



➤ Programa	Alcohol 120 %
➤ Versión	1.9.6
➤ Herramientas	OllyDbg y DUP2
➤ Compilador	
➤ Objetivos	Crear un loader
➤ Cracker	CrAmEr ☺

El programa a tratar es muy básico en cuanto a packers y protecciones, para este caso no nos orientaremos a desempaquetar y luego modificar el código para dejar limpio de restricciones.

OBJETIVO

- Crear un loader que permita correr el programa en versión full☺.

Procedimiento

Bueno empezamos con lo de costumbre y eso es cargarlo al olly y correr el ejecutable, si corre sin ningún problema entonces podemos trabajar, caso contrario debemos buscar la manera de solucionar ese problema, ya sea con plugins o con otros Ollys preparados.

En nuestro caso el OllyDbg que uso es el “**odbg110 Diablo**”, pero funciona cualquiera.

A la carga...., lo cargamos y vemos esto.

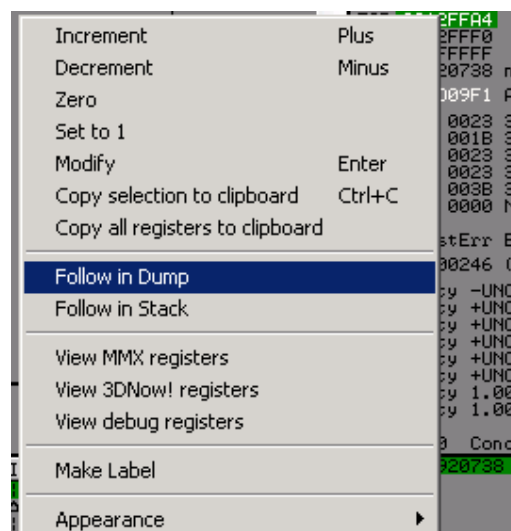
Address	Hex dump	Disassembly	Comment
009009F0	60	PUSHAD	
009009F1	BE 00708500	MOV ESI,00857000	
009009F6	8DBE 00A0BAFF	LEA EDI,DWORD PTR DS:[ESI+FFBAA000]	
009009FC	57	PUSH EDI	
009009FD	83CD FF	OR EBP,FFFFFFFF	
00900A00	EB 10	JMP SHORT 00900A12	
00900A02	90	NOP	
00900A03	90	NOP	
00900A04	90	NOP	
00900A05	90	NOP	
00900A06	90	NOP	
00900A07	90	NOP	
00900A08	8A06	MOV AL,BYTE PTR DS:[ESI]	
00900A0A	46	INC ESI	
00900A0B	8B07	MOV BYTE PTR DS:[EDI],AL	
00900A0D	47	INC EDI	
00900A0E	01DB	ADD EBX,EBX	
00900A10	75 07	JNZ SHORT 00900A19	
00900A12	8B1E	MOV EBX,DWORD PTR DS:[ESI]	
00900A14	83EE FC	SUB ESI,-4	
00900A17	11DB	ADC EBX,EBX	
00900A19	72 ED	JB SHORT 00900A08	
00900A1B	B8 01000000	MOV EAX,1	
00900A20	01DB	ADD EBX,EBX	
00900A22	75 07	JNZ SHORT 00900A2B	
00900A24	8B1E	MOV EBX,DWORD PTR DS:[ESI]	

La mayoría de los packers tienen trucos que ayudan a descomprimir mas rápido, en nuestro caso se puede utilizar el método del pushad, puesto que supondremos que no sabemos que packer es, pero si quieren saber con que nos enfrentamos, pues coloquen el peid u otro y les dirá que es un **UPX**.

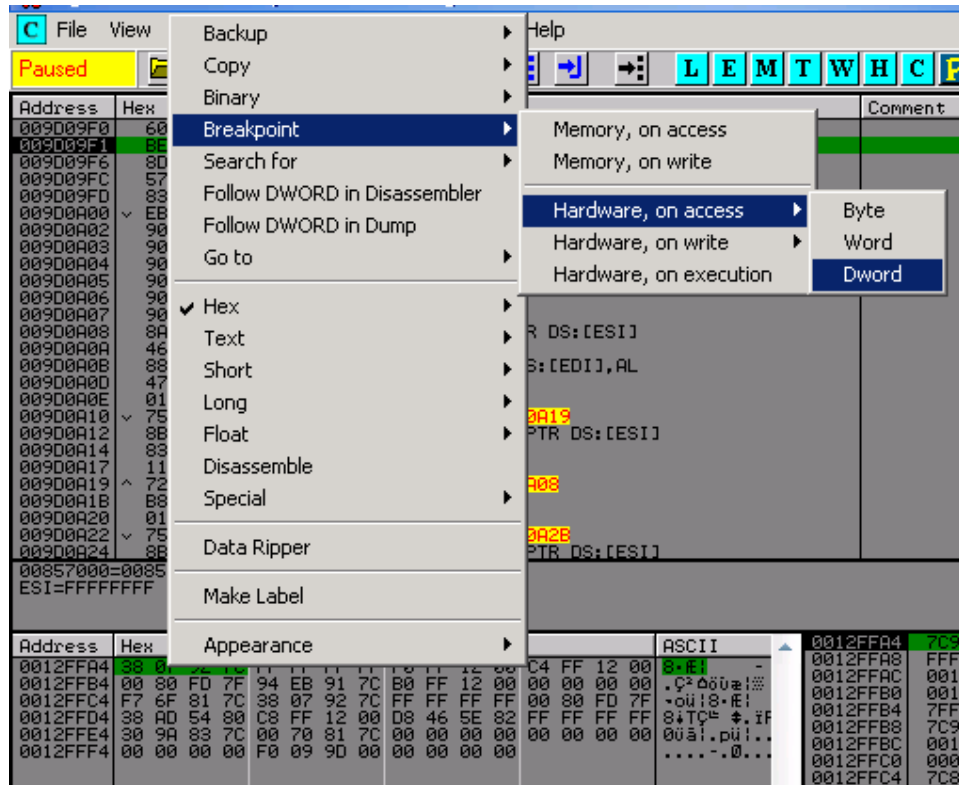
DESCOMPRESION MANUAL

METODO DEL PUSHAD

- Presionamos F8
- Vamos a los registros, clic derecho sobre el ESP y Follow in Dump.



- En memoria seleccionamos unos bytes y colocamos un HE DWORD



- Presionamos F9 y estamos sobre la rutina final de descompresión, a un paso del salto al OEP ☺

009D0B97	8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]	
009D0B98	6A 00	PUSH 0	
009D0B9D	39C4	CMP ESP,EAX	
009D0B9F	75 FA	JNZ SHORT 009D0B9E	
009D0BA1	83EC 80	SUB ESP,-80	
009D0BA4	E9 B30A3FF	JMP 0040165C	Salto al OEP
009D0BA9	0000	ADD BYTE PTR DS:[EAX],AL	
009D0BAB	00C4	ADD AH,AL	
009D0BAD	0B9D 00800C9D	OR EBX,DWORD PTR SS:[EBP+9D0C8000]	
009D0BB3	00B470 7E000000	ADD BYTE PTR DS:[EAX+ESI*2+7E1],DH	

Ahora colocamos un BP en el salto y F8 y ya estamos sobre el OEP.

0040165C	EB 10	JMP SHORT 0040166E	<<= = DEP :-)
0040165E	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401661	43	INC EBX	
00401662	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401664	48	DEC EAX	
00401665	4F	DEC EDI	
00401666	4F	DEC EDI	
00401667	4B	DEC EBX	
00401668	90	NOP	
00401669	E9 9806400	JMP 00A41706	
0040166E	A1 8B06400	MOV EAX,DWORD PTR DS:[64008B]	
00401673	C1E0 02	SHL EAX,2	
00401676	A3 8F06400	MOV DWORD PTR DS:[64008F],EAX	
0040167B	52	PUSH EDX	
0040167C	6A 00	PUSH 0	
0040167E	E8 F3DB2300	CALL 0063F276	JMP to kernel32.GetM
00401683	8BD0	MOV EDX,EAX	
00401685	FB 00F2000	CALL 00600594	

Esto indica que esta descomprimido en memoria, ahora analizamos el código y listo para trabajar.

BUSQUEDA DEL BYTE REGISTRADOR:-P

Verán en cada programa sea cual sea, existe una manera de registrarlo que suele ser bastante fácil y otra que es todo lo contrario pero es mas elegante.

La fácil es parchar para que quede registrado, pero para eso debemos hacerlo bien sino mejor no lo hacemos☺, y esto es así porque debemos de ir mejorando constantemente y dejar de ser mediocres en la vida...

La difícil es mas satisfactoria ya que se debe encontrar un serial valido o una llave que permita el registro permanente del programa sin modificarle ningún solo byte, es decir conservar su integridad, a eso es lo que debemos aspirar jeje, pero es difícil en algunos casos y es mejor y poco a poco y no querer volar antes de tiempo.

Para nuestro caso la mejor manera y la más elegante es parchar, la razón es que no hay un cuadro de registro que nos permita buscar un serial para registrar.

Lo que haremos será presionar F9 y dejar que aparezca la nag, luego pausamos, y vamos al **Stack**, buscaremos algo sospechoso, pero como sabremos que es.

Simple, en el stack están los retornos a cada sección y a cada call, así que empezamos desde donde esta parado el Olly y descendemos buscando en cada retorno.

Para mi este es interesante:

0012F404	7E399408	RETURN to USER32.7E399408
0012F408	00588C7B	RETURN to Alcohol.00588C7B from Alcohol.0063FD20
0012F40C	0012F460	Pointer to next SEH record
0012F410	00588C96	SE handler
0012F414	0012F434	
0012F418	00000001	
0012F41C	000007DA	
0012F420	011B1DC8	
0012F424	00000000	
0012F428	00000000	
0012F42C	01000001	
0012F430	011B1DC8	
0012F434	0012F4AC	
0012F438	005882D0	RETURN to Alcohol.005882D0 from Alcohol.00588B88
0012F43C	001E04DC	
0012F440	0000000F	
0012F444	00000000	
0012F448	00000000	
0012F44C	01613287	
0012F450	000000AD	
0012F454	00000034	
0012F458	007E6F58	Alcohol.007E6F58
0012F45C	005850B0	RETURN to Alcohol.005850B0 from Alcohol.005882B4
0012F460	0012F46C	Pointer to next SEH record
0012F464	00585133	SE handler
0012F468	0012F4AC	
0012F46C	0012F478	Pointer to next SEH record
0012F470	005851D4	SE handler
0012F474	0012F4AC	
0012F478	0012F4E4	Pointer to next SEH record
0012F47C	005851F4	SE handler
0012F480	0012F4AC	
0012F484	000007DA	
0012F488	011B60C8	
0012F48C	00000000	
0012F490	00000000	
0012F494	00000001	
0012F498	000002AE	
0012F49C	00000000	
0012F4A0	00000000	
0012F4A4	00000000	
0012F4A8	011B60C8	
0012F4AC	0012F578	
0012F4B0	00421710	RETURN to Alcohol.00421710

Buscamos
Algo
Interesante
;-P



0012F51C	00000000	
0012F520	00000000	
0012F524	005F2103	RETURN to Alcohol.005F2103 from Alcohol.005F1CA0
0012F528	011B9A0C	
0012F52C	011B9A10	
0012F530	011B99EC	
0012F534	00000024	
0012F538	005F1E09	RETURN to Alcohol.005F1E09 from Alcohol.005F1830
0012F53C	005F1E31	RETURN to Alcohol.005F1E31 from Alcohol.0063F378
0012F540	011B99F0	
0012F544	005DA940	Alcohol.005DA940
0012F548	00000000	
0012F54C	00000000	
0012F550	00000000	
0012F554	00000000	
0012F558	00000000	
0012F55C	00000000	
0012F560	00000000	
0012F564	011B7174	ASCII "Info"
0012F568	011B60C0	
0012F56C	011B60B0	
0012F570	011B60A0	
0012F574	011D3034	ASCII "TypedURLs"
0012F578	0012F9A8	
0012F57C	004032BD	RETURN to Alcohol.004032BD from Alcohol.00420F5C
0012F580	0064A258	Alcohol.0064A258
0012F584	011B1DC8	
0012F588	011B5744	
0012F58C	00000000	
0012F590	0000ECE3	
0012F594	0012E864	
0012F598	0012F834	
0012F59C	001846C0	
0012F5A0	0012F830	
0012F5A4	00000000	
0012F5A8	00000000	
0012F5AC	04FC00BA	
0012F5B0	02080000	



Este call es importante porque fui a ver que tenia presionando ENTER sobre el return, y vi muchas cosa que podrían ayudar a encontrar dicho byte jeje ☺

Address	Hex dump	Disassembly	Comment
0040326F	. 894D 8C	MOV DWORD PTR SS:[EBP-74],ECX	
00403272	. 66:C785 4CFDFFFF	MOV WORD PTR SS:[EBP-2B4],110	
00403276	. BA 03000000	MOV EDI,3	
00403280	. 8B45 88	MOV EAX,DWORD PTR SS:[EBP-78]	
00403283	. 8B88	MOV ECX,DWORD PTR DS:[EAX]	
00403285	. FF51 FC	CALL DWORD PTR DS:[ECX-4]	
00403288	. 66:C785 4CFDFFFF	MOV WORD PTR SS:[EBP-2B4],104	
00403291	> A1 7C9C6900	MOV EAX,DWORD PTR DS:[699C7C]	
00403296	. 50	PUSH EAX	Arg2 => 4B61FED3
00403297	. 8B15 18386500	MOV EDI,DWORD PTR DS:[653818]	Arg1 => 4B625804
0040329D	. 52	PUSH EDI	Alcohol.006147E0
0040329E	. E8 3D152100	CALL 006147E0	
004032A3	. 83C4 08	ADD ESP,8	
004032A6	. DB2D DC5B4000	FLD TBYTE PTR DS:[405BDC]	
004032AC	. DEC9	FMLP ST(1),ST(0)	
004032AE	. E8 CDD72000	CALL 00610A80	
004032B3	. A3 F0446500	MOV DWORD PTR DS:[6544F0],EAX	
004032B8	. E8 7DC01000	CALL 00420F5C	<<== genera el nag
004032BD	. 8B55 FC	MOV EDI,DWORD PTR SS:[EBP-4]	<<== Retorna
004032C0	. 8332 D3050000	MOV BYTE PTR DS:[EDX+5D3],AL	
004032C6	. E8 91E7FFFF	CALL 00401A5C	
004032C8	. 8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
004032CE	. 8A81 D3050000	MOV AL,BYTE PTR DS:[ECX+5D3]	
004032D4	. 84C0	TEST AL,AL	
004032D6	~ 74 5D	JE SHORT 00403335	
004032D8	. 6A 06	PUSH 6	
004032DA	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004032DD	. E8 02A21900	CALL 0059D4E4	
004032E2	. 50	PUSH EAX	hWnd
004032E3	. E8 F6C92300	CALL 0063FCDE	ShowWindow
004032E8	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004032EB	. E8 1C010000	CALL 00594000	

Entonces al presionar enter salio en 4032BD, lo que indica que el call de arriba es el que saca el nag y por ende cerca el byte registrador:-P

Colocamos un HE en 4032B8 y un HE en el OEP y reiniciamos.

Ahora F9 dos veces y estamos sobre el BP del call generador del nag, con F7 para entrar al call y vemos una peque comprobación que ayuda a determinar si estamos registrados o no, así que entramos en el call antes de llegar al salto condicional y con F7 llegamos a dos instrucciones muy interesantes.

00420F5C	55	PUSH EBP	
00420F5D	8BEC	MOV EBP,ESP	
00420F5F	81C4 48FFFFFF	ADD ESP,-0B8	
00420F65	B8 27BC6000	MOV EAX,0060BC27	
00420F6A	33D2	XOR EDX,EDX	
00420F6C	53	PUSH EBX	
00420F6D	56	PUSH ESI	
00420F6E	57	PUSH EDI	
00420F6F	C785 74FFFFFF 40C	MOV DWORD PTR SS:[EBP-8C],0064CA40	
00420F79	8985 78FFFFFF	MOV DWORD PTR SS:[EBP-88],ESP	
00420F7F	8985 70FFFFFF	MOV DWORD PTR SS:[EBP-90],EAX	
00420F85	66:C785 7CFFFFFF	MOV WORD PTR SS:[EBP-84],0	
00420F8E	8955 88	MOV DWORD PTR SS:[EBP-78],EDX	
00420F91	64:8B00 00000000	MOV ECX,DWORD PTR FS:[0]	
00420F98	898D 6CFFFFFF	MOV DWORD PTR SS:[EBP-94],ECX	
00420F9E	8D85 6CFFFFFF	LEA EAX,DWORD PTR SS:[EBP-94]	
00420FA4	64:A3 00000000	MOV DWORD PTR FS:[0],EAX	
00420FAA	C685 6BFFFFFF 00	MOV BYTE PTR SS:[EBP-95],0	
00420FB1	E8 A60AFFFF	CALL 00401A5C	<<== determina el reg
00420FB6	84C0	TEST AL,AL	
00420FB8	0F85 D8070000	JNZ 00421796	<<== salta si estamos registrados
00420FBE	C685 6AFFFFFF 00	MOV BYTE PTR SS:[EBP-96],0	
00420FC5	E8 F2171C00	CALL 005E27BC	
00420FCA	DD08	FSTP ST(0)	
00420FCC	B8 84246200	MOV EBX,006224A4	
00420FD1	FFD3	CALL EBX	
00420FD3	DD9D 58FFFFFF	FSTP QWORD PTR SS:[EBP-A8]	
00420FD9	8D85 64FFFFFF	LEA EAX,DWORD PTR SS:[EBP-9C]	
00420FDF	8D8D 66FFFFFF	LEA ECX,DWORD PTR SS:[EBP-9A]	
00420FE5	50	PUSH EAX	Arg1
00420FE6	8D85 58FFFFFF	LEA EAX,DWORD PTR SS:[EBP-A8]	
00420FE8	8D85 60FFFFFF	LEA ECX,DWORD PTR SS:[EBP-98]	

00401A5C	55	MOV BYTE PTR DS:[6544EC],0
00401A63	33C0	XOR EAX,EAX
00401A65	C3	RET

Entonces la situación es esta, en el primer dibujo se ve que el salto se ejecuta si **al = 1**

Lo que indicaría que estamos registrados y si **al = 0** estamos en prueba, así que la solución es obvia debemos modificar el **XOR EAX,EAX** por **MOV AL,1** y asunto solucionado jeje.

Pero como no podemos guardar esto por estar solo en memoria, lo que debemos hacer es copiar los bytes viejos y los nuevos, puesto que el objetivo es hacer un loader:-O

Bytes	nuevos	viejos
00401A63	B0	33
00401A64	01	C0

Ahora F9 y veremos en la parte de ayuda el acerca de:

REGISTRADO



SIN REGISTRAR

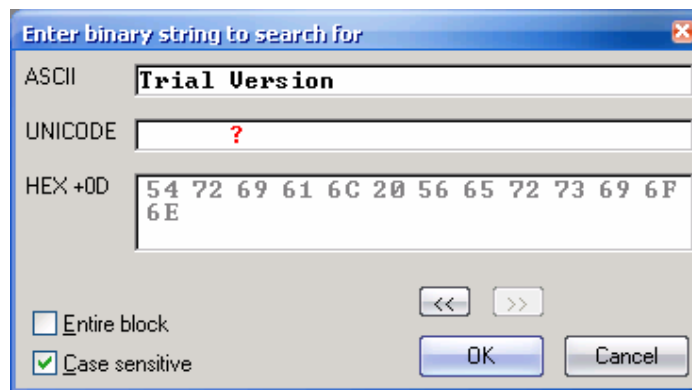


Como ven la diferencia es notoria y ya solo queda modificar el trial versión que esta en la ventana principal, pero el porque de ese mensaje, seguramente hay otra comprobación que verifica el registro, pero no nos interesa ya que el programa esta full y sin ningún problema jeje.

Ahora debemos modificar eso y listo:-P

DANDOLE LOS ULTIMOS TOQUES

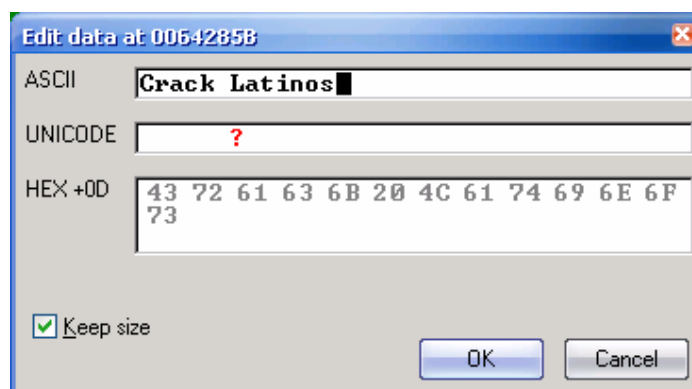
Para ya terminar esto, reiniciemos el programa y lleguemos hasta el OEP, ahora vamos a memory y busquemos la cadena fea☺.



Y para aquí jeje:

0064285B	54	72	69	61	6C	20	56	65	72	73	69	6F	6E	00	2E	2E	Trial Version...
0064286B	2E	00	26	00	00	26	00	00	26	00	00	26	00	00	26	00	...&...&...&...
0064287B	00	26	00	00	26	00	00	26	00	00	26	00	00	26	00	00	...&...&...&...
0064288B	26	00	00	26	00	00	26	00	00	26	00	00	26	00	00	26	...&...&...&...
0064289B	00	00	26	00	00	26	00	00	26	00	00	26	00	00	26	00	...&...&...&...
006428AB	00	26	00	00	26	00	00	26	00	00	26	00	00	26	00	00	...&...&...&...
006428BB	26	00	00	26	00	00	26	00	00	26	00	00	26	00	00	2E	...&...&...&...

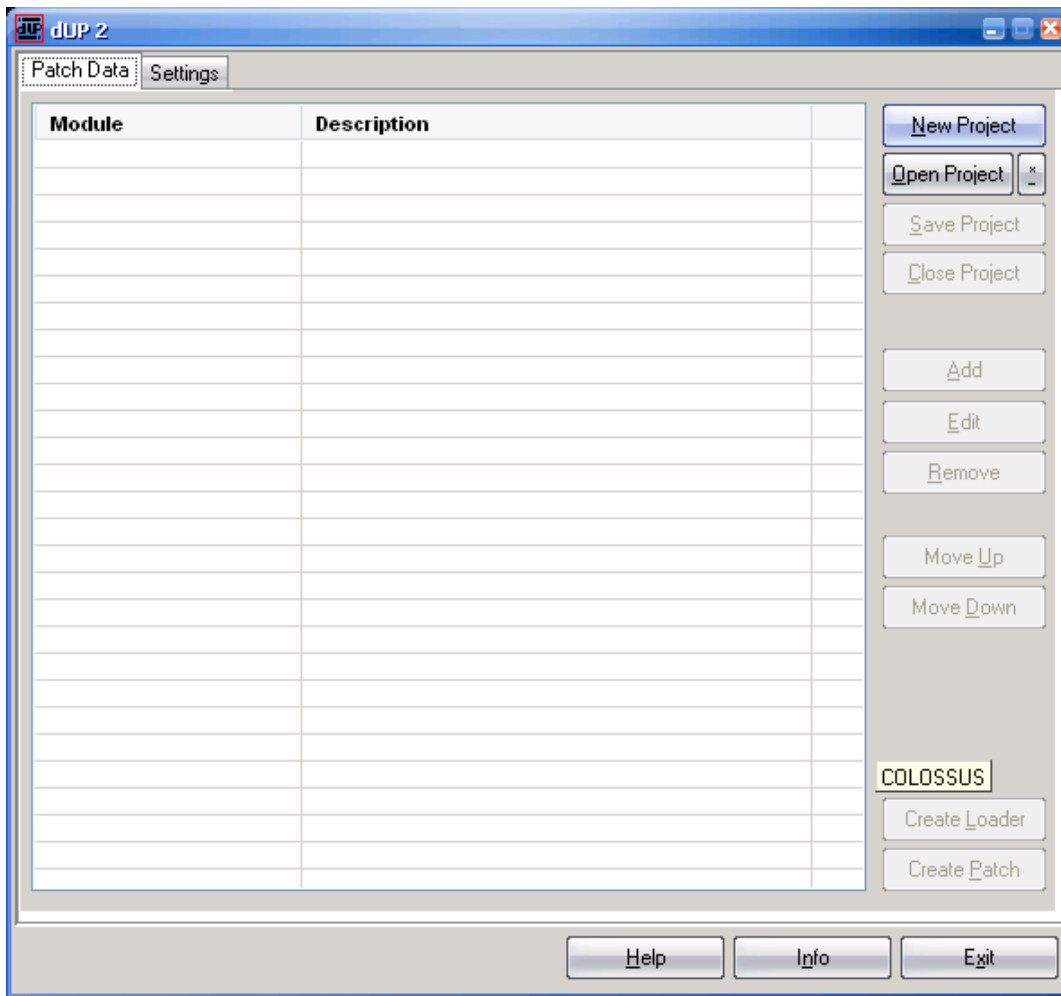
Ahora copiamos los bytes viejos y modificamos el verde con “Crack Latinos”



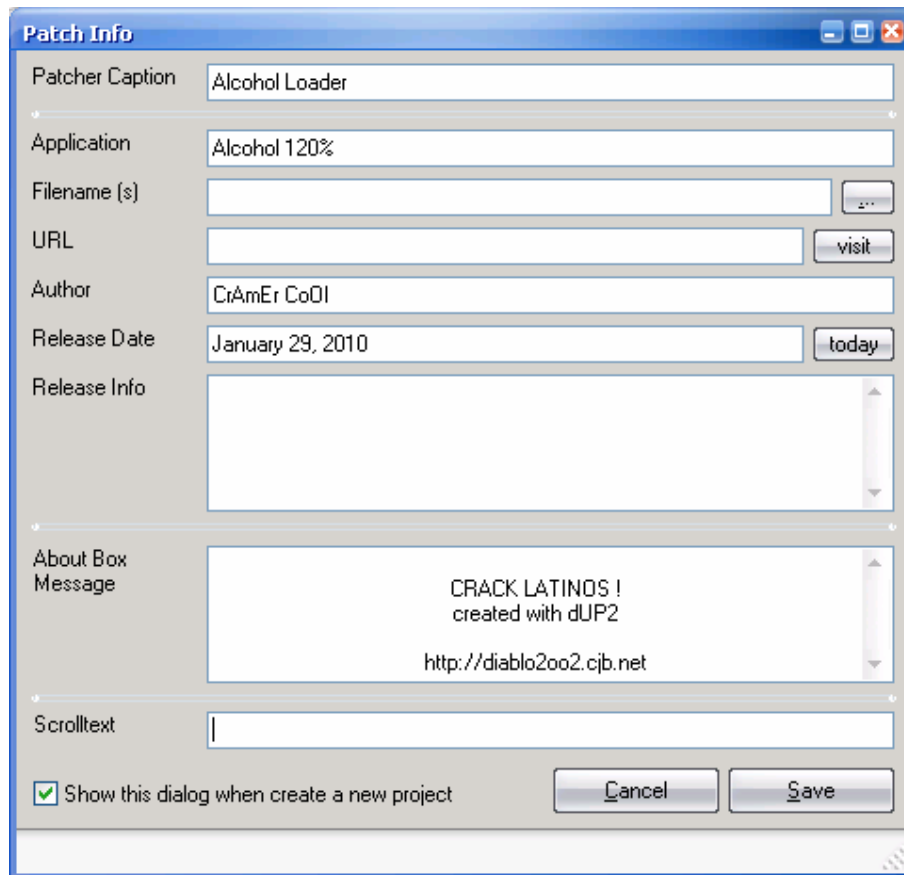
Pero que casualidad coincidió perfecto jeje, ahora copiamos los nuevos bytes, para esto siempre es bueno hacer una tablita jeje.

Dirección	Viejos	Nuevos
0064285B	54	43
0064285C	72	72
0064285D	69	61
0064285E	61	63
0064285F	6C	6B
00642860	20	20
00642861	56	4C
00642862	65	61
00642863	72	74
00642864	73	69
00642865	69	6E
00642866	6F	6F
00642867	6E	73

Listo ahora a hacer el loader, como no soy programador, lo que haremos será utilizar las herramientas que existen en el mercado:-b, para mi caso **dup2.exe** que lo anexo al tute, lo corremos y vemos esto.



Esta ventana nos va a ayudar a solucionar nuestro problema:-9, ya solo queda hacer clic en New Project y llenamos la siguiente ventana:



The 'Patch Info' dialog box is a standard Windows-style window with a blue title bar. It contains several input fields and buttons. The fields are: 'Patcher Caption' (Alcohol Loader), 'Application' (Alcohol 120%), 'Filename (s)' (empty), 'URL' (empty), 'Author' (CrAmEr CoDI), 'Release Date' (January 29, 2010), and 'Release Info' (empty). There are also buttons for 'visit' and 'today'. Below these is an 'About Box Message' section with a text area containing 'CRACK LATINOS !', 'created with dUP2', and 'http://diablo2oo2.cjb.net'. At the bottom, there is a 'Scrolltext' field, a checkbox labeled 'Show this dialog when create a new project' which is checked, and 'Cancel' and 'Save' buttons.

Patch Info

Patcher Caption: Alcohol Loader

Application: Alcohol 120%

Filename (s):

URL:

visit

Author: CrAmEr CoDI

Release Date: January 29, 2010

today

Release Info:

About Box Message:

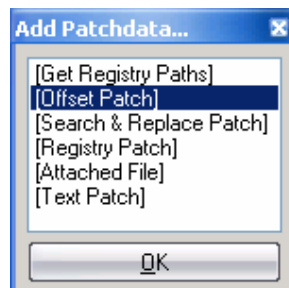
CRACK LATINOS !
created with dUP2
http://diablo2oo2.cjb.net

Scrolltext:

☒ Show this dialog when create a new project

Cancel Save

Ahora clic en save y clic en Add y en Offset Match



Y por ultimo clic en edit y agregamos los bytes de las tablas, miren las opciones que tiene habilitadas en el dialogo.

Offset Patchdata

Target File
 File: Alcohol.exe
 Old Filesize: 00186780
 New Filesize:
 CRC32: 4C40A041
☒ enable CRC32 check
☐ don't check original (old) bytes
☐ Try patching used file
 Patchmode:
☐ Standard
☒ VirtualAddress Mode [PE file]

Compare Files
 Original File:
 Patched File:
 Compare

Virtual Address	Old Byte	New Byte
00401A63	33	B0
00401A64	C0	01
0064285B	54	43
0064285C	72	72
0064285D	69	61
0064285E	61	63
0064285F	6C	6B
00642860	20	20
00642861	56	4C
00642862	65	61
00642863	72	74
00642864	73	69
00642865	69	6E
00642866	6F	6F
00642867	6E	73

Add and Edit
 Offset:
 Original Byte:
 Patched Byte:
 Edit Add
 Remove Clear List

MemCheck [for Loaders]
☐ Enable MemCheck ?
 MemoryAddress:
 MemoryValue:
 Up Down
 Items: 15

File to load in Ollydbg
 Follow in Ollydbg Cancel Save

Por ultimo clic en save y en Create Loader, le ponen nombre y lo probamos, BANG☺,

Otro que marchó y muy fácil por cierto jeje, pero un poco demoroso, por cierto algo no está cambiado y es el nombre del registro, a este no lo puedo cambiar porque está comprimida la aplicación y la dirección cambia de acuerdo a la máquina y al código del UPX☺, pero eso no importa, el programa es totalmente funcional y sin restricciones...



Esto es todo, el presente tute va dedicado a todas las personas que quieran aprender algo, no es mucho lo que se, pero como dicen algo es algo☺.

Saludos a todos en la lista ojala les guste, comentarios, sugerencias, criticas a mi mail heavyvinicio@gmail.com, gracias por leer hasta la próxima.

Agradecimientos:

A todos los que escriben tutes, por ustedes la comunidad crece constantemente jeje.

Anexo

✓ DUP2

