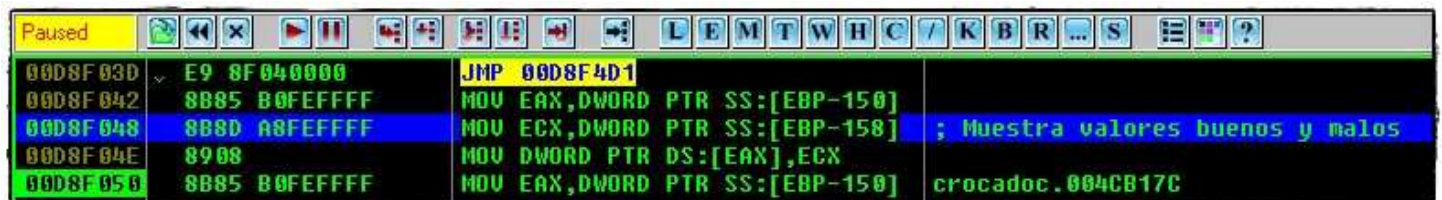


Esto no nos sirve.

Presionamos F9 de nuevo y para en:



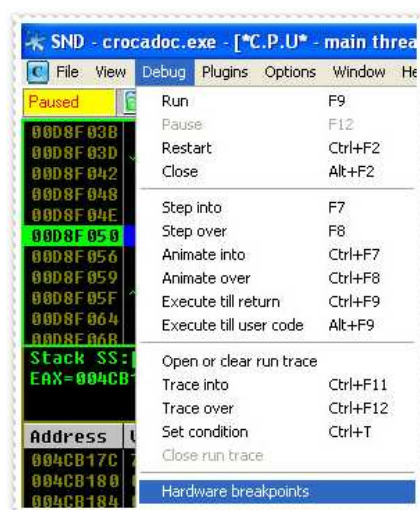
Si subimos un poco, veremos cuando vaya a meter un valor malo a la IAT.



Coloquemos un HE (Hardware Breakpoint on Execution) en la instrucción que ven arriba. En mi máquina es 00D8F048. En fin, es la instrucción:

MOV ECX,DWORD PTR SS:[EBP-158]

Quitemos el Hardware Breakpoint on write de 4CB17C.



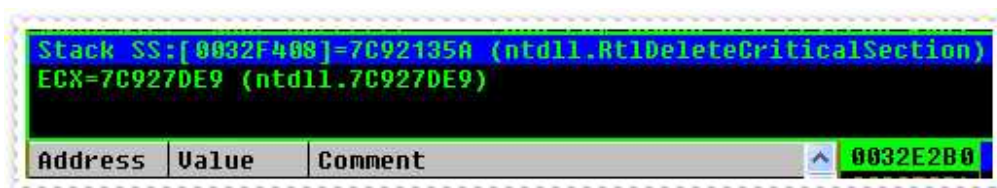
En la caja de Breakpoints, le damos Delete a 4CB17C:



Reiniciemos Olly (Ctrl+F2), damos F9 y para cuando va a escribir la primera API. (En este caso un valor bueno) porque vemos su nombre. **ntdll.RtlDeleteCriticalSection**.



Y en la aclaración de Olly también la vemos:



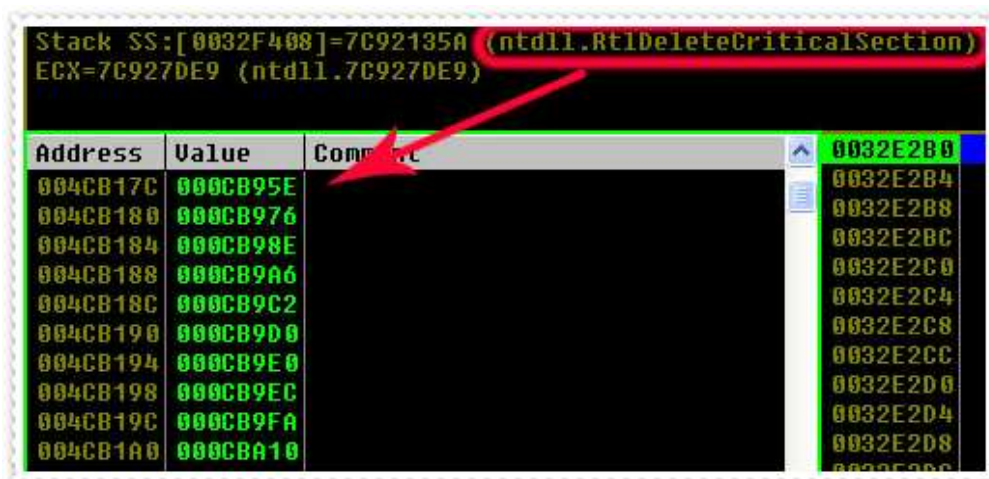
Vayamos al Dump al inicio de la IAT (4CB17C) para supervisar los valores que se van almacenando.



Y luego...



Y llegamos nuevamente a:



Desempacando Armadillo 2.52. Protección simple. 25/08/11 por Qvinson. Juto N° 3

En la imagen superior en rojo, vemos la primera API que se va a escribir en 4CB17C. Si damos F9, la escribe. Comprobémoslo.

Address	Value	Comment
004CB17C	7C92135A	ntdll.RtlDeleteCriticalSection
004CB180	000CB976	
004CB184	000CB98E	
004CB188	000CB9A6	

¿Vieron cómo apareció? Bueno, la reparación va a consistir en darle F9 poco a poco hasta que no veamos un nombre de API en:

Paused			
00D8F048	8B8D A8FEFFFF	MOV ECX,DWORD PTR SS:[EBP-158]	ntdll.RtlDeleteCriticalSection
00D8F04E	8908	MOV DWORD PTR DS:[EAX],ECX	

Por ejemplo, cuando le di 17 veces a F9 vi lo siguiente:

Paused			
00D8F048	8B8D A8FEFFFF	MOV ECX,DWORD PTR SS:[EBP-158]	
00D8F04E	8908	MOV DWORD PTR DS:[EAX],ECX	
00D8F050	8B85 B0FEFFFF	MOV EAX,DWORD PTR SS:[EBP-150]	
00D8F056	83C0 04	ADD EAX,4	
00D8F059	8985 B0FEFFFF	MOV DWORD PTR SS:[EBP-150],EAX	
00D8F05F	E9 9CFEFFFF	JMP 00D8EF0C	
00D8F064	0FB685 BCFEFFFF	MOVZX EAX,BYTE PTR SS:[EBP-144]	
00D8F06B	85C0	TEST EAX,EAX	
00D8F06D	74 76	JE SHORT 00D8F0E5	
00D8F06F	6A 00	PUSH 0	
00D8F071	8B85 C0FEFFFF	MOV EAX,DWORD PTR SS:[EBP-140]	
00D8F077	C1E0 02	SHL EAX,2	
00D8F07A	50	PUSH EAX	
00D8F07B	8B45 94	MOV EAX,DWORD PTR SS:[EBP-6C]	
00D8F07E	8B45 94	MOV EAX,DWORD PTR SS:[EBP-6C]	
Stack SS:[0032F408]=00D85DC5			
ECX=00D9A8A4, (ASCII "LoadLibraryExA")			

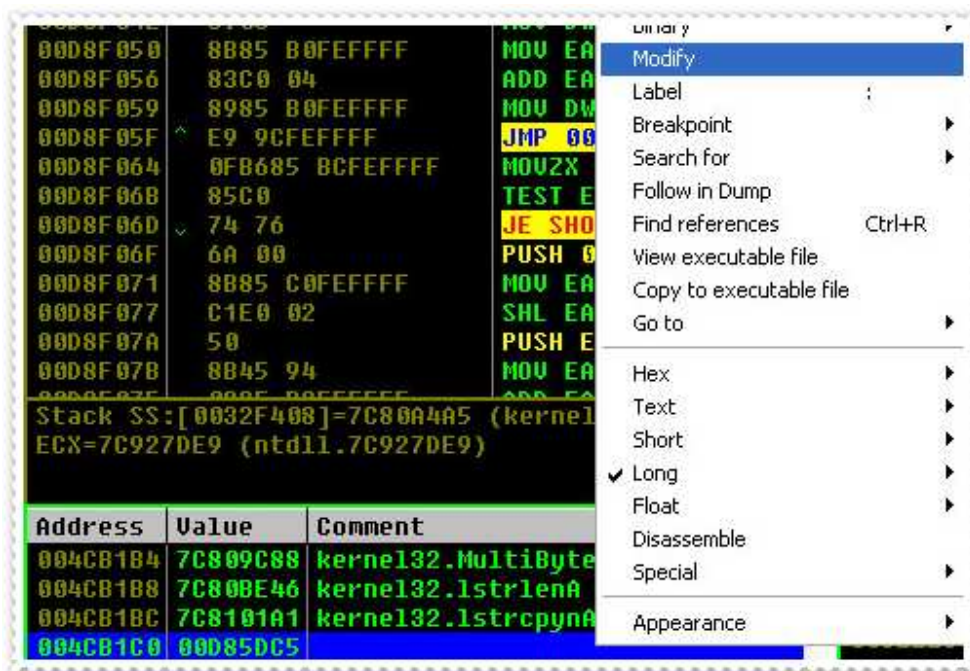
Iba guardando bien las API'S, pero de repente apareció en blanco como indica la flecha de arriba. Esto quiere decir que va a encriptar la API LoadLibraryExA. El plan es el siguiente.

Cada vez que aparezca en blanco, observamos la aclaración de OLLY y antes de darle F9, vamos a la CommandBar y tipeamos el nombre de la API colocando primero el signo “?”.

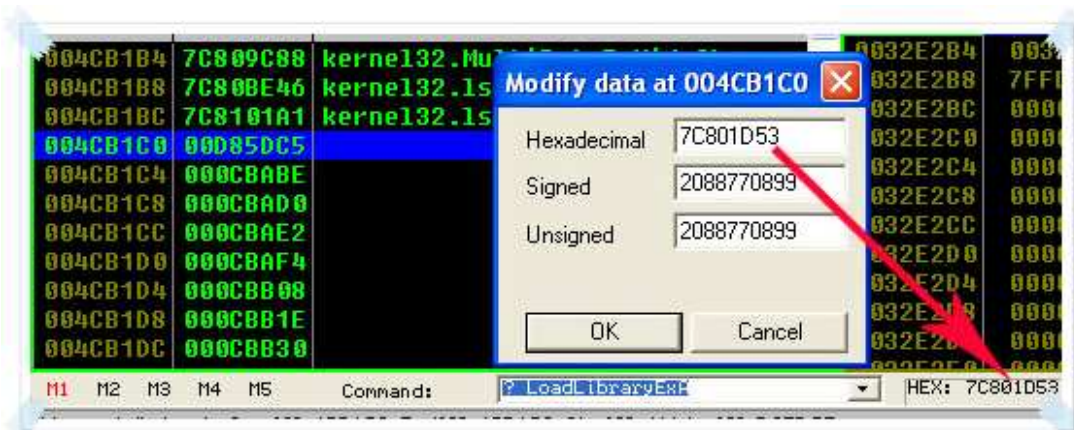
En este caso sería:



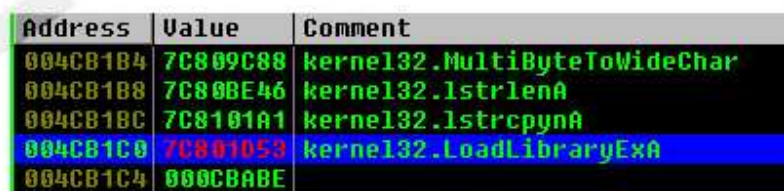
Con esto, ya tenemos el valor correcto de esa API. Así vamos a hacer con cada valor malo que vaya apareciendo. Entonces, doy F9 para que escriba el valor malo.



En 4CB1C0, escribió el valor malo 00D85DC5 (puede ser cualquiera). Le damos click derecho como en la imagen superior y a “Modify” para copiar el valor que nos dio la CommandBar.



Ésta quedaría así:



Address	Value	Comment
004CB1B4	7C809C88	kernel32.MultiByteToWideChar
004CB1B8	7C80BE46	kernel32.lstrlenA
004CB1BC	7C8101A1	kernel32.lstrcpynA
004CB1C0	7C801D53	kernel32.LoadLibraryExA
004CB1C4	000CBABE	

¿Interesante? Les digo que yo no soy el autor de esta técnica. La he visto en tutoriales que he leído. Solamente, me parece muy buena y ¿por qué no usarla? Siguiendo con F9 me encontré las siguientes API faltantes. No seguidas, claro. ¿Stolen API's? Suena chistoso.

LoadLibraryExA-----7C801D53
GetProcAddress-----7C80AE30
FreeLibrary-----7C80AC6E
ExitProcess-----7C81CAFA
CreateThread-----7C8106C7
WriteFile-----7C810E17
SetFilePointer-----7C810C1E
ReadFile-----7C801812
GetFileSize-----7C810B07
GetFileType-----7C810EE1
CreateFileA-----7C801A28
CloseHandle-----7C809BD7
MessageBoxA-----7E3D07EA
WriteFile-----7C810E17-----De nuevo
SetFilePointer-----7C810C1E-----De nuevo

ReadFile-----7C801812-----De nuevo

Desempacando Armadillo 2.52. Protección simple. 25/08/11 por Qvinson. Juto Nº 3

LoadLibraryA-----7C801D7B
GetProcAddress-----7C80AE30----De nuevo
GetEnvironmentVariableA-----7C814B82
FreeLibrary-----7C80AC6E----De nuevo
CreateThread-----7C8106C7----De nuevo
CreateFileA-----7C801A28
CloseHandle-----7C809BD7----De nuevo
MessageBoxA-----7E3D07EA

Después que escribe “MulDiv”, el programa se detiene en el OEP (ya que habíamos puesto un HE al inicio del tutorial). Es aquí donde ustedes se dan cuenta que lo que quedaba después de “MulDiv” era basura.



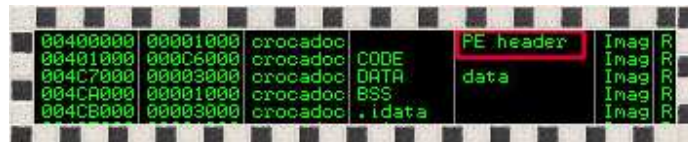
Ya con esto tenemos una IAT impecable.

REPARANDO EL “HEADER” (CABECERA)

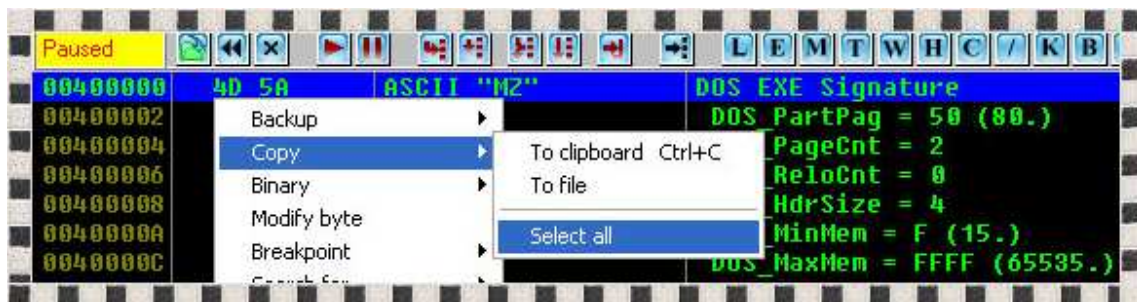
Si vamos a “M” (Memory), vemos que no está el header.



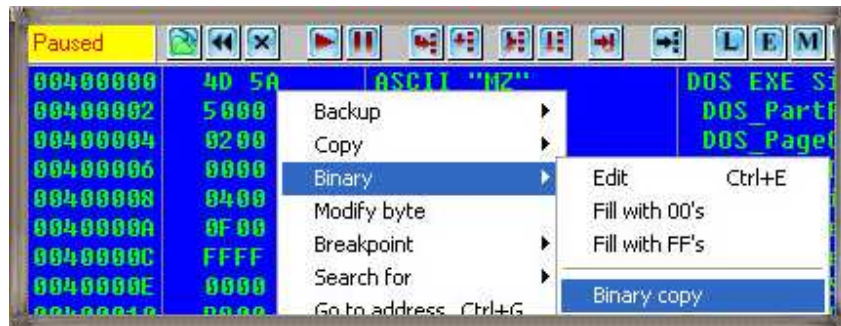
Abramos otro Olly y carguemos el programa. (El primer Olly lo dejamos quieto en el OEP). Vamos igualmente a “M” como hicimos con el primer Olly y vemos:



Ahí está nuestro header. Démosle doble click. En la ventana que nos sale hacemos lo siguiente:

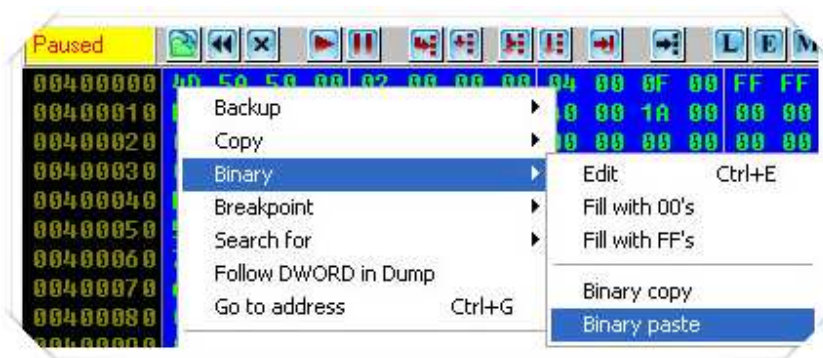


Después de haber seleccionado todo, click derecho nuevamente:



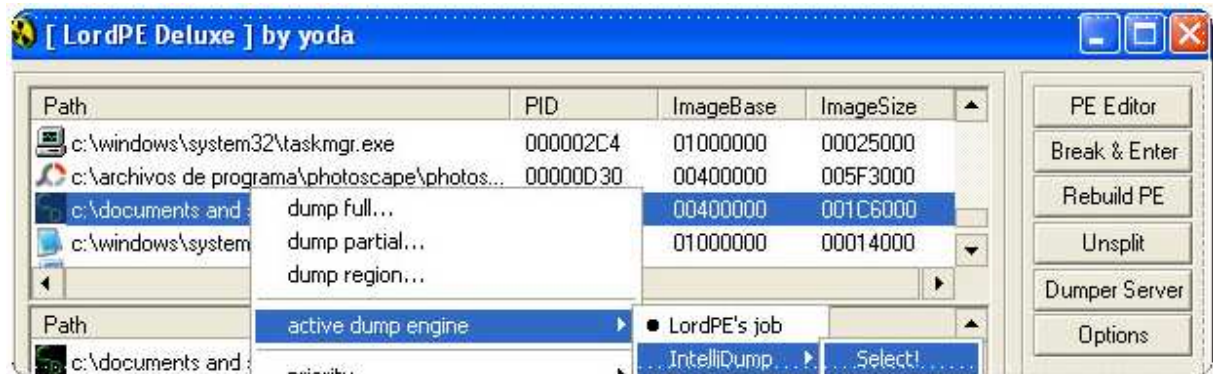
Vamos al primer Olly, en “M”, doble click en donde se supone va el header (400000).

Seleccionamos todo y pegamos así:



DUMPEANDO

Tenemos el primer Olly parado en el OEP y la IAT está reparada. Abramos LordPE, seleccionemos el proceso crocadoc.exe que es nuestro ejecutable y démosle a IntelliDump.



Luego, damos click derecho de nuevo y seleccionamos Dump full y guardamos.

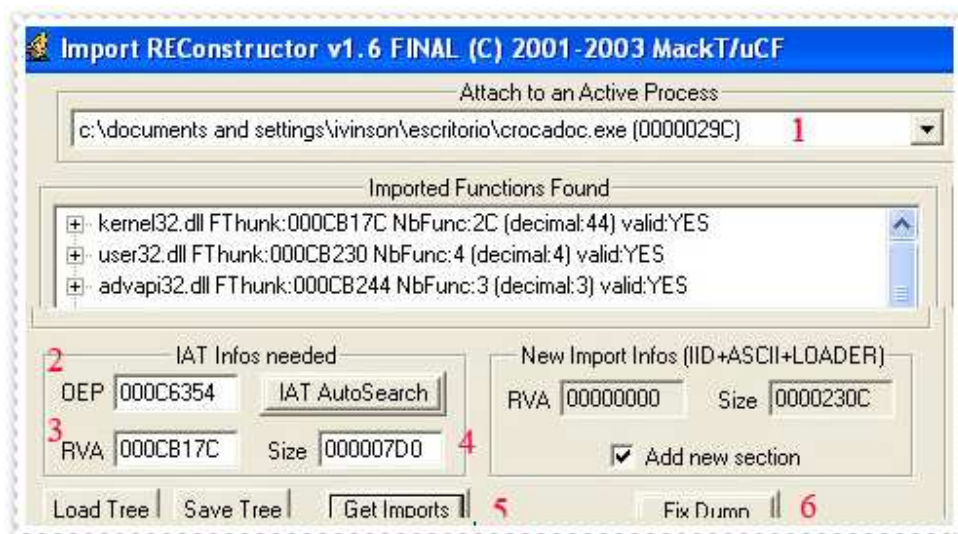


Esperamos que salga la siguiente ventana y listo.



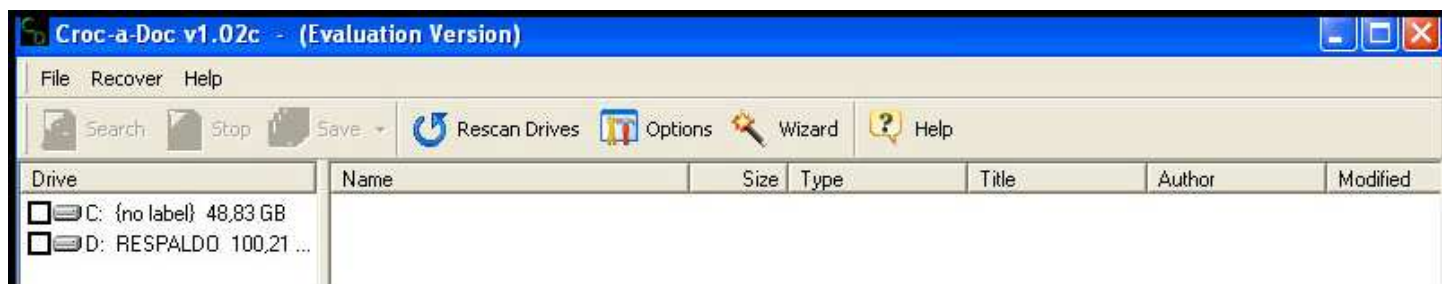
REPARANDO EL DUMPEADO

Ya guardamos el Dumpeado en el escritorio con el nombre Dumped.exe. Abramos ImpRec y sigamos los pasos abajo descritos:



Desempacando Armadillo 2.52. Protección simple. 25/08/11 por Qvinson. Juto Nº 3

Al darle Fix Dump (6), seleccionamos el Dumped.exe del paso anterior que lo habíamos guardado en el escritorio, lo ejecutamos y vuela como bruja.



ANALIZANDO LA FORMA DE REGISTRO

Si vamos a Help/About Croc-a-doc.../Register, vemos la ventana:

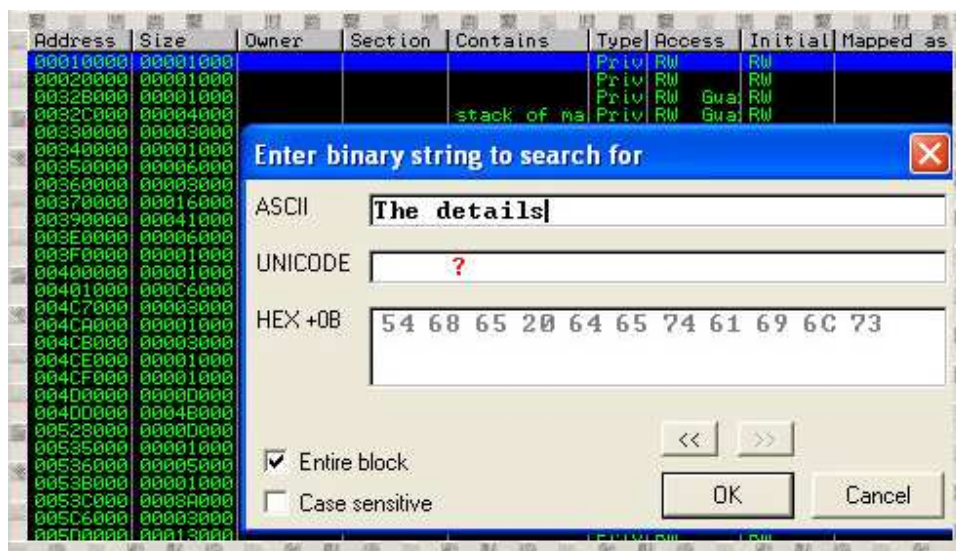


Nos dice que si nos registramos, tendremos doble beneficio. Primero, podremos guardar los documentos recuperados, y segundo, podremos optar por actualizaciones. Aceptan tarjeta de crédito, efectivo y orden de compra. Como yo estoy desempleado no voy a comprarlo y si estuviera trabajando tampoco lo haría. Cracker que se respeta no compra programas.

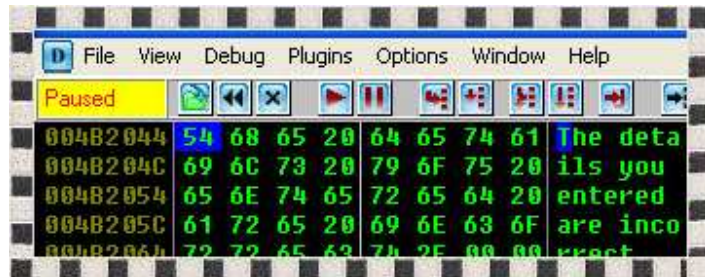
Coloquemos el nombre: Ivinson y serial: 123456789.



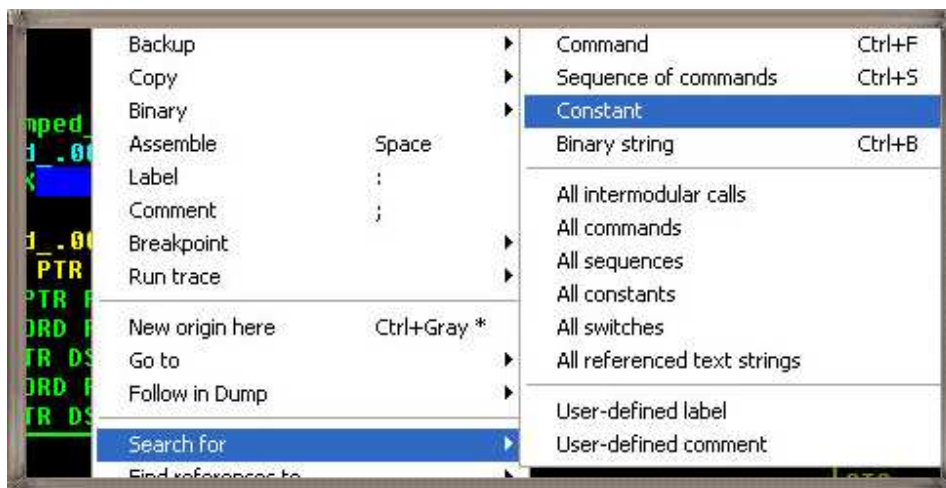
Ahí está el “chico malo” que dice “The details you entered are incorrect”. Bueno, ahora carguémoslo en Olly y a gozar. Demos F9 y miremos las text strings y nada. Vayamos a “M”, seleccionamos la primera línea y presionemos Ctrl+B. Nos sale la ventana de búsqueda y ponemos:



En mi máquina cayó en 4B2044.



Para buscar esa dirección en el Desensamblado, lo haremos de la siguiente manera:



Al dar click derecho en el desensamblado, buscaremos la constante 4B2044.



Y magia... caemos donde están todas las comprobaciones y tipos de licencia. Por ejemplo, licencia por 7 días o permanente. Las cuales podemos ver si subimos. Pueden darle al Scroll con el puntero y así suben más cómodamente porque RegPg y AvPg son muy rápidas.



Pero déjenme decirles que la comprobación más importante no está ahí, y fue la que me hizo perder mucho tiempo. Es irónico porque la primera vez no tardé mucho, pero luego olvidé como la había encontrado. Por ahora, concentrémonos en donde estamos.

Zona 1. Donde llegamos al buscar la constante 4B2044.

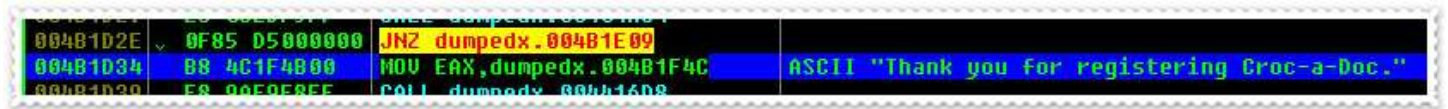
004B1EF2	B8 44204B00	MOV EAX,dumpedx.004B2044	ASCII "The details you entered are incorrect."
----------	-------------	--------------------------	--

Subamos y veamos las demás zonas importantes:

Zona 2. Salto incondicional que evita cualquier tipo de licencia.

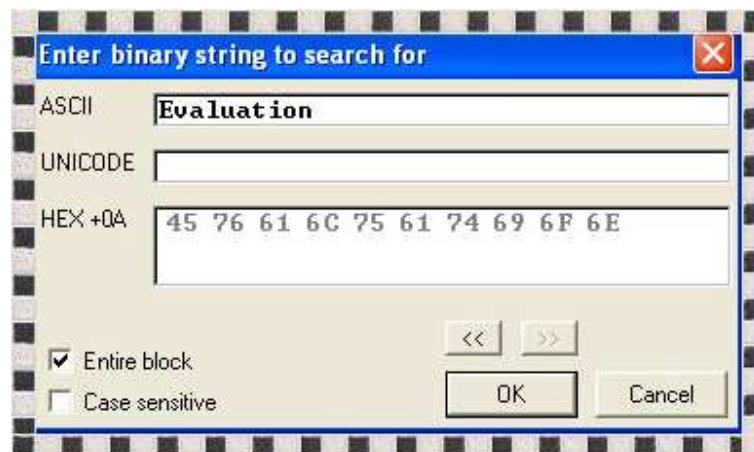
004B1CF6	0F 84 F6010000	JE dumpedx.004B1EF2	
004B1CFC	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]	
004B1CFF	BA 301F4B00	MOV EDX,dumpedx.004B1F30	ASCII "TYPE"
004B1D04	A1 4C8D4C00	MOV EAX,DWORD PTR DS:[4C8D4C]	

Zona 3. Chico bueno y un salto condicional.



PARCHANDO

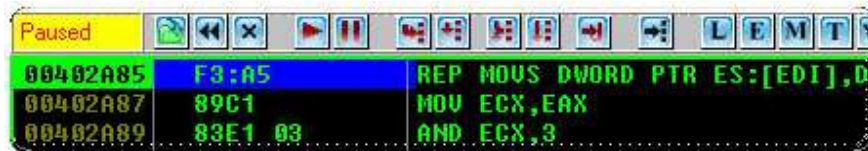
Lo primero que hice fue nopear la zona 3 para llegar directamente a Chico bueno, pero me seguía saliendo el cartel de la zona 1. Entonces decidí reemplazar la zona 1 por un JMP hacia la zona 3. Obligándolo de todas formas a que me registrara. Esta idea funcionó. Después de guardar todos los cambios, ejecutaba el programa, introducía mi nombre y cualquier clave y aparecía como registrado, pero al reiniciar, el programa seguía diciendo “Evaluation Version”. Lo primero que pensé fue en CRC32, pero el programa ya estaba desempacado y funcionando. Lo cargué en Olly nuevamente y comencé a buscar el zona 4. La cuarta maravilla. Usando la lógica, el ejecutable tiene que hacer una comparación al iniciar para chequear si estoy registrado o no; por lo que estando en Olly, fui a “M”, seleccioné la primera línea y presioné Ctrl+B para buscar el texto “Evaluation”.



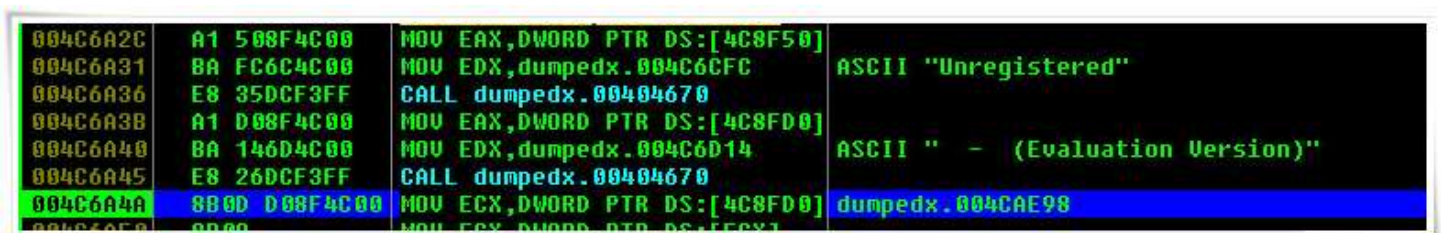
Al abrirse la ventana con el resultado, seleccionamos la palabra “Evaluation” y ponemos un Breakpoint como en la imagen.



Al darle F9, para en:



Empecemos a trazar con F8. Al darle 17 veces, caí en una zona muy buena.



De ahí es donde coge el texto “Evaluation Version”. Subamos más y veremos lo siguiente:

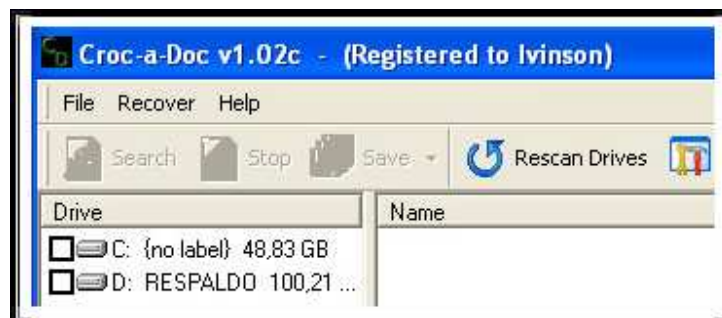


La bendita zona 4. Que en realidad es la zona 1; ya que al NOPear no hace falta hacer lo que hicimos con las zonas anteriores. Ésta es la única comprobación real.

Al NOPearlo, (Pulsamos la barra de espacio y escribimos NOP) quedaría así:



Y al guardar los cambios. Click derecho/Copy to executable /selection/click derecho/ save file, cada vez que lo ejecutemos, dirá “Registered to ‘usuario’ ”; ya que toma nuestro nombre de usuario de Windows.



Meta alcanzada.

CONCLUSIÓN

Después de haber usado casi 70 imágenes espero haberme explicado bien. Y que los que esten comenzando, como yo, vean lo interesante que puede ser el cracking no solo para tener un programa full. Lo que se busca es superar barreras, incrementar el conocimiento y ser mejor cracker cada día. Por muy pequeño que sea el reto, el conocimiento siempre va a ser mayor.

En cuanto al algoritmo del serial, invito a cualquiera de los listeros que quieran escribir un tutorial explicando como crea la clave y si lo prefieren háganle un “keygen”.

DATO CURIOSO

17 veces (F9) para llegar al valor malo en la IAT. 17 veces (F9) para llegar a la zona de “Evaluation Version”. Zona 4.

AGRADECIMIENTOS

A Dios principalmente, a mis abuelos, padres, a toda la familia CLS, y a todos los que han escrito tutos de Armadillos y otros, sin ellos este tuto no hubiera nacido.

FRASE

“Todo el tiempo que pasas quejándote de que no puedes lograr algo, inviértelo en buscar la solución” Ivinson.

Cualquier comentario hacerlo a: ipadilla63@gmail.com

Bonus: Mi dibujo más reciente. Hasta el 4to tutorial. Dios mediante.

