Keygen para el Crackme#8 de Kwazy Webbit

Comparación no lineal

By deurus
03/12/2013

ÍNDICE

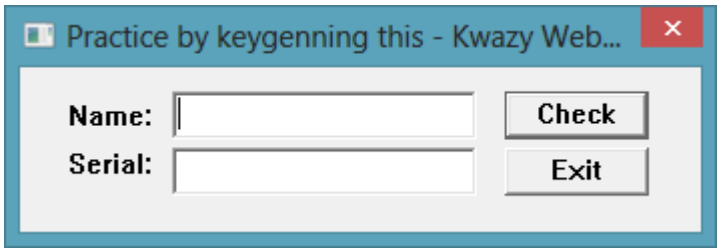
Equipo utilizado:

S.O: Windows 8 x64

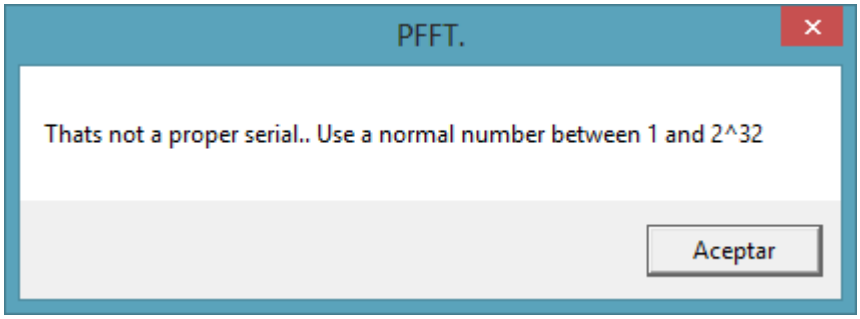
Depurador: Ollydbg 2.01 (32bits) sin plugins

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		Analyser		<STRUCT IMAGE_DOS_HEADER>
00400098		Analyser		<STRUCT IMAGE_NT_SIGNATURE>
0040009C		Analyser		<STRUCT IMAGE_FILE_HEADER>
004000B0		Analyser		<STRUCT IMAGE_OPTIONAL_HEADER>
00400110		Analyser		<STRUCT IMAGE_DATA_DIRECTORY>
00400190		Analyser		<STRUCT IMAGE_SECTION_HEADER>
004001B8		Analyser		<STRUCT IMAGE_SECTION_HEADER>
004001E0		Analyser		<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 ke
ItemID = 1000.
String = "$"
MaxCount = 50.
```

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

```
004010D5  .  6A 32          PUSH 32                          MaxCount = 50.
004010D7  .  8D45 CC        LEA EAX,[LOCAL.13]
004010DA  .  50             PUSH EAX                         String => OFFSET LOCAL.13
004010DB  .  68 E8030000    PUSH 3E8                         ItemID = 1000.
004010E0  .  57             PUSH EDI                         hDialog => [ARG.1]
004010E1  .  FF15 1820400(  CALL DWORD PTR DS:[<&USER32.GetDlgItemT  USER32.GetDlgItemTextA
004010E7  .  85C0           TEST EAX,EAX
004010E9  .v 75 0D          JNZ SHORT 004010F8
004010EB  .  50             PUSH EAX
004010EC  .  68 A4234000    PUSH OFFSET 004023A4             ASCII "PFFT."
004010F1  .  68 84234000    PUSH OFFSET 00402384             ASCII "You might want to enter a name?"
004010F6  .v EB 1F          JMP SHORT 00401117
```

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

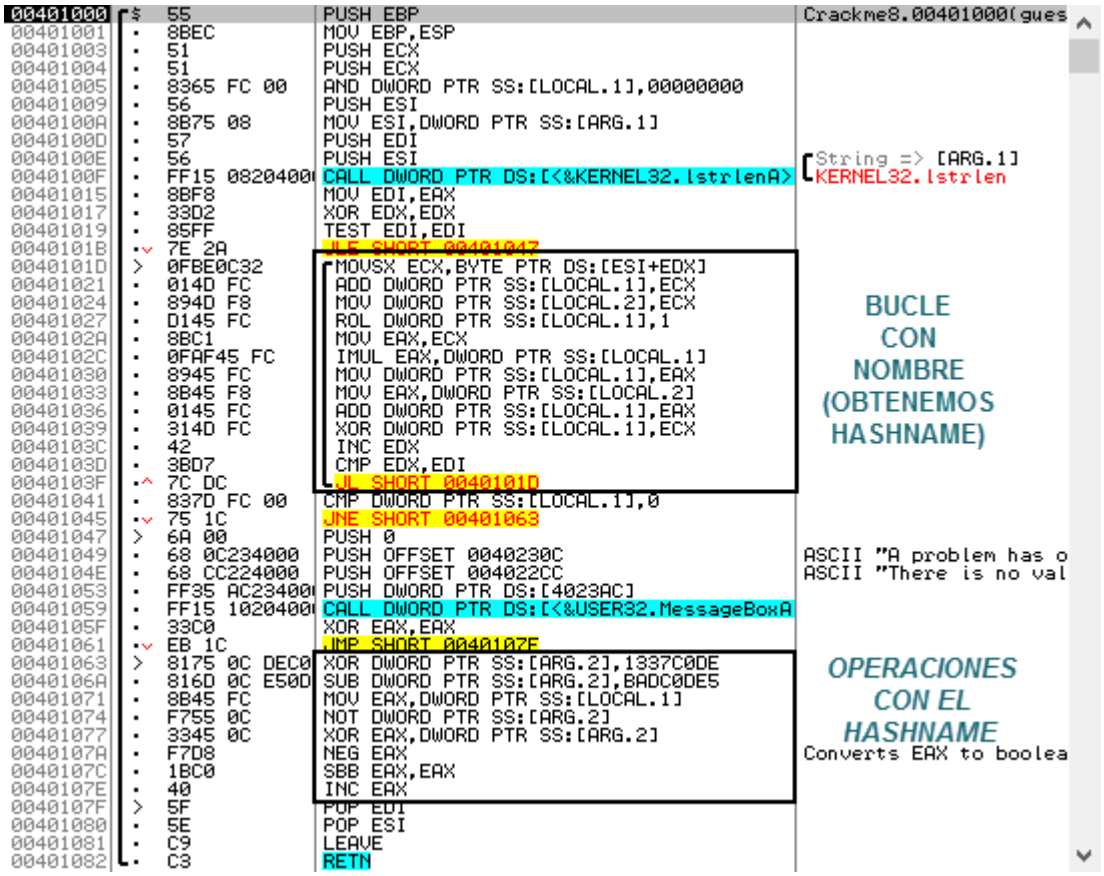
Continuamos con la ejecución y para en el siguiente breakpoint, esta vez el referente a **USER32.GetDlgItemInt**, vamos al offset 401108 a ver que nos espera.

```
RETURN from USER32.GetDlgItemInt to Crackme8.00401108
```

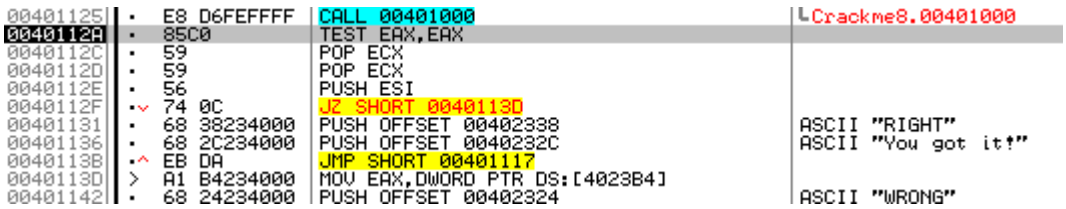
Se puede ver claramente que carga en **EAX** nuestro número de serie en hexa, lo compara con **ESI** que vale 0 y si son iguales nag de error y si no continua a 401120 donde guarda en la pila nuestro nombre y serial y llama al offset **401000.**

```
00401102  .  FF15 1420400(  CALL DWORD PTR DS:[<&USER32.GetDlgItemI  USER32.GetDlgItemInt
00401108  .  3BC6           CMP EAX,ESI
0040110A  .v 75 14          JNE SHORT 00401120
0040110C  .  56             PUSH ESI
0040110D  .  68 A4234000    PUSH OFFSET 004023A4             ASCII "PFFT."
00401112  .  68 40234000    PUSH OFFSET 00402340             ASCII "Thats not a proper serial.. Use a
00401117  >  57             PUSH EDI
00401118  .  FF15 1020400(  CALL DWORD PTR DS:[<&USER32.MessageBoxA
0040111E  .^ EB A9          JMP SHORT 004010C9
00401120  >  50             PUSH EAX                         Arg2
00401121  .  8D45 CC        LEA EAX,[LOCAL.13]
00401124  .  50             PUSH EAX                         Arg1 => OFFSET LOCAL.13
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  .v 74 0C          JZ SHORT 0040113D
00401131  .  68 38234000    PUSH OFFSET 00402338             ASCII "RIGHT"
00401136  .  68 2C234000    PUSH OFFSET 0040232C             ASCII "You got it!"
0040113B  .^ EB DA          JMP SHORT 00401117
0040113D  >  A1 B4234000    MOV EAX,DWORD PTR DS:[4023B4]
00401142  .  68 24234000    PUSH OFFSET 00402324             ASCII "WRONG"
00401147  .  6A 0B          PUSH 0B
00401149  .  59             POP ECX
0040114A  .  99             CDQ
0040114B  .  F7F9           IDIV ECX
0040114D  .  FF3495 28204(  PUSH DWORD PTR DS:[EDX*4+402028] PTR to ASCII "Now now, that wasn't even
00401154  .  57             PUSH EDI
```

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



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.

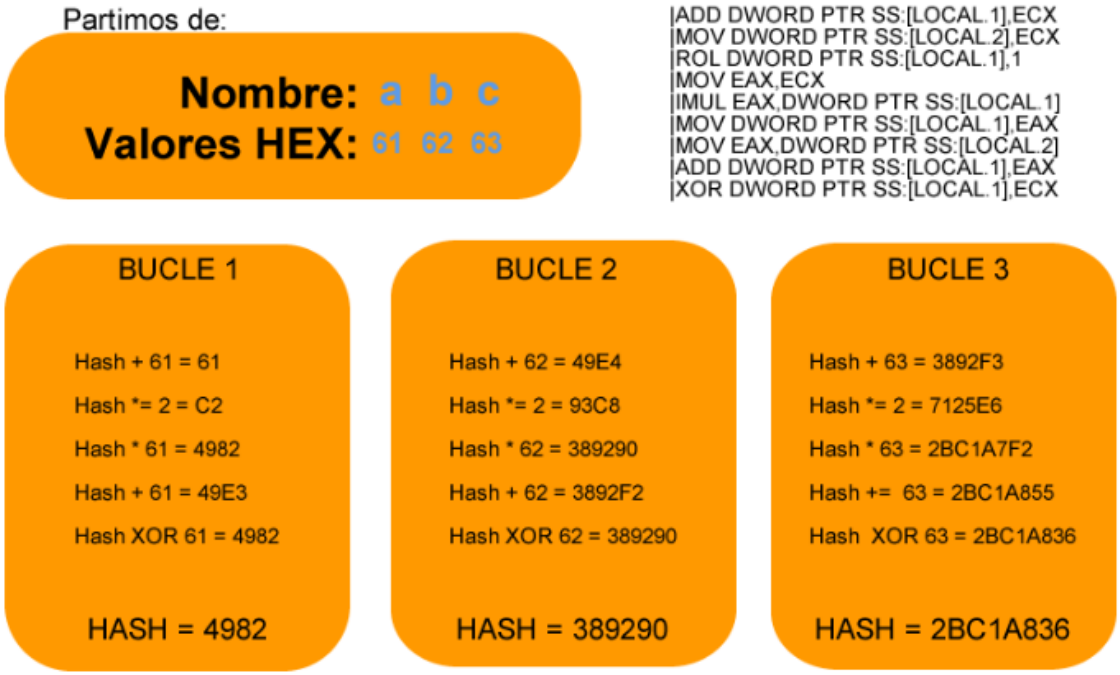


En resumen:

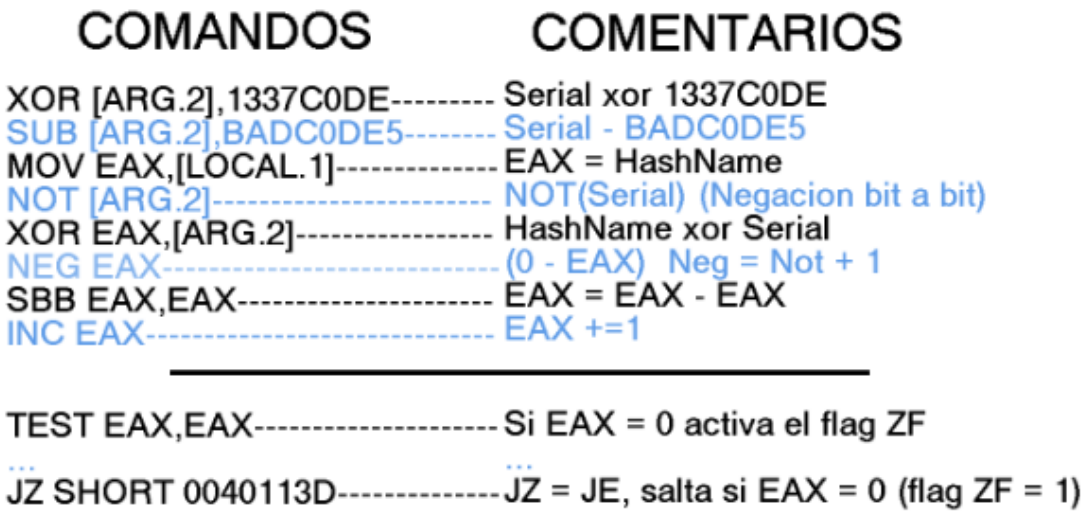
1. **Obtenemos el HashName.**
2. **Realizamos unas operaciones a ese HashName (LOCAL.1) y al serial introducido (ARG.2).**
3. **Si EAX <> 0 entonces serial correcto.**

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.



5. Entendiendo la comprobación del serial



En resumen:

1. **Necesitamos que EAX <> 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.

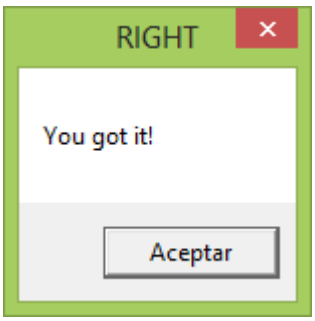
6. Generando el serial válido

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
        jl @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/