

Programa	Ultra iso 9.5.2.2836
Protección	Aspack v2.12
Objetivo	Más que desempacar descifrar el md5 para nuestro usuario y contraseña
Web/descarga	<a href="https://www.ezbsystems.com/ultraiso/">https://www.ezbsystems.com/ultraiso/</a>
Dificultad	3 de 10
Plataforma	Windows 10
Lenguaje	Borland c++
Herramientas	Ollydbg, calculadora de Windows, ASPackDie_1.41
Cracker	Carlos Ismael
Fecha	23 de octubre del 2015
Contacto	<a href="mailto:Charly-091@hotmail.com">Charly-091@hotmail.com</a>

Bueno aquí estamos de nuevo como ya saben mi nombre es Carlos Ismael Tun Tun originario de Yucatan Mexico, pues este es mi siguiente tutorial, como siempre he dicho espero que no sea el último.

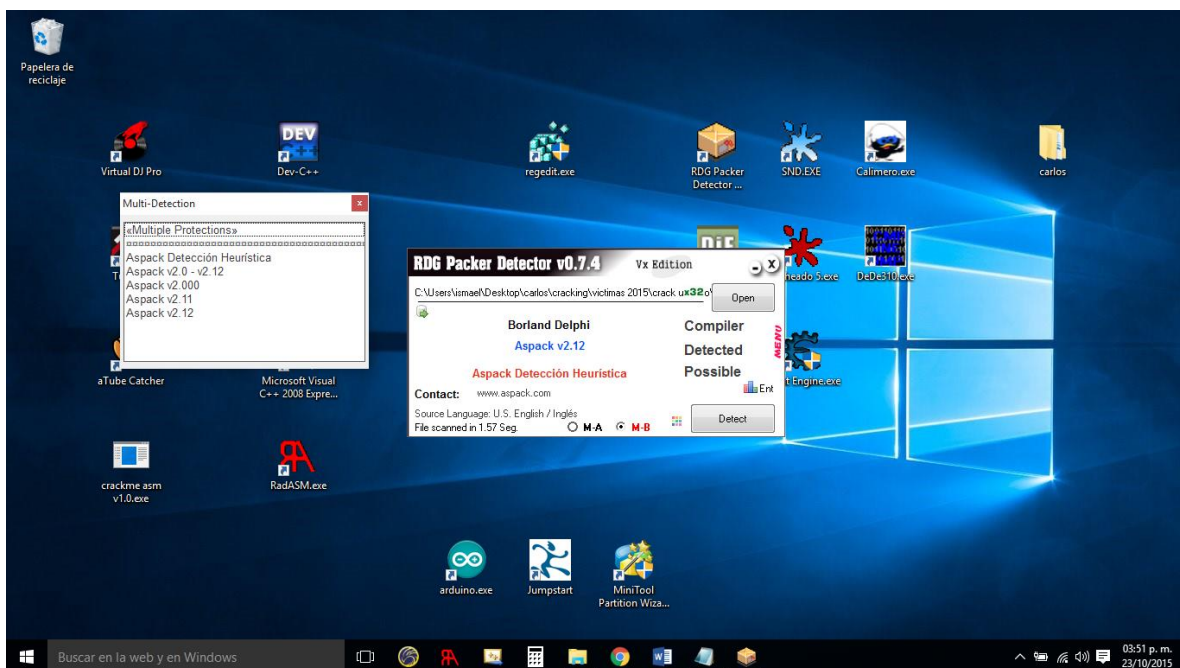
Es grato para mí escribir para difundir el conocimiento y más si se trata de este tipo de cosas, el cual es muy poco conocido en nuestro país.

Disclaimer

Este tutorial fue escrito con fines educativos y para difundir el conocimiento, el programa que se uso es solo para puro ejemplo.

Pues comencemos:

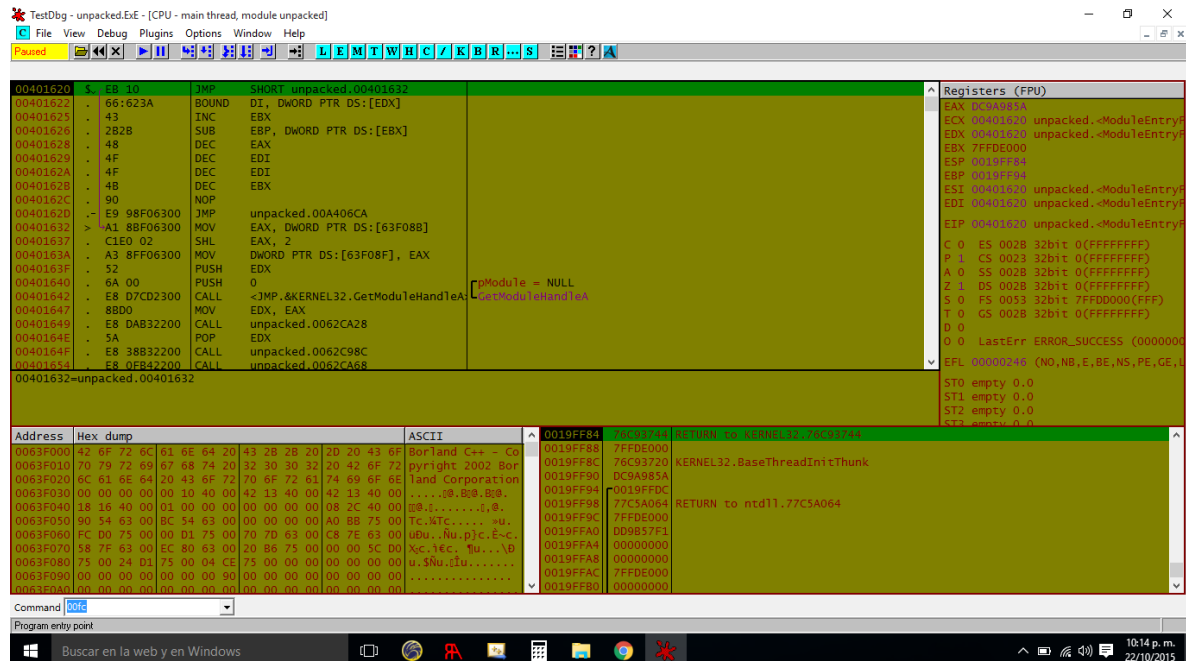
Lo primero es pasarle el detector de packers. Y esto es lo que nos muestra.



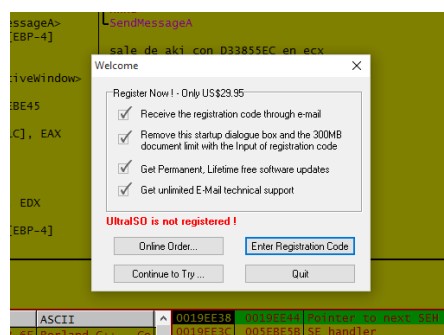
Vemos que está protegido con el aspack v2.12, me dio FIACA desempacarlo ya que sabemos cómo se hace, buscamos pushad breackpoint en esp dos veces F8 mas y ya, lo dumpeamos luego con el iRec lo preparamos y ya. Pero la verdad lo hice con el desempacador de akira (ASPackDie\_1.41) que lo descargamos de la web de Ricardo en:

<http://ricardonarvaja.info/WEB/OTROS/HERRAMIENTAS/A-B-C-D-E/> lo usamos con el programa y ya podemos seguir sale??

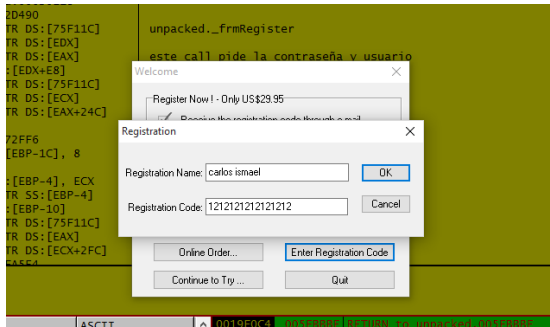
Entry point del programa....



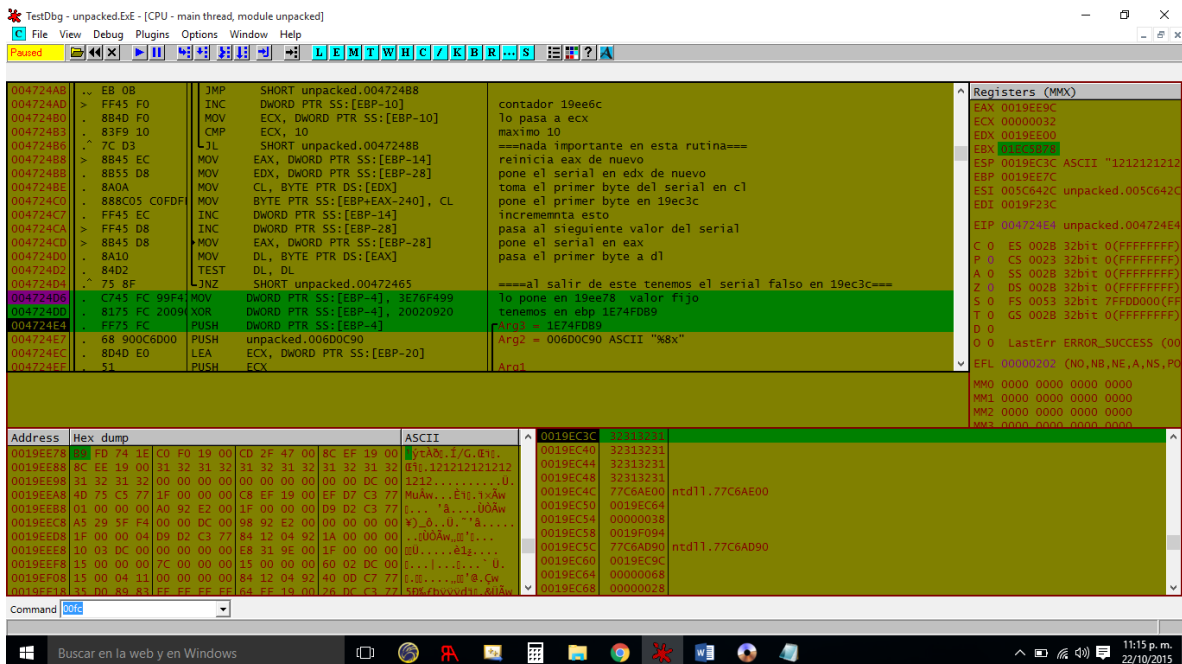
Iniciamos el programa



Metemos unos datos así jejejeje



Aquí se genera un valor muy importante: 1e74fdb9, el cual es una constante, siempre va a ser el mismo para todos los nombres que metamos.



En las dos siguientes imágenes lo transforma en ASCII con los siguientes procedimientos, (son muy fáciles de entender, de seguro lo usare en alguna aplicación más tarde jejejejeje ; ) )

The screenshot shows a debugger window with the following components:

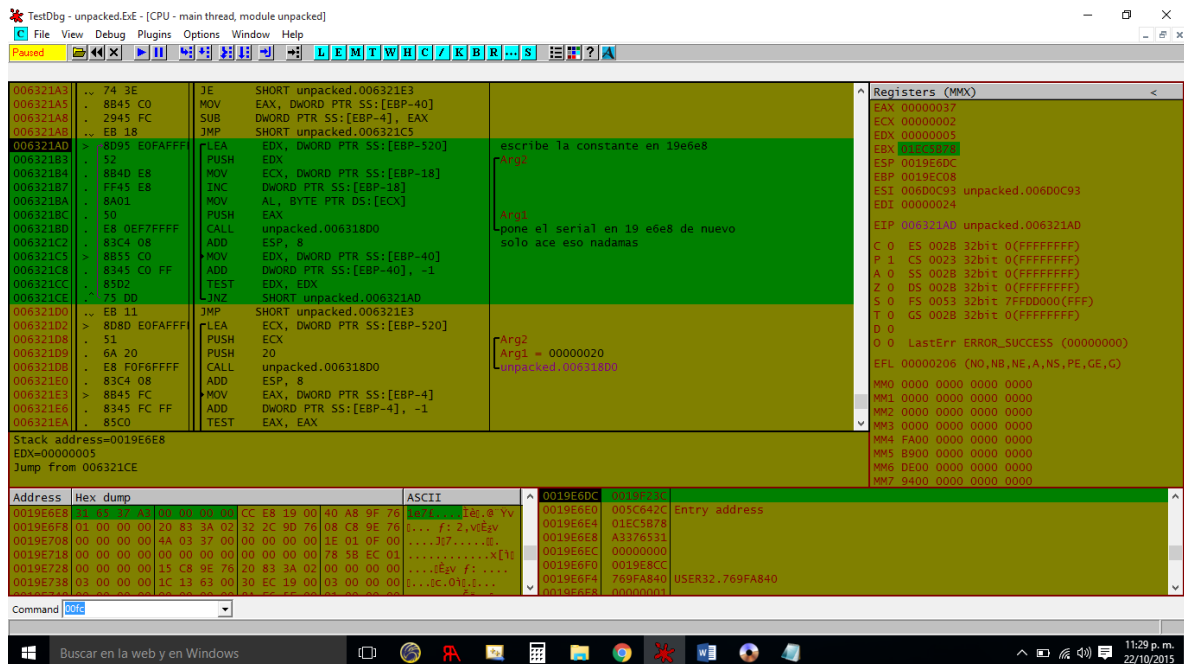
- Assembly View:** Displays assembly instructions with their addresses and hex values. The selected instruction is `JMP SHORT unpacked.006351E9` at address `006351D0`. The instruction is highlighted in blue.
- Registers (MMX):** A list of registers and their values. The `EAX` register is highlighted in blue, showing the value `00000000`.
- Hex dump:** A view of the memory dump showing the data being processed. The selected instruction is `JMP SHORT unpacked.006351E9` at address `006351D0`.
- Command:** A text box for entering commands.

Tengo seleccionado donde lo va guardar y también sabemos que lo saca del procedimiento de la imagen anterior. Queda claro??

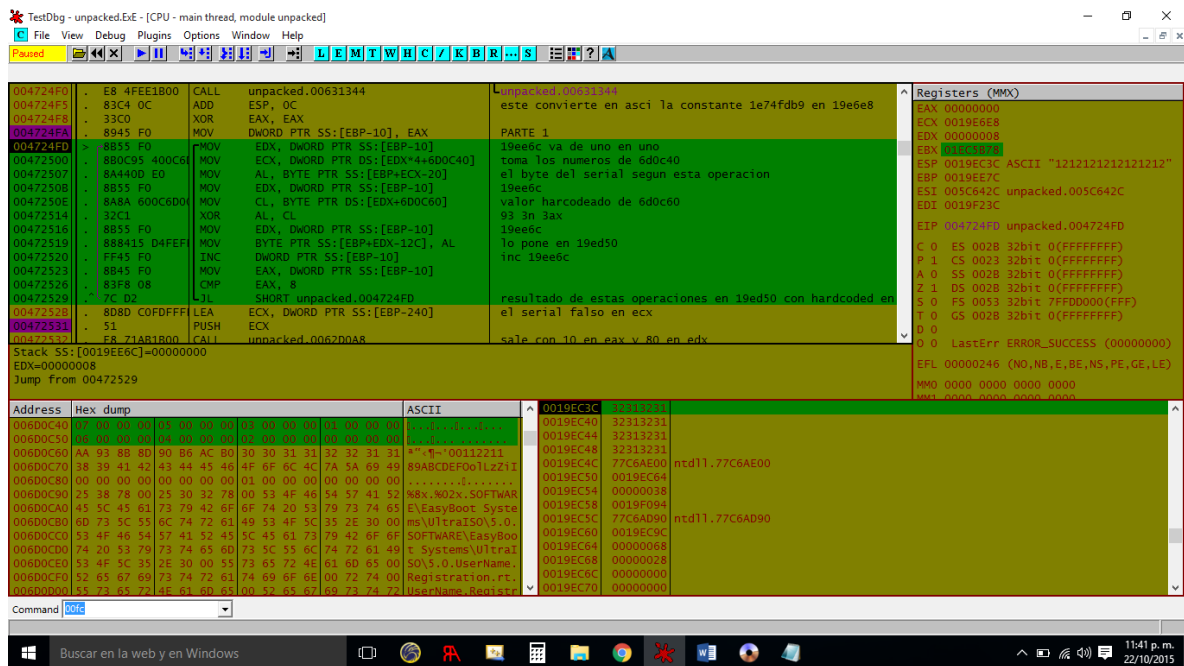
The screenshot shows a debugger window with the following components:

- Assembly View:** Displays assembly instructions with their addresses and hex values. The selected instruction is `JMP SHORT unpacked.006351E9` at address `006351D0`. The instruction is highlighted in blue.
- Registers (MMX):** A list of registers and their values. The `EAX` register is highlighted in blue, showing the value `00000000`.
- Hex dump:** A view of the memory dump showing the data being processed. The selected instruction is `JMP SHORT unpacked.006351E9` at address `006351D0`.
- Command:** A text box for entering commands.

Pasa el valor contante a la dirección siguiente esto con el propósito de que no nos perdamos lo que hace el programa, les recuerdo que tiene mucho código que repite y repite las cosas o se la pasa copiando los valores en direcciones diferentes varias veces, lo digo de una vez por si acaso se quedan con “¿y de donde saco ese valor?” jejejejee ok?? Bueno sigamos..



En la siguiente imagen seleccionamos la parte en la que trabaja con la constante



Estos números son los que tomara para calcular la posición siguiente el cual se traduce como

Posición = Contador\*4 + dirección de memoria

0\*4=0 posición 7

1\*4=4 posición 5

2\*4=8 posición 3

3\*4=12 posición 1

4\*4=16 posición 6

5\*4=20 posición 4

6\*4=24 posición 2

7\*4=28 posición 00

Estos son los números obtenidos en cada vuelta

Estos son las posiciones ya ordenados (yo lo ilustro así ya que el programa no trabaja de esta manera)

9d4ebf71= 39 64 34 65 62 66 37 31

00472507	< 8A40D E0	MOV	AL, BYTE PTR SS:[EBP+ECX-20]	el byte del serial segun esta operacion	EBP 0
00472508	< 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	ESI 0
00472509	< 8A8A 600C60	MOV	CL, BYTE PTR DS:[EDX+600C60]	valor harcodeado de 600C60	EDI 0
00472514	< 32C1	XOR	AL, CL	93 3n 3ax	EIP 0
00472516	< 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	
00472519	< 888415 D4FEF	MOV	BYTE PTR SS:[EBP+EDX-12C], AL	lo pone en 19ed50	
00472520	< FF45 F0	INC	DWORD PTR SS:[EBP-10]	inc 19ee6c	
00472523	< 8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]		
00472526	< 83F8 08	CMP	EAX, 8		
00472529	< 7C D2	JL	SHORT unpacked.004724FD	resultado de estas operaciones en 19ed50 con hardcoded en	
0047252B	< 8D8D C0DFFF	LEA	ECX, DWORD PTR SS:[EBP-240]	el serial falso en ecx	
00472531	< 51	PUSH	ECX		
00472532	< F8 71A81B00	CALL	unpacked.0062D0A8	sale con 10 en eax v 80 en edx	

Address	Hex dump	ASCII		0019EC3C	32313231
00600C60	AA 93 8B 8D 90 B6 AC B0	30 30 31 31 32 32 31 31	00112711	0019EC40	32313231
00600C70	38 39 41 42 43 44 45 46	4F 6F 6C 4C 7A 5A 69 49	89ABCDEF06112211	0019EC44	32313231
00600C80	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....I.....	0019EC48	32313231
00600C90	25 38 78 00 25 30 32 78	00 53 4F 46 54 52 41 52	88x.802x.S0FTWAR	0019EC4C	77C6AE00

La operación siguiente lo toma como posición para el byte de la constante que vamos a tomar la cual la traducimos como:

Byte a tomar=dirección de memoria+posición

Estos son los valores que toma según la operación anterior

AA 93 8B 8D 90 B6 AC B0

004724FA	< 8945 F0	MOV	DWORD PTR SS:[EBP-10], EAX	PARTE 1	EDX 00000000
004724FD	> 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c va de uno en uno =contador	EBX 01EC3B78
00472500	< 8B0C95 400C6	MOV	ECX, DWORD PTR DS:[EDX*4+600C40]	toma los numeros de 60c40	ESP 0019EC3C ASCII
00472507	< 8A40D E0	MOV	AL, BYTE PTR SS:[EBP+ECX-20]	el byte del serial segun esta operacion	EBP 0019EE7C
00472508	< 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	ESI 005C642C unpacked
0047250E	< 8A8A 600C60	MOV	CL, BYTE PTR DS:[EDX+600C60]	valor harcodeado de 600C60	EDI 0019F23C
00472514	< 32C1	XOR	AL, CL	93 3n 3ax	EIP 00472507 unpacked
00472516	< 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	
00472519	< 888415 D4FEF	MOV	BYTE PTR SS:[EBP+EDX-12C], AL	lo pone en 19ed50	
00472520	< FF45 F0	INC	DWORD PTR SS:[EBP-10]	inc 19ee6c	
00472523	< 8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]		
00472526	< 83F8 08	CMP	EAX, 8		
00472529	< 7C D2	JL	SHORT unpacked.004724FD	resultado de estas operaciones en 19ed50 con hardcoded en	
0047252B	< 8D8D C0DFFF	LEA	ECX, DWORD PTR SS:[EBP-240]	el serial falso en ecx	
00472531	< 51	PUSH	ECX		
00472532	< F8 71A81B00	CALL	unpacked.0062D0A8	sale con 10 en eax v 80 en edx	

Address	Hex dump	ASCII		0019EC3C	32313231
0019EE5C	31 65 37 34 06 54 07 35	00 37 5B 00 10 00 00 00	00472507 [.]...	0019EC40	32313231
0019EE6C	00 00 00 00 88 FD 19 00	05 D4 63 00 89 FD 74 1E	....8..0C.7yt	0019EC44	32313231
0019EE7C	25 38 78 00 25 30 32 78	00 53 4F 46 54 52 41 52	88x.802x.S0FTWAR	0019EC48	32313231

Toma valores de ahí según el contador

Después les hace xor

AA 93 8B 8D 90 B6 AC B0

xor

39 64 34 65 62 66 37 31

00472500	8B0C95 400C6	MOV	ECX, DWORD PTR DS:[EDI*4+600C40]	toma los numeros de 600c40	ESP 0019EC3C
00472507	8A440D E0	MOV	AL, BYTE PTR SS:[EBP+ECX-20]	el byte del serial segun esta operacion	EBP 0019ED52
0047250B	8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	ESI 005C642C u
00472510	8B55 00000000	MOV	CL, BYTE PTR DS:[10x+800c40]	valor hardcodeado de 800c40	EDI 0019F23C
00472514	32C1	XOR	AL, CL	93 3n 3ax	EIP 00472510 u
00472516	8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	C 0 ES 002B 3
00472519	8B415 D4FEF	MOV	BYTE PTR SS:[EBP+EDX-12C], AL	lo pone en 19ed50	P 1 CS 0023 3
00472520	FF45 F0	INC	DWORD PTR SS:[EBP-10]	inc 19ee6c	A 0 SS 002B 3
00472523	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]		Z 1 DS 002B 3
00472526	83F8 08	CMP	EAX, 8		S 0 FS 0053 3
00472529	7C D2	JL	SHORT unpacked.0047252F	resultado de estas operaciones en 19ed50 con hardcoded en	T 0 GS 002B 3
0047252B	8D8D C0FDFFF	LEA	ECX, DWORD PTR SS:[EBP-240]	el serial falso en ecx	D 0
00472531	51	PUSH	ECX		D 0 LastErr E
00472532	FA 71A81B00	CALL	unpacked.0062D0A8	sale con 10 en eax v 80 en edx	EFL 00000246 (
Address	Hex dump	ASCII			
006D0C60	AA 93 8B 8D 90 B6 AC B0	30 30 31 31 32 32 31 31	0019EC3C	32313231	
006D0C70	3B 39 41 42 43 44 45 46	4F 4F 4F 4F 4F 4F 4F 4F	0019EC40	32313231	
006D0C80	00 00 00 00 00 00 00 00	01 00 00 00 00 00 00 00	0019EC44	32313231	
006D0C90	25 38 78 00	25 30 32 78 00 53 4F 46	0019EC48	32313231	
		88x.X02x.S0FTwAR	0019EC4C	77C6AE00	ntd11.77C6AE00

El resultado lo guarda ahí..

Resultado= 93 F7 BF E8 F2 D0 9B 81 la cual queda en 19ED50

A partir de aquí si queremos hacer nuestro keygen podemos tomar este valor como constante no es necesario hacer todo este maraño... jejejejejeje verdad???

0047250B	8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	ESI 005C642C un
00472510	8A8A 600C6D0	MOV	CL, BYTE PTR DS:[EDX+600C6D0]	valor hardcodeado de 600c6d0	EDI 0019F23C
00472514	32C1	XOR	AL, CL	93 3n 3ax	EIP 00472510 un
00472516	8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	19ee6c	C 0 ES 002B 3
00472519	8B415 D4FEF	MOV	BYTE PTR SS:[EBP+EDX-12C], AL	lo pone en 19ed50	P 1 CS 0023 3
00472520	FF45 F0	INC	DWORD PTR SS:[EBP-10]	inc 19ee6c	A 0 SS 002B 3
00472523	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]		Z 0 DS 002B 3
00472526	83F8 08	CMP	EAX, 8		S 1 FS 0053 3
00472529	7C D2	JL	SHORT unpacked.0047252F	resultado de estas operaciones en 19ed50 con hardcoded en	T 0 GS 002B 3
0047252B	8D8D C0FDFFF	LEA	ECX, DWORD PTR SS:[EBP-240]	el serial falso en ecx	D 0
00472531	51	PUSH	ECX		D 0 LastErr ER
00472532	FA 71A81B00	CALL	unpacked.0062D0A8	sale con 10 en eax v 80 en edx	EFL 00000282 (N
Address	Hex dump	ASCII			
0019ED50	93 F7 BF E8 F2 D0 9B 81	88 ED 19 00 40 A8 9F 76	0019EC3C	32313231	
0019ED60	60 5D 93 83 FE FF FF FF	C8 ED 19 00 70 43 9D 76	0019EC40	32313231	
0019ED70	D5 OF 36 00 00 00 00 00	06 00 00 00 01 00 00 00	0019EC44	32313231	
		06x.....2....2....	0019EC48	32313231	

Parte 2 en la dirección 472559

Toma los bytes del serial que pusimos según el contador

00472550	> 8B55 F0	MOV	EDX, DWORD PTR SS:[EBP-10]	contador en 19ee6c	ECX 00000000
0047255E	8A8C15 C0FDE	MOV	CL, BYTE PTR SS:[EBP+EDX-240]	byte del serial en 19ec3c	EDX 00000000
00472565	8B45 EC	MOV	EAX, DWORD PTR SS:[EBP-14]		EBX 036C5B18
00472568	8A9405 D4FEF	MOV	DL, BYTE PTR SS:[EBP+EAX-12C]	valor tomado de 19ed50	ESP 0019EC3C ASCII
0047256F	32CA	XOR	CL, DL		EBP 0019EE7C
00472571	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	19ee6c	ESI 005C642C unpack
00472574	8B8C05 D4FDE	MOV	BYTE PTR SS:[EBP+EAX-22C], CL	el resultado del xor va a 19ec50	EDI 0019F23C
0047257B	FF45 EC	INC	DWORD PTR SS:[EBP-14]		EIP 00472570 unpack
0047257E	8B55 EC	MOV	EDX, DWORD PTR SS:[EBP-14]		C 1 ES 002B 32bit
00472581	83FA 08	CMP	EDX, 8		P 1 CS 0023 32bit
00472584	7C 05	JL	SHORT unpacked.0047258B		A 0 SS 002B 32bit
00472586	33C9	XOR	ECX, ECX		Z 0 DS 002B 32bit
00472588	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX		S 1 FS 0053 32bit
0047258B	FF45 F0	INC	DWORD PTR SS:[EBP-10]		T 0 GS 002B 32bit
00472591	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	contador	D 0
00472594	8B55 F4	MOV	EDX, DWORD PTR SS:[EBP-C]	limite 10	D 0 LastErr ERROR_
00472596	7C 03	JL	SHORT unpacked.0047259B	xor del serial falso con valores tomados de 19ed50	EFL 00000287 (N0,B)
00472598	33C9	XOR	ECX, ECX	resultado del trabajo esta en 19ec50	ST0 empty 0.0
0047259A	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX		ST1 empty 0.0
0047259D	33C0	XOR	EAX, EAX		ST2 empty 0.0
Stack SS:[0019ED52]=19					ST3 empty 0.0
Stack SS:[0019EC3C]=31 ('1')					ST4 empty 125.00000
CL=00					
Address	Hex dump	ASCII			
0019EC3C	31 31 31 31 31 31 31 31	31 31 31 31 31 31 31 31	0019EC3C	32313231	
0019EC4C	00 AE C6 77 64 EC 19 00	38 00 00 00 94 F0 19 00	0019EC40	32313231	
		..4wdN.8..."0c.	0019EC44	32313231	
			0019EC48	32313231	

Ahí es donde toma los bytes de la operación anterior luego hace xor entre los dos valores y el resultado lo guarda en: 19ec50 como podemos ver en la imagen

0047255E	8A8C15 C0FDF	MOV	CL, BYTE PTR SS:[EBP+EDX-240]	byte del serial en 19ec3c
00472565	8B45 EC	MOV	EAX, DWORD PTR SS:[EBP-14]	
00472568	8A8C05 D4FEE	MOV	DL, BYTE PTR SS:[EBP+EAX-12C]	valor tomado de 19ed50
0047256F	32CA	XOR	CL, DL	
00472571	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	19ee5c
00472574	8B8C05 D4FDF	MOV	BYTE PTR SS:[EBP+EAX-22C], CL	el resultado del xor va a 19ec50
0047257B	FF45 EC	INC	DWORD PTR SS:[EBP-14]	
0047257E	8B55 EC	MOV	EDX, DWORD PTR SS:[EBP-14]	
00472581	83FA 08	CMP	EDX, 8	
00472584	7C 05	JL	SHORT unpacked.0047258B	
00472586	33C9	XOR	ECX, ECX	
00472588	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX	
0047258B	FF45 F0	INC	DWORD PTR SS:[EBP-10]	
0047258E	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	contador
00472591	8B55 F4	MOV	EDX, DWORD PTR SS:[EBP-C]	límite 10
00472594	3BC2	CMP	EAX, EDX	
00472596	7C C3	JL	SHORT unpacked.0047259B	
00472598	33C9	XOR	ECX, ECX	xor del serial falso con valores tomados de 19ed50
0047259A	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX	resultado del trabajo esta en 19ec50
0047259D	33C0	XOR	EAX, EAX	

Para los que no se acuerden, en la primera operación, solo género 8 bytes, las cuales tomo de 1e74fdb9, ok??

Eso quiere decir que no alcanza para hacer xor a todos los valores del serial falso porque nos da un total de 16 bytes, entonces esa instrucción que hemos seleccionado en la imagen reinicia el contador y comienza a tomar los valores desde el principio pero no reinicia el contador que hace que tomemos byte por byte del serial falso, ok??? No se pierdan jejeje

00472574	8B8C05 D4FDF	MOV	BYTE PTR SS:[EBP+EAX-22C], CL	el resultado del xor va a 19ec50
0047257B	FF45 EC	INC	DWORD PTR SS:[EBP-14]	
0047257E	8B45 EC	MOV	EDX, DWORD PTR SS:[EBP-14]	
00472581	83FA 08	CMP	EDX, 8	
00472584	7C 05	JL	SHORT unpacked.0047258B	
00472586	33C9	XOR	ECX, ECX	
00472588	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX	
0047258B	FF45 F0	INC	DWORD PTR SS:[EBP-10]	
0047258E	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	contador
00472591	8B55 F4	MOV	EDX, DWORD PTR SS:[EBP-C]	límite 10
00472594	3BC2	CMP	EAX, EDX	
00472596	7C C3	JL	SHORT unpacked.0047259B	
00472598	33C9	XOR	ECX, ECX	xor del serial falso con valores tomados de 19ed50
0047259A	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX	resultado del trabajo esta en 19ec50
0047259D	33C0	XOR	EAX, EAX	

Eso quiere decir que los valores a ser xoreados quedan de esta manera:

0019EC3C 31 32 31 32 31 32 31 32 31 32 31 32 31 32 31 32 1212121212121212

0019ED50 93 F7 BF E8 F2 D0 9B 81 93 F7 BF E8 F2 D0 9B 81 “÷¿èòD”

Para darnos como resultado:

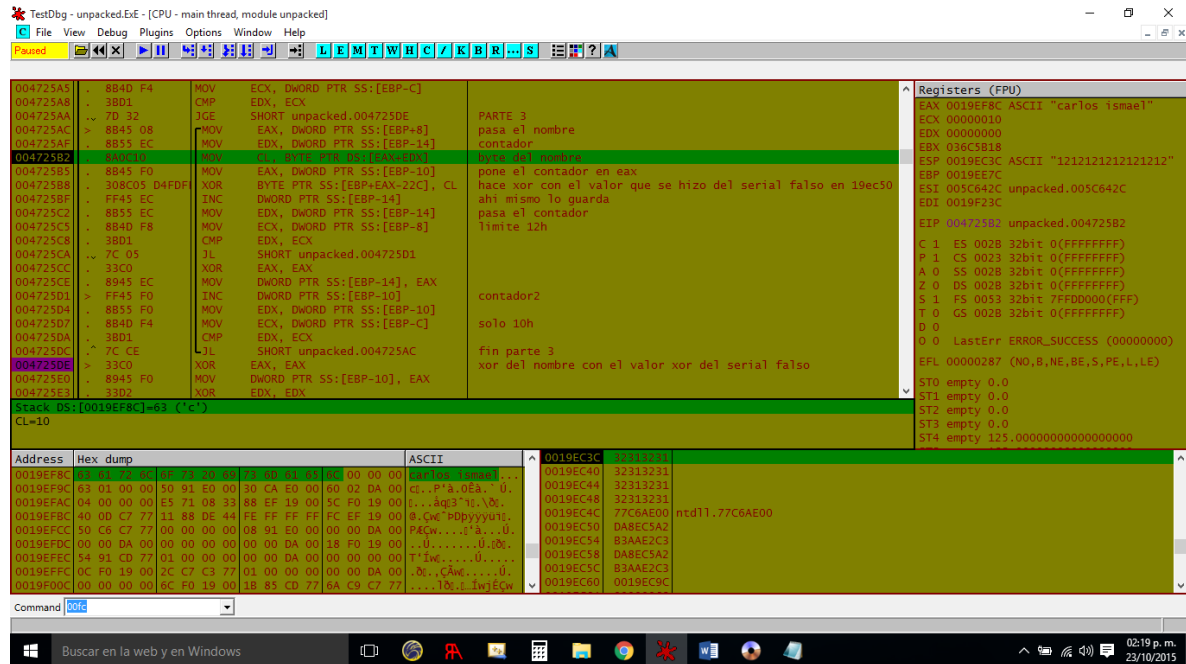
0019EC50 A2 C5 8E DA C3 E2 AA B3 A2 C5 8E DA C3 E2 AA B3 çĂŽŰĂă³çĂŽŰĂă³

Bueno pues ahí está en la imagen de abajo

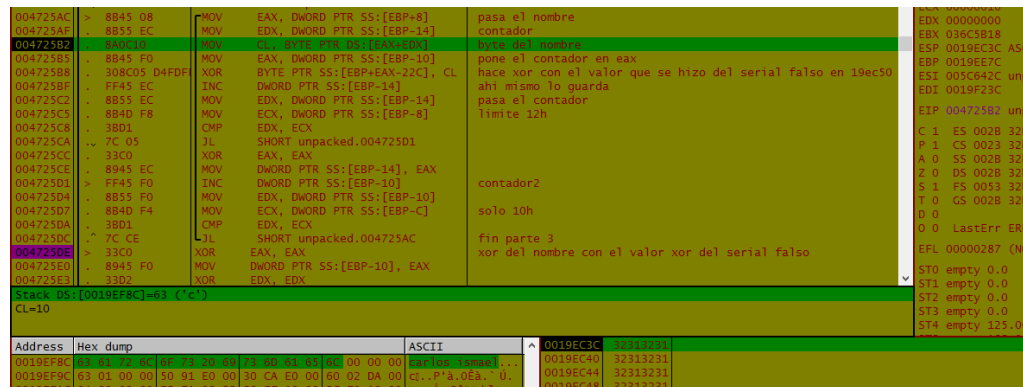
0047258B	FF45 F0	INC	DWORD PTR SS:[EBP-10]	contador
0047258E	8B45 F0	MOV	EAX, DWORD PTR SS:[EBP-10]	límite 10
00472591	8B55 F4	MOV	EDX, DWORD PTR SS:[EBP-C]	
00472594	3BC2	CMP	EAX, EDX	
00472596	7C C3	JL	SHORT unpacked.0047259B	
00472598	33C9	XOR	ECX, ECX	xor del serial falso con valores tomados de 19ed50
0047259A	894D EC	MOV	DWORD PTR SS:[EBP-14], ECX	resultado del trabajo esta en 19ec50
0047259D	33C0	XOR	EAX, EAX	

Ahora pasamos a la parte 3





En la imagen anterior vemos que toma el byte del nombre según el contador y lo pone en cl



Luego vemos que toma el byte de la operación anterior y hace xor byte a byte con el del nombre que introducimos, esta vez no lo cambia de lugar e deja ahí mismo, como podemos ver en la imagen de abajo:

The screenshot displays the x64dbg debugger interface. The main window shows assembly code with instructions like MOV, CMP, JGE, and MOV. The right pane shows the 'Registers (FPU)' section with values for EAX, EDX, EBX, ESP, EBP, ESI, EDI, and EIP. The bottom pane shows the 'Stack' section with memory addresses and hex values. The command line at the bottom shows 'C:\Program Files\Trend Micro\AMSS\AMSS.exe'.

Esto lo podemos traducir en nuestro keygen como:

```
0019EF8C 63 61 72 6C 6F 73 20 69 73 6D 61 65 6C
```

Carlos ismael

```
0019EC50 A2 C5 8E DA C3 E2 AA B3 A2 C5 8E DA C3 E2 AA B3
```

čĀŽÚĀā³čĀŽÚĀā³

Como podemos ver nuestro nombre solo tiene 13 caracteres y el valor anterior 16, lo que el programa hace para remediarlo es, con la instrucción que seleccionamos en la imagen de bajo, reinicia el contador y así toma los caracteres desde el principio de nuevo para completar los 16 que necesita.

00472581	-	F745 EC	INC	DWORD PTR SS:[EBP-14]	ahi mismo lo guarda	ESI 005C642C unp
004725C2	-	8855 EC	MOV	EDX, DWORD PTR SS:[EBP-14]	pasa el contador	EDI 0019F23C
004725C5	-	884D F8	MOV	ECK, DWORD PTR SS:[EBP-8]	limite 12h	
004725C8	-	3BD1	CMP	EDX, ECX		EIP 004725DC unp
004725CA	-	7C 05	JL	SHORT unpacked.004725D1		C 1 E5 002B 32b
004725CC	-	33C0	XOR	EAX, EAX		P 1 ES 0023 32b
004725CE	-	8845 EC	MOV	DWORD PTR SS:[EBP-14], EAX		A 0 55 002B 32b
004725D1	-	F745 F0	INC	DWORD PTR SS:[EBP-10]	contador2	Z 0 05 002B 32b
004725D4	-	8855 F0	MOV	EDX, DWORD PTR SS:[EBP-10]		S 1 F5 002B 32b
004725D7	-	884D F4	MOV	ECK, DWORD PTR SS:[EBP-C]	solo 10h	T 0 D5 002B 32b
004725DA	-	3BD1	CMP	EDX, ECX		D 0
004725DE	-	7C CE	JL	SHORT unpacked.004725AC	fin parte 3	E 0 LastErr (ERR
004725E0	>	33C0	XOR	EAX, EAX	xor del nombre con el valor xor del serial falso	OP 00000287 (CNC
004725E0	-	8845 F0	MOV	DWORD PTR SS:[EBP-10], EAX		
004725E3	-	3302	XOR	EDX, EDX		ST0 empty 0.0
004725E5	-	8955 F0	MOV	DWORD PTR SS:[EBP-10], EDX		ST1 empty 0.0
004725D1=unpacked.004725D1						ST2 empty 0.0
						ST3 empty 0.0
						ST4 empty 125.00
Address	Hex dump	ASCII			0019EC3C 32313231	
0019EC30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	[unpacked.004725D1]			0019EC40 32313231	
0019EC60	5C EC 19 00 68 00 00 00 28 00 00 00 00 00 00 00	[unpacked.004725D1]			0019EC44 32313231	

Esta operación se transforma en

```
0019EF8C 63 61 72 6C 6F 73 20 69 73 6D 61 65 6C 63 61 72
```

carlos ismaelcar

```
0019EC50 A2 C5 8E DA C3 E2 AA B3 A2 C5 8E DA C3 E2 AA B3
```

čĀŽÚĀā³čĀŽÚĀā³

Dando como resultado:

0019EC50 C1 A4 FC B6 AC 91 8A DA D1 A8 EF BF AF 81 CB C1

Áxü¶-‘ŠÚÑ”i¿-â<sup>a3</sup>

A partir de aquí si se dan cuenta ya tenemos un valor md5 en los valores hexa que vemos son dos valores por cada posición dando un total de 32 valores o caracteres, ahora solo toca pasarlo a ASCII, vamos a ver como lo hace el programa:

Le aviso de una vez que tiene varios calls a lo mejor son para operaciones de comparación o cosas así no estoy seguro ya que me perdí en varios, solo vamos a ver la parte en la que convierte los valores sale? Bueno pues vamos a verlo. Es call de ahí es muy extenso...

Registers (FPU)

EAX	00000000
ECX	00000010
EDX	00000000
EBX	036C3818
ESP	0019EC3C ASCII "1212121212121212"
EBP	0019EE7C
ESI	005C642C unpacked.005C642C
EDI	0019F23C
EIP	004725E8 unpacked.004725E8
CS	00000000
DS	00000000
SS	00000000
ES	00000000
FS	00000000
GS	00000000
IOPL	00000000
LastError	ERROR_SUCCESS (00000000)
EFL	00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 125.000000000000000000

Stack 55:[0019EE6C]=00000000  
ECX=00000010  
Jump from 00472619

Address Hex dump ASCII

0019EC30	C1 A4 FC B6 AC 91 BA DA D1 A8 EF BF AF 81 C8 C1	Amu~'SUN 1 EA
0019EC31	9C EC 19 00 68 00 00 00 28 00 00 00 00 00 00	010.h...{.....
0019EC32	00 00 00 00 60 00 00 00 00 00 00 00 80 A5 37 02	.....'7
0019EC33	00 00 00 00 00 00 00 00 11 00 00 00 10 D2 F5 60	.....i00m
0019EC34	00 CC C7 77 00 E5 09 00 26 00 00 00 87 C8 9E 76	AlQwD...&...fEv
0019EC35	48 04 00 00 00 00 00 00 11 00 00 00 54 29 77 03	He.....T)e
0019EC36	4B 05 FF FF B4 02 00 00 01 00 00 00 54 C9 70 03	Kyy'...T)Pe
0019EC37	5C EE 19 00 15 C8 9E 76 D5 0F 36 00 45 72 08 33	\10.Egv06.Ert3
0019EC38	7C 7D 21 32 30 EE 19 00 AC EA 5F 00 4B 05 FF FF	}12010.-E...Kyy

Command

Buscar en la web y en Windows

Dentro del call vemos uno mas

00631362 |. E8 D1050000 CALL unpacked.00631938 ; \unpacked.00631938

Vemos que al entrar ahí un montón de instrucciones que quien sabe que hacen pero damos a "step over" o F8 yo les diré donde parar sale???

00631A23 |./F248D 2A1A63>||JMP DWORD PTR DS:[ECX\*4+631A2A] ; a la tercera vuelta de este jmp ya podemos ver lo que hace con el valor, que ya sabes, por supuesto, pero ay que ver como lo hacen ellos.

Registers (FPU)

EAX	00000000
ECX	00000010
EDX	00000000
EBX	036C3818
ESP	0019EC3C ASCII "1212121212121212"
EBP	0019EE7C
ESI	005C642C unpacked.005C642C
EDI	0019F23C
EIP	004725E8 unpacked.004725E8
CS	00000000
DS	00000000
SS	00000000
ES	00000000
FS	00000000
GS	00000000
IOPL	00000000
LastError	ERROR_SUCCESS (00000000)
EFL	00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 125.000000000000000000

Stack 55:[0019EE6C]=00000000  
ECX=00000010  
Jump from 00472619

Address Hex dump ASCII

0019EC30	C1 A4 FC B6 AC 91 BA DA D1 A8 EF BF AF 81 C8 C1	Amu~'SUN 1 EA
0019EC31	9C EC 19 00 68 00 00 00 28 00 00 00 00 00 00	010.h...{.....
0019EC32	00 00 00 00 60 00 00 00 00 00 00 00 80 A5 37 02	.....'7
0019EC33	00 00 00 00 00 00 00 00 11 00 00 00 10 D2 F5 60	.....i00m
0019EC34	00 CC C7 77 00 E5 09 00 26 00 00 00 87 C8 9E 76	AlQwD...&...fEv
0019EC35	48 04 00 00 00 00 00 00 11 00 00 00 54 29 77 03	He.....T)e
0019EC36	4B 05 FF FF B4 02 00 00 01 00 00 00 54 C9 70 03	Kyy'...T)Pe
0019EC37	5C EE 19 00 15 C8 9E 76 D5 0F 36 00 45 72 08 33	\10.Egv06.Ert3
0019EC38	7C 7D 21 32 30 EE 19 00 AC EA 5F 00 4B 05 FF FF	}12010.-E...Kyy

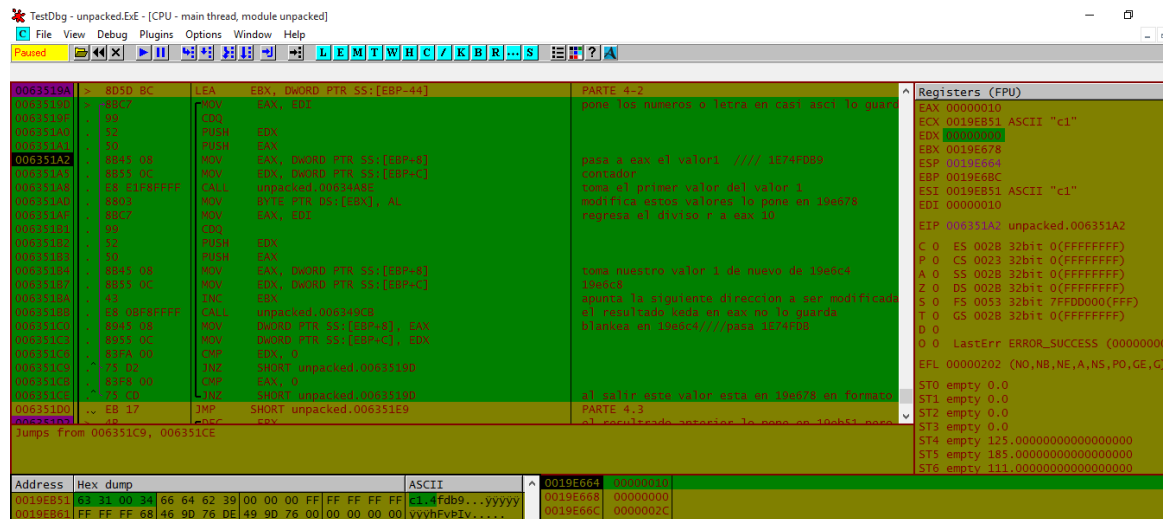
Command

Buscar en la web y en Windows

```
00631D3F |. 8B50 FC |MOV EDX, DWORD PTR DS:[EAX-4] ; toma un byte del
serial que está en 19ec38
```

```
00631D42 |. 8955 D0      |MOV     DWORD PTR SS:[EBP-30], EDX      ; lo pasa a 19ebd8 ==
valor 1
```

Bueno en la siguiente imagen ya podemos deducir lo que va a hacer



En este call

[illegible]

Los convierte a números o letras normales con esta operación

Ej:  $C1 / 10 == C$  resultado 1

Luego en la segunda vuelta lo vuelve a hacer

C / 10 = 0   resultado = C

Lo hace dos veces por cada byte y al salir guarda el resultado en:19e678 es un a manera de convertir a ASCII.

The screenshot shows a debugger window titled "TestDbg - unpacked.Exe - [CPU - main thread, module unpacked]". The assembly window displays instructions from address 00634ACC to 00634AF7. The registers window on the right shows the state of the FPU registers, with EAX, ECX, and EDI containing the value 00000000. The command window at the bottom shows the command "00634AF7".

Después con estas instrucciones lo guarda en: 19eb51.

Al salir de aquí también lo deja en minúsculas.

The screenshot shows a debugger window titled "TestDbg - unpacked.Exe - [CPU - main thread, module unpacked]". The assembly window displays instructions from address 006351C8 to 006351F9. The registers window on the right shows the state of the FPU registers, with EAX, ECX, and EDI containing the value 00000000. The command window at the bottom shows the command "006351F9".

Otro call que lo deja en 19e6e8

006321A8	EB 18	JMP	SHORT unpacked.006321C5		
006321AD	8061 E0FAFFF	LEA	ECX, DWORD PTR SS:[EBP-520]	scribete la constante en 19ed50	
006321B3	12	PUSH	ECX		
006321B4	8B40 EB	MOV	ECX, DWORD PTR SS:[EBP-18]	Arg2	
006321B7	FF43 EB	INC	DWORD PTR SS:[EBP-18]		
006321BA	8B03	MOV	AL, BYTE PTR DS:[ECX]		
006321BD	59	PUSH	EAX		
006321C0	EB 0077FFF	CALL	unpacked.006318D0	Arg1	
006321C2	83C4 08	ADD	ESP, 8	poner el serial en la casilla de nuevo solo sea eso nada mas	
006321C3	8B53 C0	MOV	ECX, DWORD PTR SS:[EBP-40]		
006321C8	8345 C0 FF	ADD	DWORD PTR SS:[EBP-40], -1		
006321CC	8302	TEST	EDX, EDX		
006321D0	72 00	JNB	unpacked.006321E3		
006321D2	806D E0FAFFF	JMP	SHORT unpacked.006321E3		
006321D8	51	PUSH	ECX		
006321D9	6A 20	PUSH	20	Arg2	
006321DB	EB F0F6FFF	CALL	unpacked.006318D0	Arg1 = 00000020 unpacked.006318D0	
006321E0	83C4 08	ADD	ESP, 8		
006321E3	8B45 FC	MOV	EAX, DWORD PTR SS:[EBP-4]		
006321E8	8345 FC FF	ADD	DWORD PTR SS:[EBP-4], -1		
006321EE	83C4	TEST	EAX, EAX		
Jump from 006321CE					
Address	Hex dump	ASCII	0019E6DC	0019E132	
0019E6E8	61 31 37 34 66 64 62 39 CC E8 19 00 40 A8 9F 76	174fdb91e0 yv	0019E6E0	005C642C	Entry address
0019E6F8	01 00 00 00 10 97 37 02 32 2C 9D 76 08 C8 9E 76	...-7 2,viEv	0019E6E4	038C5B18	

Dentro encontramos esto que es el encargado de copiarlo a esa dirección:

006318F8 |. 88040B MOV BYTE PTR DS:[EBX+ECX], AL

No le veo mucha ciencia.

00632265 |. E8 1EF6FFF CALL unpacked.00631888 luego con este call lo copia a 19ed50 con la instrucción seleccionada en la imagen. Entremos para ver como lo hace.

00631321	8B73 10	MOV	ESI, DWORD PTR SS:[EBP+10]		
00631324	8B50 0C	MOV	EBX, DWORD PTR SS:[EBP+4C]		
00631327	57	PUSH	EDI		
00631328	8B43 08	MOV	EAX, DWORD PTR SS:[EBP+8]	cantidad de bytes a copiar	
0063132B	10	PUSH	EAX	direccion de origen	
0063132C	8B16	MOV	EDX, DWORD PTR DS:[ESI]		
0063132E	12	PUSH	EDX	direccion de destino	
00631330	EB 7AB0E7F	CALL	unpacked.00631C88	lo copia a la ed 50	
00631334	83C4 0C	ADD	ESP, 0C		
00631337	011E	ADD	DWORD PTR DS:[ESI], EBX		
00631339	8B0E	MOV	ECX, DWORD PTR DS:[ESI]		
0063133B	C601 00	MOV	BYTE PTR DS:[ECX], 0		
0063133E	8BC3	MOV	EAX, EBX		
00631340	5E	POP	ESI		
00631341	5B	POP	EBX		
00631342	5D	POP	EBP		
00631343	C3	RETN			
00631344	55	PUSH	EBP		
00631345	8BEC	MOV	EBP, ESP		
00631347	8B45 08	MOV	EAX, DWORD PTR SS:[EBP+8]	resultado final del paso 3 19ed50	
0063134A	8B4D 08	LEA	ECX, DWORD PTR SS:[EBP+8]	19ed50	
EBX=00000002					
Address	Hex dump	ASCII	0019E6AC	00000000	
0019E6B0	05 31 34 00 00 9B 81 B8 ED 19 00 40 A8 9F 76	19ed50 yv	0019E680	0019E6E8	
0019E6B0	70 5D 9C 44 FE FF FF FF C8 ED 19 00 70 43 9D 76	p1ed0pyyE1.pCv	0019E684	0019E6D0	
0019E6D0	D5 0F 36 00 00 00 00 00 06 00 00 00 01 00 00 00	06.....	0019E688	006318B7	RETURN to unpacked.006318B7

Al pasar ese rep podemos ver que se copió a esta dirección

0062CEA1	8BCA	MOV	ECX, EDX		
0062CEA3	83E1 03	AND	ECX, 3		
0062CEA5	F3A4	REP	MOVSB BYTE PTR ES:[EDI], BYTE PTR DS:[ESI]	esta instruccion pasa el byte a 19ed50	
0062CEA8	5F	POP	EDI		
0062CEA9	5E	POP	ESI		
0062CEAA	5D	POP	EBP		
0062CEAB	C3	RETN			
0062CEAC	55	PUSH	EBP		
0062CEAD	8BEC	MOV	EBP, ESP		
0062CEAF	56	PUSH	ESI		
0062CEB0	57	PUSH	EDI		
0062CEB1	8B7D 08	MOV	EDI, DWORD PTR SS:[EBP+8]		
Address	Hex dump	ASCII	0019E690	00000000	
0019E6D0	05 31 34 00 00 9B 81 B8 ED 19 00 40 A8 9F 76	19ed50 yv	0019E684	0019EC30	
0019E6D0	70 5D 9C 44 FE FF FF FF C8 ED 19 00 70 43 9D 76	p1ed0pyyE1.pCv	0019E688	0019E684	

Al salir de ese bucle tenemos el md5 para nuestro nombre en 19ed50.

TestDbg - unpacked.Exe [CPU - main thread, module unpacked]

File View Debug Plugins Options Window Help

Registers (MMX)

EAX 00000000  
ECX 0019E6E8  
EDX 00000002  
EBX 036C5B18  
ESP 0019EC3C ASCII "12121212"  
EBP 0019EE7C  
ESI 005C642C unpacked.005C642C  
EDI 0019F23C  
EIP 00472619 unpacked.00472619  
C 0 ES 002B 32bit 0(FFFFFFFF)  
P 1 CS 0023 32bit 0(FFFFFFFF)  
A 0 SS 002B 32bit 0(FFFFFFFF)  
Z 1 DS 002B 32bit 0(FFFFFFFF)  
S 0 FS 0053 32bit 7FFDD000  
T 0 GS 002B 32bit 0(FFFFFFFF)  
D 0  
O 0 LastErr ERROR\_SUCCESS  
EFL 00000246 (NO,NB,E,BE,N)

MM0 0000 0000 0000 0000  
MM1 0000 0000 0000 0000  
MM2 0000 0000 0000 0000  
MM3 0000 0000 0000 0000  
MM4 FAD0 0000 0000 0000  
MM5 B900 0000 0000 0000  
MM6 DE00 0000 0000 0000

Address Hex dump ASCII

0019ED90 00 0F 36 00 00 00 00 00 06 00 00 00 01 00 00 00 :06.....  
0019ED90 CE 05 01 00 00 80 7F 01 00 00 00 60 7C 21 32 :00..y t...12  
0019ED90 3C F2 19 00 00 CC C7 77 C8 1A 6C 03 FE FF FF FF :<0t.AIQwE1pyyy  
0019ED90 00 EE 19 00 00 00 00 00 7E 05 01 00 00 00 00 00 :.t.....8.....  
0019ED90 8C ED 19 00 13 CA 90 76 18 EE 19 00 40 A8 9F 76 :BtC:Evxt:0 Yv

Command [00472619]

Buscar en la web y en Windows

Finalmente vemos como es el proceso de registro en este caso lo salto jejeje pero si estuviese todo correcto lo guarda ahí.

00472624 68 01000080 PUSH 80000080  
00472629 E8 5EB3C100 CALL <JMP.&ADVAPI32.RegOpenKeyA>  
0047262E 85C0 TEST EAX, EAX  
00472630 74 18 JE SHORT unpacked.00472640  
00472632 8D4D DC LEA EAX, DWORD PTR SS:[EBP-24]  
00472635 51 PUSH ECX  
00472636 68 00C6D000 PUSH unpacked.006D0CCD  
00472638 68 01000080 PUSH 80000080  
00472640 E8 2FBC1C00 CALL <JMP.&ADVAPI32.RegCreateKeyA>  
00472645 85C0 TEST EAX, EAX  
00472647 74 04 JE SHORT unpacked.0047264D  
00472649 33C0 XOR EAX, EAX  
0047264B EB 53 JMP SHORT unpacked.004726A0  
0047264D 4F75 08 PUSH DWORD PTR SS:[EBP+8]  
00472650 E8 53AA1B00 CALL unpacked.0063D0A8

hkey = HKEY\_CURRENT\_USER  
RegOpenKeyA  
este salto dice si lo guarda ahí o no  
plantea  
hkey = "SOFTWARE\EasyBoot Systems\UltraISO\5.0"  
hkey = HKEY\_CURRENT\_USER  
RegCreateKeyA  
el nombre

EBP 0019EE7C  
ESI 005C642C unpacked.005C642C  
EDI 0019F23C  
EIP 00472635 unpacked.00472635  
C 0 ES 002B 32bit 0(FFFFFFFF)  
P 0 CS 0023 32bit 0(FFFFFFFF)  
A 0 SS 002B 32bit 0(FFFFFFFF)  
Z 0 DS 002B 32bit 0(FFFFFFFF)  
S 0 FS 0053 32bit 7FFDD000  
T 0 GS 002B 32bit 0(FFFFFFFF)  
D 0  
O 0 LastErr ERROR\_SUCCESS  
EFL 00000217 (NO,NB,E,BE,N)

Y ya, al parecer es todo lo que este programa hace.

Podemos tomar este md5 y escribirlo en el registro a mano con los parámetros que más abajo podemos ver, uno para el nombre y otro para el registro.

00472658 6A 01 PUSH 1  
0047265D 6A 00 PUSH 0  
0047265F 68 E70C6D00 PUSH unpacked.006D0CE7  
00472664 FF75 DC PUSH DWORD PTR SS:[EBP-24]  
00472667 E8 38B3C100 CALL <JMP.&ADVAPI32.RegSetValueExA>  
0047266C 8D93 D4FEFF LEA EDX, DWORD PTR SS:[EBP-12C]  
00472672 52 PUSH EDX  
00472673 E8 30AA1B00 CALL unpacked.0062D0A8  
00472678 59 POP ECX  
00472679 40 INC EAX  
0047267A 50 PUSH EAX  
0047267B 8D8D D4FEFF LEA ECX, DWORD PTR SS:[EBP-12C]  
00472681 51 PUSH ECX  
00472682 6A 01 PUSH 1  
00472684 6A 00 PUSH 0  
00472686 68 F00C6D00 PUSH unpacked.006D0CFD  
0047268B FF75 DC PUSH DWORD PTR SS:[EBP-24]  
0047268E E8 11B3C100 CALL <JMP.&ADVAPI32.RegSetValueExA>

hkey = HKEY\_CURRENT\_USER  
RegOpenKeyA  
este salto dice si lo guarda ahí o no  
plantea  
hkey = "SOFTWARE\EasyBoot Systems\UltraISO\5.0"  
hkey = HKEY\_CURRENT\_USER  
RegCreateKeyA  
el nombre

EBP 0019EE7C  
ESI 005C642C unpacked.005C642C  
EDI 0019F23C  
EIP 00472635 unpacked.00472635  
C 0 ES 002B 32bit 0(FFFFFFFF)  
P 0 CS 0023 32bit 0(FFFFFFFF)  
A 0 SS 002B 32bit 0(FFFFFFFF)  
Z 0 DS 002B 32bit 0(FFFFFFFF)  
S 0 FS 0053 32bit 7FFDD000  
T 0 GS 002B 32bit 0(FFFFFFFF)  
D 0  
O 0 LastErr ERROR\_SUCCESS  
EFL 00000217 (NO,NB,E,BE,N)

Vamos a hacerlo con un pequeño código el cual es:

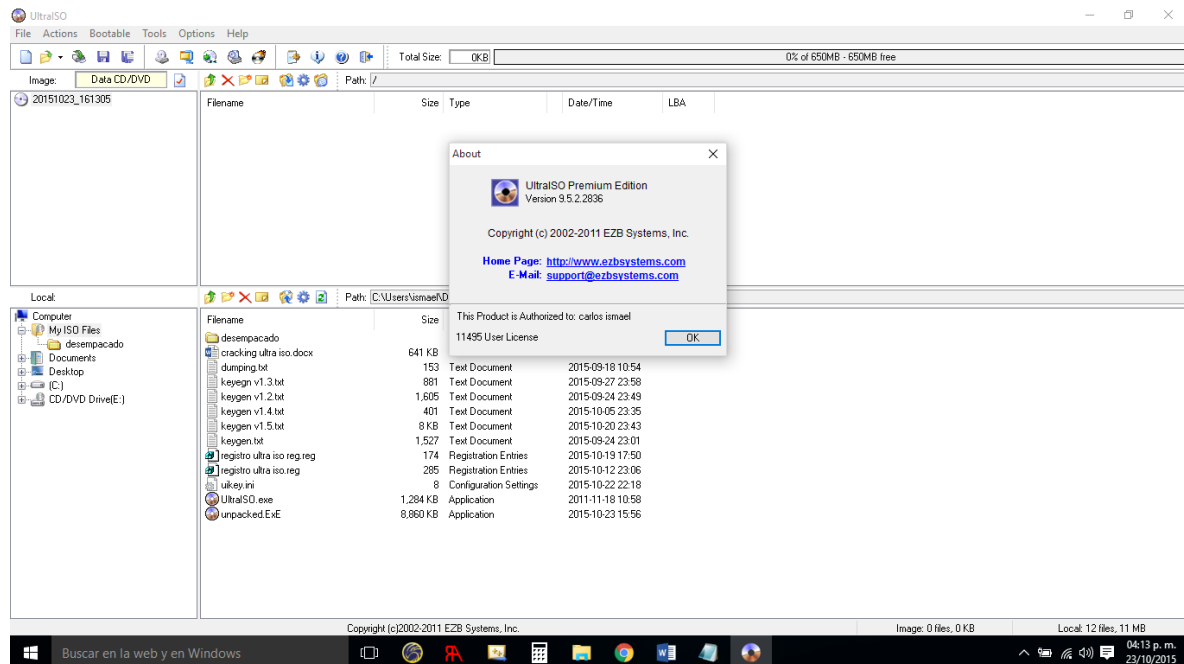
REGEDIT4

[HKEY\_CURRENT\_USER\SOFTWARE\EasyBoot Systems\UltraISO\5.0]

"UserName"="carlos ismael"

"Registration"="c1a4fcb6ac918adad1a8efbfaf81c8c2"

En el bloc de notas lo guardamos con la extensión .reg lo ejecutamos y ya



Listo es todo por ahora, lo del keygen lo dejo para más adelante ya que ando algo corto de tiempo pero tan pronto lo tenga lo subo a las web, sale???

Por lo pronto diré que es un programa muy bien hecho muchas rutinas como para perderse y nunca salir de nuevo, me imagino que ese es el plan de los programadores, de esto me agarro para decir que si hay algún error ya sea en la escritura o en los códigos de verdad mis disculpas, como humanos que somos tenemos derecho a cometer errores, me salte muchas partes pero así debe ser si no el tutorial no tendría fin, pues en fin cualquier queja comentario o sugerencia no lo duden, más abajo les dejo como contactar y gracias por leer.

Bueno pues por mi parte es todo, un saludo a toda la lista de crackslatinos que sin ellos no hubiera sido posible para mi llegar hasta aquí, gracias por leer el tutorial y practicarlo, y un saludo muy especial a los grandes maestros: Ricardo narvaja, invision, indulgeo, nox y muchos más que gracias a ellos soy capaz de hacer este tipo de cosas, aunque no los conozco en persona sé que están ahí y nos siguen apoyando como grandes personas que son.