

Keygen para el Crackme#8 de Kwazy Webbit

Comparación no lineal

By deurus
03/12/2013

ÍNDICE

1.	Primeras impresiones	2
2.	Al ataque con Ollydbg	2
4.	Sacando el "HashName"	5
5.	Entendiendo la comprobación del serial	5
6.	Generando el serial válido	6
7.	Keygen en ensamblador	7
8.	Enlaces	7

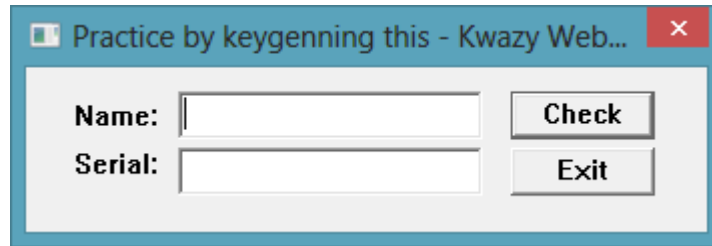
Equipo utilizado:

S.O: Windows 8 x64

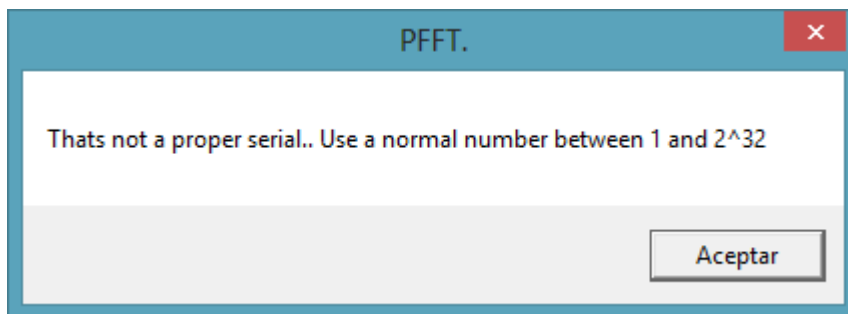
Depurador: Ollydbg 2.01 (32bits) sin plugins

Analizador: PEiD 0.95

1. Primeras impresiones



El crackme es el típico de usuario y número de serie. Si no introduces un nombre te salta un messagebox indicándotelo y si introduces cualquier información sale un mensaje de error.

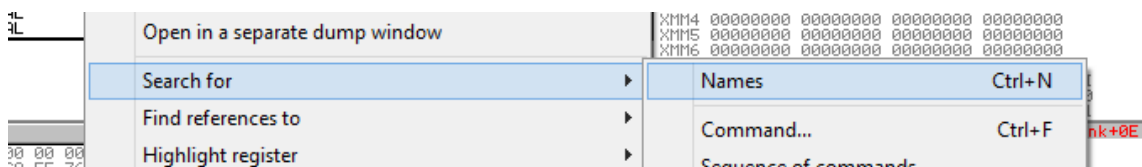


Si dejamos solamente el serial en blanco nos sale un mensaje de error muy interesante diciéndonos que introduzcamos un número entre 1 y 2^{32} . Por lo tanto ya sabemos que nuestro serial está entre 1 y 4294967296.

PEiD no arroja resultados pero una primera impresión con Ollydbg hace creer que está programado en ensamblador y que no está comprimido.

2. Al ataque con Ollydbg

Cargamos el crackme en Ollydbg y hacemos click derecho Search for > Names



Vemos dos referencias interesantes como son:

- &USER32.GetDlgItemInt
- &USER32.GetDlgItemTextA

Ponemos sendos breakpoints y damos al play.

Names in Crackme8				
Address	Section	Type	Ordinal	Name
00400000		Analysér		<STRUCT IMAGE_DOS_HEADER>
00400098		Analysér		<STRUCT IMAGE_NT_SIGNATURE>
0040009C		Analysér		<STRUCT IMAGE_FILE_HEADER>
004000B0		Analysér		<STRUCT IMAGE_OPTIONAL_HEADER>
00400110		Analysér		<STRUCT IMAGE_DATA_DIRECTORY>
00400190		Analysér		<STRUCT IMAGE_SECTION_HEADER>
004001B8		Analysér		<STRUCT IMAGE_SECTION_HEADER>
004001E0		Analysér		<STRUCT IMAGE_SECTION_HEADER>
00401182	.text	Export		<ModuleEntryPoint>
00402000	.rdata	Import		&KERNEL32.ExitProcess
00402004	.rdata	Import		&KERNEL32.GetModuleHandleA
00402008	.rdata	Import		&KERNEL32.lstrlenA
00402010	.rdata	Import		&USER32.MessageBoxA
00402014	.rdata	Import		&USER32.GetDlgItemInt
00402018	.rdata	Import		&USER32.GetDlgItemTextA
0040201C	.rdata	Import		&USER32.EndDialog
00402020	.rdata	Import		&USER32.DialogBoxParamA

Vemos que para en **USER32.GetDlgItemTextA** y que retorna al offset **4010E7**

```

----- Rnd NEAR Mask 1 1 1 1 1 1
RETURN from USER32.GetDlgItemTextA to Crackme8.004010E7
hDialog = 009B0732, class = #32770, text = Practice by k
ItemID = 1000.
String = "*"
MaxCount = 50.

```

Vamos a **4010E7** y vemos que pasa.

```

004010D5 | 6A 32          PUSH 32
004010D7 | 8D45 CC        LEA EAX,[LOCAL.13]
004010DA | 50            PUSH EAX
004010DB | 68 E8030000    PUSH 3E8
004010DE | 57            PUSH EDI
004010E1 | FF15 18204000 CALL DWORD PTR DS:[<&USER32.GetDlgItemT
004010E7 | 85C0          TEST EAX,EAX
004010E9 | 75 00          JNZ SHORT 004010F8
004010EB | 50            PUSH EAX
004010EC | 68 A4234000    PUSH OFFSET 004023A4
004010F1 | 68 84234000    PUSH OFFSET 00402384
004010F6 | EB 1F          JMP SHORT 00401117
MaxCount = 50.
String => OFFSET LOCAL.13
ItemID = 1000.
hDialog => [ARG.1]
USER32.GetDlgItemTextA
ASCII "PFFT."
ASCII "You might want to enter a name?"

```

Hace un Test eax,eax por si hemos introducido algún nombre y si no es así nos muestra la nag.

Continuamos con la ejecución y para en el siguiente breakpoint, esta vez el referente a **USER32.GetDlgItemInt**, vamos al offset 401108 a ver que nos espera.

```

----- INTR INTR INTR INTR INTR INTR
RETURN from USER32.GetDlgItemInt to Crackme8.00401108

```

Se puede ver claramente que carga en **EAX** nuestro número de serie en hexa, lo compara con **ESI** que vale 0 y si son iguales nag de error y si no continua a 401120 donde guarda en la pila nuestro nombre y serial y llama al offset **401000**.

```

00401102 | FF15 14204000 CALL DWORD PTR DS:[<&USER32.GetDlgItemInt
00401108 | 3BC6          CMP EAX,ESI
0040110A | 75 14          JNE SHORT 00401120
0040110C | 56            PUSH ESI
0040110D | 68 A4234000    PUSH OFFSET 004023A4
00401112 | 68 40234000    PUSH OFFSET 00402340
00401117 | 57            PUSH EDI
00401118 | FF15 10204000 CALL DWORD PTR DS:[<&USER32.MessageBoxA
0040111E | EB A9          JMP SHORT 004010C9
00401120 | 50            PUSH EAX
00401121 | 8D45 CC        LEA EAX,[LOCAL.13]
00401124 | 50            PUSH EAX
00401125 | E8 D6FEFFFF    CALL 00401000
0040112A | 85C0          TEST EAX,EAX
0040112C | 59            POP ECX
0040112D | 59            POP ECX
0040112E | 56            PUSH ESI
0040112F | 74 0C          JZ SHORT 00401130
00401131 | 68 38234000    PUSH OFFSET 00402338
00401136 | 68 2C234000    PUSH OFFSET 0040232C
00401138 | EB DA          JMP SHORT 00401117
0040113D | A1 B4234000    MOV EAX,DWORD PTR DS:[4023B4]
00401142 | 68 24234000    PUSH OFFSET 00402324
00401147 | 6A 0B          PUSH 0B
00401149 | 59            POP ECX
0040114A | 99            CDQ
0040114B | F7F9          IDIV ECX
0040114D | FF3495 282040 PUSH DWORD PTR DS:[EDX*4+402028]
00401154 | 57            PUSH EDI
USER32.GetDlgItemInt
ASCII "PFFT."
ASCII "That's not a proper serial.. Use :
Arg2
Arg1 => OFFSET LOCAL.13
Crackme8.00401000
ASCII "RIGHT"
ASCII "You got it!"
ASCII "WRONG"
PTR to ASCII "Now now, that wasn't even

```

Veamos que hay en el offset **401000**.

00401000	55	PUSH EBP	Crackme8.00401000(gues
00401001	8BEC	MOV EBP,ESP	
00401003	51	PUSH ECX	
00401004	51	PUSH ECX	
00401005	8365 FC 00	AND DWORD PTR SS:[LOCAL.1],00000000	
00401009	56	PUSH ESI	
0040100A	8B75 08	MOV ESI,DWORD PTR SS:[ARG.1]	
0040100D	57	PUSH EDI	
0040100E	56	PUSH ESI	
0040100F	FF15 08204000	CALL DWORD PTR DS:[&KERNEL32.lstrlenA]	[String => [ARG.1] KERNEL32.lstrlen
00401015	8BF8	MOV EDI,EAX	
00401017	33D2	XOR EDX,EDX	
00401019	85FF	TEST EDI,EDI	
0040101B	7E 2A	JL SHORT 00401047	
0040101D	0FBEC32	MOVSX ECX,BYTE PTR DS:[ESI+EDX]	
00401021	014D FC	ADD DWORD PTR SS:[LOCAL.1],ECX	
00401024	894D F8	MOV DWORD PTR SS:[LOCAL.2],ECX	
00401027	D145 FC	ROL DWORD PTR SS:[LOCAL.1],1	
0040102A	8BC1	MOV EAX,ECX	
0040102C	0FAF45 FC	IMUL EAX,DWORD PTR SS:[LOCAL.1]	
00401030	8945 FC	MOV DWORD PTR SS:[LOCAL.1],EAX	
00401033	8B45 F8	MOV EAX,DWORD PTR SS:[LOCAL.2]	
00401036	0145 FC	ADD DWORD PTR SS:[LOCAL.1],EAX	
00401039	314D FC	XOR DWORD PTR SS:[LOCAL.1],ECX	
0040103C	42	INC EDX	
0040103D	3BD7	CMP EDX,EDI	
0040103F	7C DC	JL SHORT 0040101D	
00401041	837D FC 00	CMP DWORD PTR SS:[LOCAL.1],0	
00401045	75 1C	JNE SHORT 00401063	
00401047	6A 00	PUSH 0	
00401049	68 0C234000	PUSH OFFSET 0040230C	
0040104E	68 CC224000	PUSH OFFSET 004022CC	
00401053	FF35 AC234000	PUSH DWORD PTR DS:[4023AC]	
00401059	FF15 10204000	CALL DWORD PTR DS:[&USER32.MessageBoxA]	
0040105F	33C0	XOR EAX,EAX	
00401061	EB 1C	JMP SHORT 0040107E	
00401063	8175 0C DEC0	XOR DWORD PTR SS:[ARG.2],1337C0DE	
0040106A	816D 0C E50D	SUB DWORD PTR SS:[ARG.2],BADCODES	
00401071	8B45 FC	MOV EAX,DWORD PTR SS:[LOCAL.1]	
00401074	F755 0C	NOT DWORD PTR SS:[ARG.2]	
00401077	3345 0C	XOR EAX,DWORD PTR SS:[ARG.2]	
0040107A	F7D8	NEG EAX	
0040107C	1BC0	SBB EAX,EAX	
0040107E	40	INC EAX	
0040107F	5F	POP EDI	
00401080	5E	POP ESI	
00401081	C9	LEAVE	
00401082	C3	RETN	

BUCLE CON NOMBRE (OBTENEMOS HASHNAME)

ASCII "A problem has o
ASCII "There is no val

OPERACIONES CON EL HASHNAME
Converts EAX to boolea

Aquí vemos una primera parte con un bucle en el que interviene nuestro nombre y donde obtendremos el "HashName" y posteriormente una operaciones aritméticas en las que finalmente modifica el valor de EAX. Tengamos en cuenta que la comprobación final es un **Test eax,eax** o lo que es lo mismo, comprueba si EAX = 0 y si es 0 salta al mensaje de error como vemos en la imagen siguiente.

00401125	E8 D6FEFFFF	CALL 00401000	Crackme8.00401000
0040112A	85C0	TEST EAX,EAX	
0040112C	59	POP ECX	
0040112D	59	POP ECX	
0040112E	56	PUSH ESI	
0040112F	74 0C	JZ SHORT 0040113D	
00401131	68 38234000	PUSH OFFSET 00402338	
00401136	68 2C234000	PUSH OFFSET 0040232C	
0040113B	EB DA	JMP SHORT 00401117	
0040113D	A1 B4234000	MOV EAX,DWORD PTR DS:[4023B4]	
00401142	68 24234000	PUSH OFFSET 00402324	

ASCII "RIGHT"
ASCII "You got it!"

ASCII "WRONG"

En resumen:

1. Obtenemos el HashName.
2. Realizamos unas operaciones a ese HashName (LOCAL.1) y al serial introducido (ARG.2).
3. Si EAX <> 0 entonces serial correcto.

4. Sacando el "HashName"

Veamos un ejemplo de obtención del hashname para el usuario "abc". El bucle se repetirá tantas veces como letras tenga el nombre.

Partimos de:

Nombre: a b c
Valores HEX: 61 62 63

```
[ADD DWORD PTR SS:[LOCAL.1],ECX
|MOV DWORD PTR SS:[LOCAL.2],ECX
|ROL DWORD PTR SS:[LOCAL.1],1
|MOV EAX,ECX
|IMUL EAX,DWORD PTR SS:[LOCAL.1]
|MOV DWORD PTR SS:[LOCAL.1],EAX
|MOV EAX,DWORD PTR SS:[LOCAL.2]
|ADD DWORD PTR SS:[LOCAL.1],EAX
|XOR DWORD PTR SS:[LOCAL.1],ECX
```

BUCLE 1

Hash + 61 = 61
Hash * 2 = C2
Hash * 61 = 4982
Hash + 61 = 49E3
Hash XOR 61 = 4982

HASH = 4982

BUCLE 2

Hash + 62 = 49E4
Hash * 2 = 93C8
Hash * 62 = 389290
Hash + 62 = 3892F2
Hash XOR 62 = 389290

HASH = 389290

BUCLE 3

Hash + 63 = 3892F3
Hash * 2 = 7125E6
Hash * 63 = 2BC1A7F2
Hash += 63 = 2BC1A855
Hash XOR 63 = 2BC1A836

HASH = 2BC1A836

5. Entendiendo la comprobación del serial

COMANDOS	COMENTARIOS
XOR [ARG.2],1337C0DE-----	Serial xor 1337C0DE
SUB [ARG.2],BADCODE5-----	Serial - BADCODE5
MOV EAX,[LOCAL.1]-----	EAX = HashName
NOT [ARG.2]-----	NOT(Serial) (Negacion bit a bit)
XOR EAX,[ARG.2]-----	HashName xor Serial
NEG EAX-----	(0 - EAX) Neg = Not + 1
SBB EAX,EAX-----	EAX = EAX - EAX
INC EAX-----	EAX +=1
<hr/>	
TEST EAX,EAX-----	Si EAX = 0 activa el flag ZF
JZ SHORT 0040113D-----	JZ = JE, salta si EAX = 0 (flag ZF = 1)

En resumen:

1. Necesitamos que $EAX \neq 0$.
2. Necesitamos que $(HashName XOR Serial) = 0$ ya que:
 - a. La negación de 0 es 0 → $NEG(0) = 0$
 - b. La resta con acarreo de $0 - 0 = 0$ → $SBB 0,0 = 0$

Hay que tener en cuenta que la resta con acarreo (SBB) de cualquier número, dará como resultado en EAX = FFFFFFFF, que al incrementar en 1 quedará en 0.

Por lo tanto si cumplimos las condiciones anteriormente expuestas, al incrementar EAX con INC EAX, este quedará en 1 haciendo nuestro serial válido.

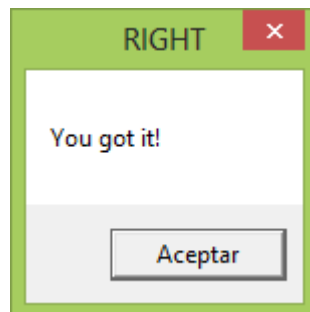
6. Generando el serial válido

Las operaciones que se realizan sobre nuestro serial son NOT, SUB y XOR. Por suerte para nosotros son reversibles quedando nuestro serial así:

Serial válido = [NOT(HashName) + 0xBADC0DE5] XOR 0x1337C0DE

Para el nombre “abc” sería:

[(NOT(734111798) + 3134983653)] XOR 322420958 = 2620237168



7. Keygen en ensamblador

Como no es propósito de este manual enseñar a hacer un keygen desde 0, muestro el código importante y adjunto los links del código fuente.

```

XOR EBX, EBX
XOR ECX, ECX
XOR EDX, EDX
XOR EDI, EDI
XOR ESI, ESI
invoke RtlZeroMemory, addr temp, sizeof temp
invoke RtlZeroMemory, addr temp2, sizeof temp2
invoke RtlZeroMemory, addr szName, sizeof szName
invoke RtlZeroMemory, addr szSerial, sizeof szSerial
;
invoke GetDlgItemText, hWin, 1001, addr szName, sizeof szName
;
CMP EAX, 0h
JB @MinSize
CMP EAX, 020h
JA @MaxSize
MOV EDI, EAX
XOR EAX, EAX
XOR ECX, ECX
XOR EDX, EDX
MOV dword ptr ds:[temp], 0
MOV dword ptr ds:[temp2], 0
;-----
;Saco el HashName
@bucle:
MOVZX ecx, byte ptr ds:[edx+szName]
ADD dword ptr ds:[temp], ecx
MOV dword ptr ds:[temp2], ecx
ROL dword ptr ds:[temp], 1
MOV eax, ecx
IMUL eax, dword ptr ds:[temp]
MOV dword ptr ds:[temp], eax
MOV eax, dword ptr ds:[temp2]
ADD dword ptr ds:[temp], eax
XOR dword ptr ds:[temp], ecx
INC edx
CMP edx, edi
j1 @bucle
;-----
;Saco mi numero de serie valido
;Formula: x = ((NOT (HashName)+BADCODE5h) XOR 1337CODEh)
;
MOV eax, dword ptr[temp] ;Copio el hashname a eax
NOT eax ;Not al hashname
ADD eax, 0BADCODE5h ;le sumo BADCODE5
XOR eax, 01337CODEh ;XOR 1337CODE
invoke wsprintf, addr szSerial, addr szFormat, eax ;Cargo en szSerial el valor sin signo de eax
invoke SetDlgItemText, hWin, 1002, addr szSerial ;Muestro el serial en la caja de texto

```

8. Enlaces

Crackme + Keygen en ASM + WinASM studio 5.1.5 [31MB]

<https://deurus.info/archivos/manuales/>