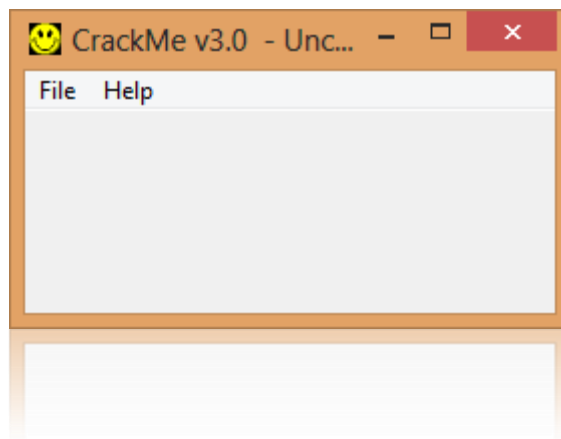


# Keygen para el CrackMe#03 de Cruehead

Archivo llave



By deurus  
08/09/2014

# ÍNDICE

1. Introducción .....	2
2. El algoritmo .....	2
3. Recapitulando .....	4
4. Enlaces.....	6
5. Crackeando Crackmes by deurus .....	6

*Equipo utilizado:*

*S.O: Windows 7 x32 /Windows 8 x64*

*Depurador: Ollydbg 1.10 (32bits) con plugins*

*Analizador: PEiD 0.95*

## 1. Introducción

Esta es la tercera y última entrega de los crackmes de **Cruehead**. En esta ocasión nos enfrentamos a un “**keyfile**”, un **archivo llave** para que nos entendamos. Tiene un poco más de dificultad que los anteriores pero es ideal para los que empiezan.

## 2. El algoritmo

Si iniciamos el crackme no pasa nada, lo único que vemos es la palabra “**UNCRACKED**” en el título. Abrimos el crackme con Olly y empezamos. En las “**string references**” vemos el nombre del archivo llave “**crackme3.key**”. Lo creamos y escribimos el serial 12345678 y empezamos a tracear.

El **CMP EAX,-1** significa que está comprobando que el archivo no esté vacío, como no es nuestro caso continuamos.

00401000	6A 00	PUSH 0	Module Name = NULL
00401002	E8 7D040000	CALL <JMP.&KERNEL32.GetModuleHandleA>	Kernel32.GetModuleHandleA
00401007	A3 F3204000	MOV DWORD PTR DS:[4020E9],EAX	
0040100C	C705 F3204000	MOV DWORD PTR DS:[4020F9],0	
00401016	6A 00	PUSH 0	
00401018	68 80000000	PUSH 80	hTemplate = NULL
0040101D	6A 03	PUSH 3	Attributes = FILE_ATTRIBUTE_NORMAL
0040101F	6A 00	PUSH 0	CreationDisposition = OPEN_EXISTING
00401021	6A 03	PUSH 3	pSecurity = NULL
00401023	68 000000C0	PUSH C0000000	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
00401028	68 D7204000	PUSH OFFSET 004020D7	DesiredAccess = GENERIC_READ GENERIC_WRITE
0040102D	E8 76040000	CALL <JMP.&KERNEL32.CreateFileA>	CreateFileA
00401032	83F8 FF	CMP EAX,-1	Kernel32.CreateFileA
00401035	75 0C	JNE SHORT 00401043	CONST FFFFFFFF => INVALID_HANDLE_VALUE
00401037	68 0E214000	PUSH OFFSET 0040210E	
0040103C	E8 B4020000	CALL 004012F5	ASCII "CrackMe v3.0 - Uncracked"
00401041	EB 6B	JMP SHORT 004010AE	

A continuación vemos que compara nuestra longitud de serial con **0x12 (18 en decimal)**. Nuestro serial tiene 8 dígitos así que nos tira fuera.

00401032	83F8 FF	CMP EAX,-1	CONST FFFFFFFF => INVALID_HANDLE_VALUE
00401035	75 0C	JNE SHORT 00401043	ASCII "CrackMe v3.0"
00401037	68 0E214000	PUSH OFFSET 0040210E	
0040103C	E8 B4020000	CALL 004012F5	
00401041	EB 6B	JMP SHORT 004010AE	
00401043	A3 F5204000	MOV DWORD PTR DS:[4020F5],EAX	
00401048	B8 12000000	MOV EAX,12	
0040104D	BB 08204000	MOV EBX,OFFSET 00402008	
00401052	6A 00	PUSH 0	
00401054	68 00214000	PUSH OFFSET 004021A0	
00401059	50	PUSH EAX	
0040105A	53	PUSH EBX	
0040105B	FF35 F5204000	PUSH DWORD PTR DS:[4020F5]	
00401061	E8 30040000	CALL <JMP.&KERNEL32.ReadFile>	
00401066	83D0 A0214000	CMP DWORD PTR DS:[4021A0],12	
0040106D	75 C8	JNE SHORT 004010B7	
0040106F	68 08204000	PUSH OFFSET 00402008	
00401074	E8 98020000	CALL 00401311	
00401079	8135 F3204000	XOR DWORD PTR DS:[4020F9],12345678	
00401083	83C4 04	ADD ESP,4	
00401086	68 08204000	PUSH OFFSET 00402008	
0040108B	E8 AC020000	CALL 0040133C	

Escribimos en el archivo llave el serial “**deurus123456789012**” y volvemos a tracear. Vemos que ahora si pasa los filtros iniciales y llegamos a la primera zona interesante. En la imagen está explicado pero os hago un resumen. En el bucle lo que hace es un XOR a los primeros 14 dígitos de nuestro serial con los valores del 41 al 4E (4F finaliza). **El bucle solo se rompe si llegamos a 4F o si el resultado del XOR da 0.** Además en **EAX** **acumula la suma** del resultado del XOR.

<pre> 00401311   . 33C9      XOR ECX,ECX 00401313   . 33C0      XOR EAX,EAX 00401315   . 8B7424 04  MOV ESI,DWORD PTR SS:[ARG.1] 00401319   . B3 41     MOV BL,41 0040131B   &gt; 8A06      MOV AL,BYTE PTR DS:[ESI] 0040131D   . 32C3      XOR AL,BL 0040131F   . 8B06      MOV BYTE PTR DS:[ESI],AL 00401321   . 46        INC ESI 00401322   . FEC3      INC BL 00401324   . 0105 F9204000 ADD DWORD PTR DS:[4020F9],EAX 0040132A   . 3C 00     CMP AL,0 0040132C   ^ 74 07     JE SHORT 00401335 0040132E   . FEC1      INC CL 00401330   . 80FB 4F   CMP BL,4F 00401333   ^ 75 E6     JNE SHORT 0040131B 00401335   &gt; 890D 49214000 MOV DWORD PTR DS:[402149],ECX 0040133B   . C3        RETN </pre>	<pre> CRACKME3_cruehead.00401311(guessed Arg1) ; Coje nuestro digito ; 41 XOR digito ; 42 XOR digito... ; Incrementa BL (42,43,44,45,46,47,48...) ; EAX va sumando el resultado del XOR ; Si el resultado del XOR da 0 salimos del bucle ; Si BL llega a 4F fin del bucle </pre>
--	--

Ejemplo:

1	d	e	u	r	u	s	1	2	3	4	5	6	7	8	9	0	1	2
2	64	65	75	72	75	73	31	32	33	34	35	36	37	38				
3	XOR																	
4	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E				
5	-----																	
6	25	27	36	36	30	35	76	7A	7A	7E	7E	7A	7A	76	= 4ED (Suma)			

A continuación hace **XOR** entre **12345678** y **4ED**, coje los 4 últimos dígitos de nuestro serial y los compara.

<pre> 00401079   . 8135 F9204000 XOR DWORD PTR DS:[4020F9],12345678 00401083   . 83C4 04    ADD ESP,4 00401086   . 68 08204000 PUSH OFFSET 00402008 0040108B   . E8 AC020000 CALL 0040133C 00401090   . 83C4 04    ADD ESP,4 00401093   . 3B05 F9204000 CMP EAX,DWORD PTR DS:[4020F9] 00401099   . 0F94C0    SETE AL 0040109C   . 50        PUSH EAX 0040109D   . 84C0      TEST AL,AL 0040109F   ^ 74 96     JZ SHORT 004010B7 004010A1   . 68 0E214000 PUSH OFFSET 0040210E 004010A6   . E8 9B020000 CALL 00401346 </pre>	<pre> ; 12345678 XOR SUMNombre Arg1 = CRACKME3_cruehead.402008 CRACKME3_cruehead.0040133C ; Coje los 4 ultimos digitos del serial ; Compara el resultado del XOR con los digitos 15,16,17,18 ; Si no coincide "UNCRACKED" ASCII "CrackMe v3.0" </pre>
---	---

Ejemplo:

```

1 12345678 XOR 4ED = 12345295
2 Compara 12345295 con 32313039
3 32313039 = 2109, nuestros 4 últimos dígitos al revés. Recordemos que nuestro serial era "deurus123456789012"

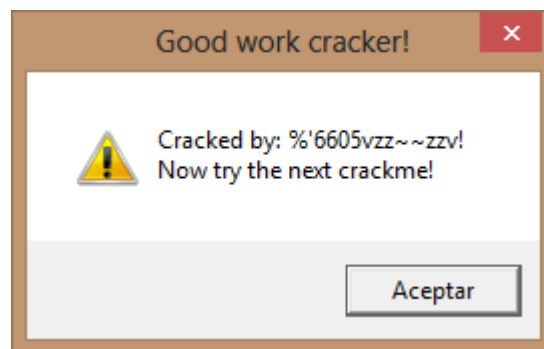
```

El serial bueno para el nombre **deurus12345678** serían los bytes correspondientes de “12345295”, es decir, nuestro serial bueno sería:

Ejemplo:

```
1 Necesitamos 12345295 para la comparación.
2 12 34 52 95 hexadecimal
3 18 52 82 149 decimal
4 Tenemos que escribirlo al revés. Con Alt-Izq + 149 escribimos el primer caracter, el resto igual.
5 Nuestro serial quedaría: deurus12345678òR4;
```

Metemos el serial y vemos que lo acepta pero que nos muestra un nombre extraño. Esto es porque nos está mostrando **los bytes del nombre xoreados**, tendremos que hacer un XOR antes al nombre que queramos para que lo muestre correctamente.



### 3. Recapitulando

Con lo que sabemos ahora hay que empezar a coger el toro por los cuernos. Lo primero que queremos que muestre el nombre deurus y no deurus12345678. Para ello **debemos cortar el bucle** y eso solo lo podemos hacer forzando que el resultado del XOR sea 0. Ok pues para **deurus** el siguiente valor de BL, es decir el séptimo, en el bucle sería 47 lo que corresponde a la letra G. Pues si ponemos de serial **deurusGxxxxxxxxxx** ya tendríamos la primera parte solucionada.

Pero recordemos que necesitamos XORear el nombre inicialmente, luego debemos escribir el resultado del XOR.

Ejemplo:

```

1 d e u r u s
2 64 65 75 72 75 73
3
4 41 42 43 44 45 46
5 -----
6 25 27 36 36 30 35
7
8 25 27 36 36 30 35 ----- debemos meter esto en el archivo llave.
9
10 41 42 43 44 45 46
11 -----
12 64 65 75 72 75 73 ----- Al desenscriptarlo el bucle se verá nuestro nombre correctamente.
13
14 En el archivo llave escribiremos: %'6605

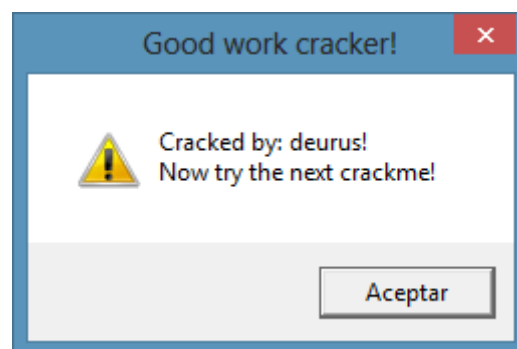
```

Ahora nos faltaría **calcular el nuevo SUM**. Como el resultado del XOR ahora es nuestro nombre, basta con sumar sus valores ascii (64+65+75+72+75+73 == 0x298)

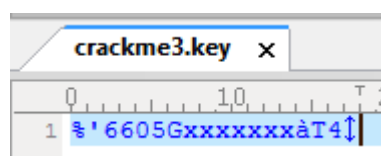
$0x12345678 \text{ XOR } 0x298 == 0x123454E0$

Luego nuestros 4 últimos dígitos deben ser lo correspondiente a los bytes E0, 54, 34, 12. Los pasamos a decimal y los escribimos en el archivo llave con el truco del ALT-Izq que hemos comentado antes.

El contenido final del archivo llave para el nombre **deurus** sería: **%'6605GxxxxxxxxÓT4↕**



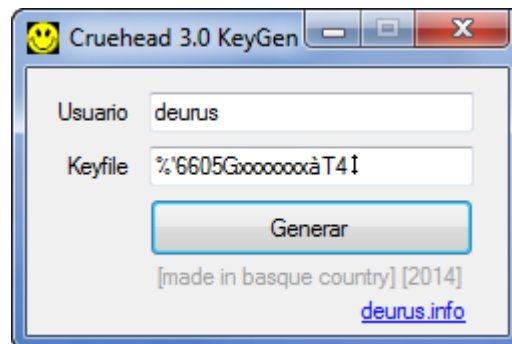
Aquí vemos el contenido del archivo llave normal.



Y aquí lo vemos con un editor hexadecimal. Como veis se ven claramente los bytes E0, 54, 34, 12.

```
crackme3.key x
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: 25 27 36 36 30 35 47 78 78 78 78 78 78 78 E0 54 ; %'6605GxxxxxxxxàT
00000010h: 34 12 ; 4.
```

Os dejo un keygen hecho en .Net para que probéis. Os genera el contenido del archivo y el archivo “crackme3.key”.



## 4. Enlaces

- Crackme
- Keygen
- Cruehead's Crackme 1.0 Keygen [1/3]
- Cruehead's Crackme 2.0 Serial [2/3]

## 5. Crackeando Crackmes by deurus

- <https://mega.co.nz/#F!88BRwYoT!O0TzTSZYCdczKLOfrOyGw>
- [Lolabits.es/blogcracking](http://Lolabits.es/blogcracking) (Clave: **blogcrackhack**)