

## Crackme La Roca by Flamer [x64DBG][MinGWC++][SERIAL][LUISFECAB]

---



Software	Crackme La Roca by Flamer
Protección	Serial.
HERRAMIENTAS	<p>Windows 7 Home Premium SP1 x32 Bits (SO donde trabajamos) x64DBG (Feb 14 2018) Detect It Easy (DIE) v1.01</p> <p><a href="#">DESCARGAR HERRAMIENTAS</a></p> <p><a href="#">DESCARGAR TUTO+ARCHIVOS</a></p>
SOLUCIÓN	SERIAL.
AUTOR	LUISFECAB
RELEASE	Junio 7 2019 [TUTORIAL 015]

# INTRODUCCIÓN

Sigo con este nuevo tutorial para no perder el impulso y las ganas de escribir que tengo en estos momentos, y que es gracias, al lograr vencer el **Crackme** <[RandomWeird-Single.by.nextco](#)>, que ya compartí el tutorial con todos ustedes. Eso me ha recargado de ganas para seguir escribiendo tutos que tenía pendientes; aunque escribirlos toma su tiempo y dedicación pero no importa porque me gusta mucho hacerlos.

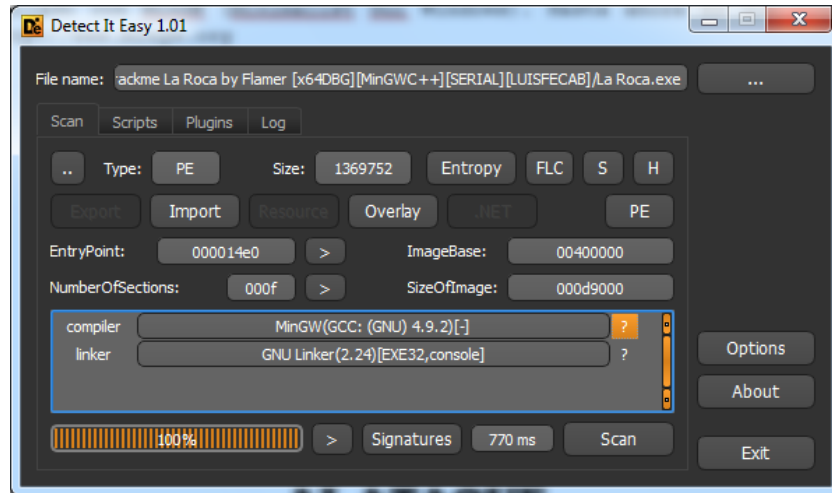
Hoy, Mayo 8, empezaré a escribir el tutorial del **Crackme** <[La.Roca](#)> hecho por **Flamer**. Este **Crackme** lo había resuelto ya hace unos meses atrás pero no había hecho el tutorial porque estaba obsesionado con poder completar el <[RandomWeird-Single.by.nextco](#)> y si leyeron mi tuto anterior sabrán que se me hizo difícil, ese **Crackme** tiene su miga.

<[La.Roca](#)> nos ofrece buscar un **SERIAL VÁLIDO**, es un **Crackme** ideal para practicar y mejorar nuestras habilidades cuando se trata de buscar o hallar seriales. Este **Crackme** nos permitirá desarrollar nuestro "*ojo de cracker*".

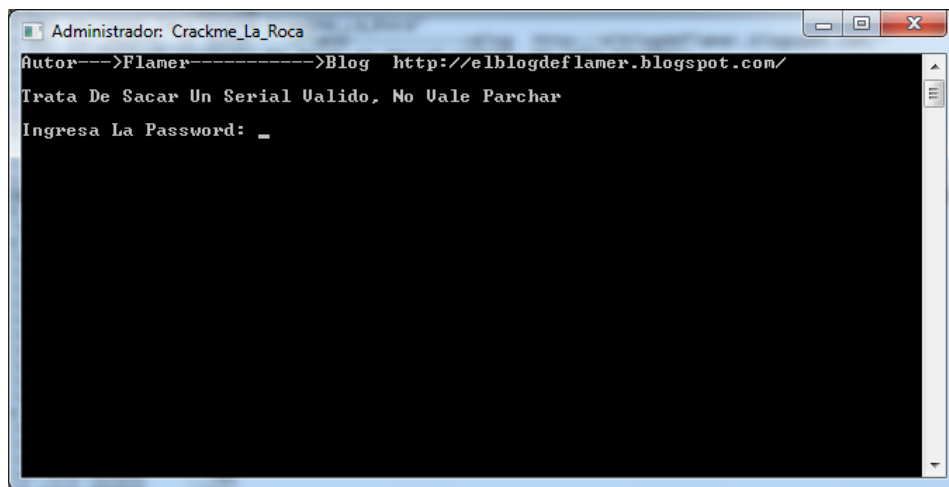
Para terminar esta pequeña introducción, quiero como siempre saludar a la lista de **CracksLatinoS** y por supuesto a mis amigos de **PeruCrackerS**.

## ANALISIS INICAL

Para empezar lo analizaremos con el <Detect It Easy (DIE) v1.01> y si trae algo raro.

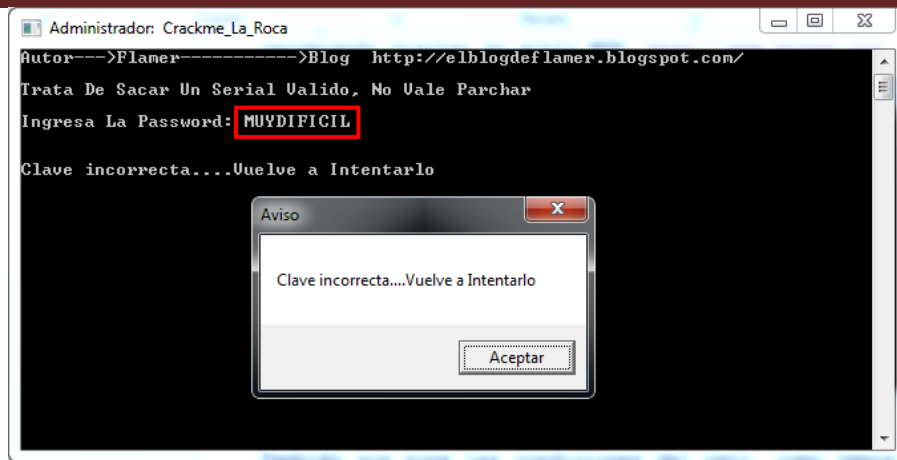


Fue compilado con **MinGW** (**Min**imalist **G**NU for **W**indows). Hasta ahora es que conozco el **MinGW**, desarrollado para hacer aplicaciones nativas de Windows en C++, y es por eso que en nombre del tutorial lo puse como **MinGWC++**. Para más información te recomiendo visitar su sitio WEB, <http://www.mingw.org>.



Vemos que fue hecho por **Flamer** y nos deja la dirección para que visitemos de blog. También nos pone las condiciones del reto, como vemos es sacar un **SERIAL VÁLIDO**. Metamos un **SERIAL** y probemos a ver qué nos sale. Hace días que no utilizaba mi **SERIAL** de batalla pero hoy llegó su día de utilizarlo, "**MUYDIFICIL**".

# Crackme La Roca by Flamer [x64DBG][MinGWC++][SERIAL][LUISFECAB]



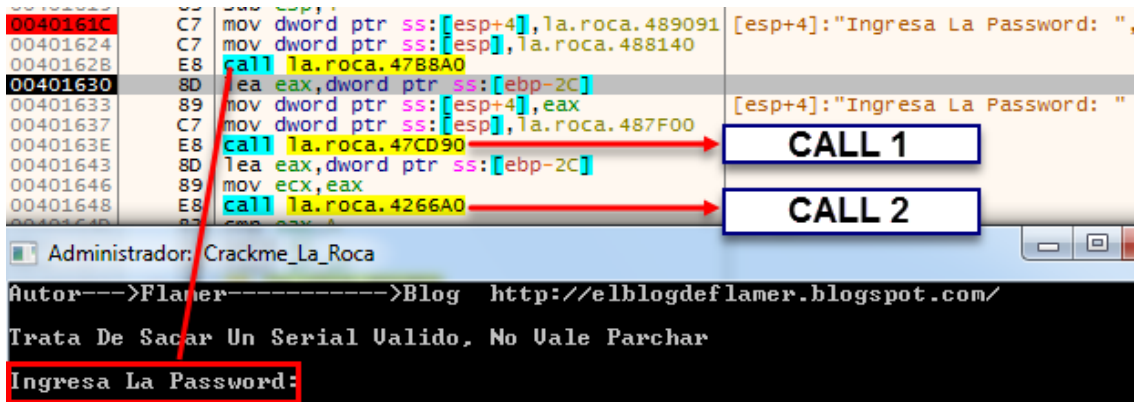
Era lo lógico, que nos diera a conocer al <CHICO MALO> que nos muestra un mensaje de "Clave incorrecta...". Y si vemos en la consola, también aparece "Clave incorrecta....Vuelve a Intentarlo". Eso nos indica que lo primero que podemos hacer es buscar las **Strings** y ver si por ahí podemos atacarle.

Ya con lo analizado podemos decir que no tiene nada raro y como vimos esta hecho en **C++** compilado con **MinGW**. Listo, y nos vamos al **AL ATAQUE**, que siempre es la mejor parte del tutorial para mi gusto.

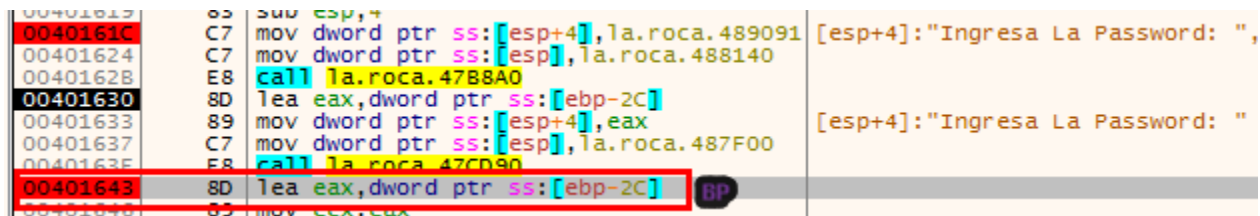


## Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

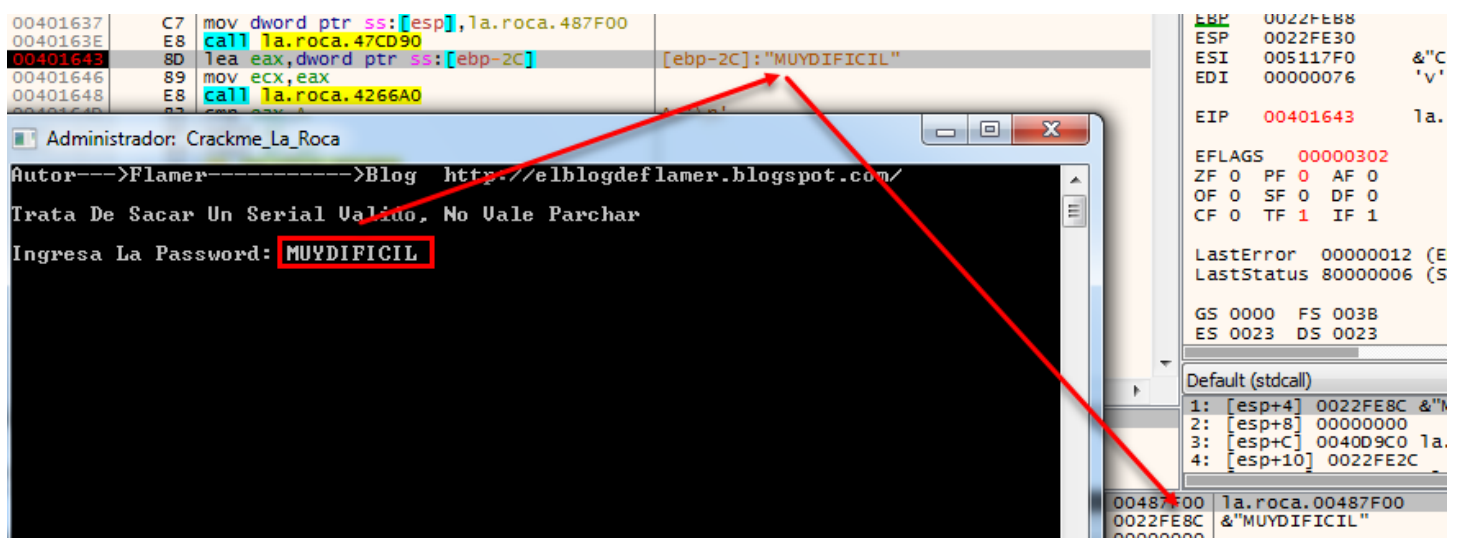
Nos detuvimos en **0040161C** y vemos que en **[esp+4]** tiene la **String** "Trata De Sacar Un Serial Valido", no es la que inicialmente vimos, pero si vemos en **la.roca.489091** tiene la **String** "Ingresa La Password:" que será cargada y que en **0040162B** **call** **la.roca.47B8A0** se cargara la **String** en consola. Traciamos con <F8> hasta pasar ese **CALL** y así evitar entrar en él.



Estamos parados en **00401630**, hemos pasado ese **CALL** y podemos ver que ya en consola se cargó la **String** "Ingresa La Password:". Nos quedan los otros **CALL** y que por lo que explicamos en el tutorial anterior, **1682** podemos decir que el **0040163E** **call** **la.roca.47CD90** se terminará de cargar la consola y ahí mismo será capturado nuestro **SERIAL** o **PASSWORD**. Lo que haremos será poner un <BREAKPOINT> en **00401643** para detenernos cuando hallamos ingresado nuestro **SERIAL**.



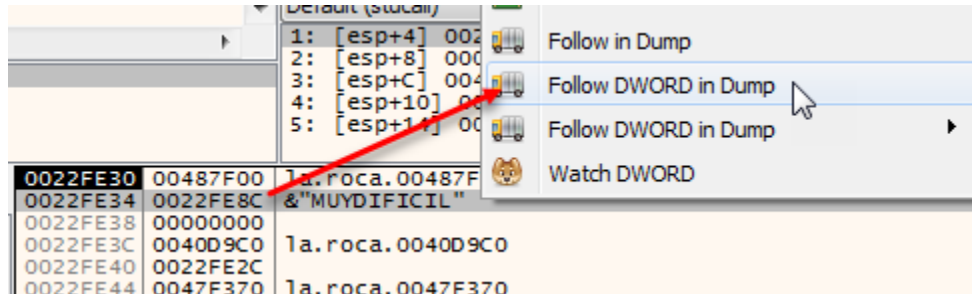
Ahí vemos puesto nuestro <BREAKPOINT>. Continuemos con la ejecución del programa hasta que nos detengamos en ese último <BREAKPOINT>. Para que no se pierdan, el **Crackme** se ejecuta completamente y nosotros debemos ingresar nuestro **SERIAL** y probarlo para llegar al <BREAKPOINT>, mi **SERIAL** de batalla, "MUYDIFICIL".



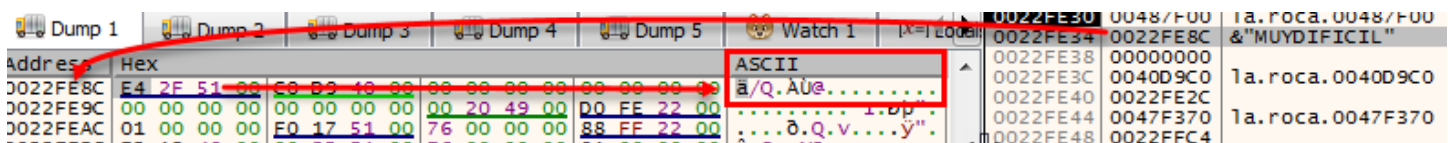


# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

Nos detuvimos donde queríamos después del `call 1a.roca.47CD90` en `00401643`, y si seguimos las **FLECHAS ROJAS** de la imagen de arriba nos muestran que en `[ebp-2C]` se encuentra almacenado nuestro **SERIAL**, "MUYDIFICIL", y que la **PILA** o **STACK** nos muestra y confirma que efectivamente en `0022FE8C` se almacena el **SERIAL**. Busquemos nuestro **SERIAL** en memoria, veamos la dirección `0022FE8C` en el **DUMP**.



Con <Clic Derecho>Follow DWORD in Dump>. Veamos qué nos muestra el **DUMP**.



No está nuestro **SERIAL**, en la columna **ASCII** no hay nada parecido a "MUYDIFICIL", pero si en la **PILA** si aparece y entonces como lo entendemos. Bueno, bueno, ya mis avezados **Crackers** sabrán la respuesta, bueno en realidad no tengo una respuesta para eso, solo puedo compartir lo que yo entiendo, puedo estar equivocado que es lo más seguro pero mi explicación es que debemos recordar que las direcciones en corchetes (`[]`) como es nuestro caso `[ebp-2C]`, nos indican que debemos trabajar con el valor almacenado en esa dirección de memoria no con la dirección en sí misma. En nuestro caso la dirección `0022FE8C` contiene los siguientes valores `E42F5100` y no nuestro **SERIAL**, pues resulta que ese valor `E42F5100`, viene siendo una dirección en donde se almacena nuestro **SERIAL**. Bien, dejemos esa explicación hasta ahí que seguro ya todos saben eso, es que a veces me desvío del tema principal y me pongo a explicar otras cosas.

Estoy retomando el escrito de este tutorial que lo dejé abandonado ya casi un mes porque me enfraqué en una feroz lucha contra un reto programado por **Bym24v**. Terminando este tuto me pongo a escribir el de ese reto, llamado <Ret0>. Dejemos lo anterior y continuemos con el análisis que ya ni sé por dónde iba. Retomemos, lo que sigue es en `00401648` `call 1a.roca.4266A0`, que lo que hace es cargar en **EAX** la longitud de nuestro **SERIAL**. Es muy importante que conozcamos de antemano la longitud del **SERIAL** que usemos, no solo en este caso, si no siempre, porque cuando estemos crackeando y notemos que se carga un valor podamos sospechar que puede ser la longitud de nuestro **SERIAL** y que a lo mejor ha de ser utilizada para algún cálculo o comparación, y que para en este caso es una comparación.

# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

**COMPARA NUESTRA LONGITUD**

**MUEVE MI SERIAL A EAX. UNA INSTRUCCIÓN BASURA PORQUE LUEGO EAX RECIBE OTRO VALOR.**

**MOVERÁ NUESTRA LOGITUD DEL SERIAL AL REGISTRO EAX QUE EN MI CASO ES 0xA (10).**

Hide FPU

EAX	00622FE4	"MUYDIFICIL"
EBX	0022FEA0	
ECX	0022FE8C	&"MUYDIFICI"
EDX	00000012	
EBP	0022FEB8	
ESP	0022FE2C	
ESI	006217F0	&"C:\\Users
EDI	00000076	'v'

eax=00622FE4 "MUYDIFICIL"  
dword ptr [eax-C]=[00622FD8]=A '\n'  
.text:004266A2 la.roca.exe:\$266A2 #25AA2

Bien, *"una imagen vale que mil palabras"*. Sigamos las flechas en la captura de arriba que resume lo que hace ese **CALL**, que en realidad no es nada solo cargar nuestra longitud. Observemos las siguientes instrucciones en donde compara nuestra longitud y después tomará o no un salto.

**AL HACER LA COMPARACIÓN Y COMO EAX=0x10, ENTONCES SE ACTIVARA ZF=1**

**PASANDO LA INSTRUCCIÓN PARA HACER LA COMPARACIÓN**

**COMO HAY IGUALDAD, ZF=1**

Hide FPU

EAX	0000000A				
EBX	0022FEA0				
ECX	0022FE8C				
EDX	00000012				
EBP	0022FEB8				
ESP	0022FE30				
ESI	006217F0				
EDI	00000076				
EIP	0040164D				
EFLAGS	00000206				
ZE	0	PF	1	AF	0
OF	0	SE	0	DF	0
CF	0	TF	0	IF	1

Hide FPU

EAX	0000000A				
EBX	0022FEA0				
ECX	0022FE8C				
EDX	00000012				
EBP	0022FEB8				
ESP	0022FE30				
ESI	006217F0				
EDI	00000076				
EIP	00401650				
EFLAGS	00000246				
ZE	1	PF	1	AF	0
OF	0	SF	0	DF	0
CF	0	TF	0	IF	1

Como vemos se nos activó la **FLAG-Z=1**, y estamos parados en **00401650 sete al**, la instrucción **SETE** lo que hace cargar el valor que tiene **FLAG-Z** a un registro y para este caso es **EAX** pero en la posición **AL** que es el último **BYTE** de **EAX**. Si hemos



## Crackme La Roca by Flamer [x64DBG][MinGWC++][SERIAL][LUISFECAB]

hecho el curso del **Maestro Ricardo**, [OLLY DESDE CERO](#), podremos recordar lo que indica **AL** en el registro **EAX**. Luego hace un **TEST** en **00401653 test al,al** y de ahí entramos al salto.

```
0040163E E8 call la.roca.47CD90
00401643 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
00401646 89 mov ecx,eax
00401648 E8 call la.roca.4266A0
0040164D 83 cmp eax,A A: '\n'
00401650 0F sete al
00401653 84 test al,al
00401655 0F je la.roca.4018F6
00401658 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
0040165E C7 mov dword ptr ss:[esp],1
00401665 89 mov ecx,eax
00401667 E8 call la.roca.4541B0
0040166C 83 sub esp,4
0040166F 0F movzx eax,byte ptr ds:[eax]
00401672 0F movsx eax,al
00401675 89 mov dword ptr ss:[ebp-20],eax
00401678 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
00401683 83 sub esp,4
00401686 8B jmp la.roca.4018F4
00401688 8D lea eax,dword ptr ss:[ebp-30]
0040168B C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
0040168E C7 mov dword ptr ss:[ebp-68],3
00401691 89 mov ecx,eax
00401694 E8 call la.roca.454100
00401697 83 sub esp,4
0040169A 8B jmp la.roca.4018F4
0040169D 8B jmp la.roca.4018F4
004016A0 8D lea eax,dword ptr ss:[ebp-30]
004016A3 C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
004016A6 C7 mov dword ptr ss:[ebp-68],3
004016A9 89 mov ecx,eax
004016AC E8 call la.roca.454100
004016AF 83 sub esp,4
004016B2 8B jmp la.roca.401911
004016B5 8B jmp la.roca.401911
004016B8 8D lea eax,dword ptr ss:[ebp-30]
004016BB C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
004016BE C7 mov dword ptr ss:[ebp-68],3
004016C1 89 mov ecx,eax
004016C4 E8 call la.roca.454100
004016C7 83 sub esp,4
```

EAX 00000001  
EBX 0022FEA0  
ECX 0022FE8C  
EDX 00000012  
EBP 0022FE88  
ESP 0022FE30  
ESI 006217F0  
EDI 00000076  
EIP 00401655  
EFLAGS 00000202  
ZF 0 PF 0 AF 0  
OF 0 SF 0 DF 0  
CF 0 TF 0 IF 1

Pasando la instrucción **TEST** que activa nuestra **FLAG-Z=0**, y con eso no tomamos el salto **00401655 je la.roca.4018F6** que nos enviará al **<CHICO MALO>**. Sigamos el salto para corroborar lo que dijimos.

```
0040163E E8 call la.roca.47CD90
00401643 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
00401646 89 mov ecx,eax
00401648 E8 call la.roca.4266A0
0040164D 83 cmp eax,A A: '\n'
00401650 0F sete al
00401653 84 test al,al
00401655 0F je la.roca.4018F6
00401658 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
0040165E C7 mov dword ptr ss:[esp],1
00401665 89 mov ecx,eax
00401667 E8 call la.roca.4541B0
0040166C 83 sub esp,4
0040166F 0F movzx eax,byte ptr ds:[eax]
00401672 0F movsx eax,al
00401675 89 mov dword ptr ss:[ebp-20],eax
00401678 8D lea eax,dword ptr ss:[ebp-2C] [ebp-2C]: "MUYDIFICIL"
00401683 83 sub esp,4
00401686 8B jmp la.roca.4018F4
00401688 8D lea eax,dword ptr ss:[ebp-30]
0040168B C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
0040168E C7 mov dword ptr ss:[ebp-68],3
00401691 89 mov ecx,eax
00401694 E8 call la.roca.454100
00401697 83 sub esp,4
0040169A 8B jmp la.roca.4018F4
0040169D 8B jmp la.roca.4018F4
004016A0 8D lea eax,dword ptr ss:[ebp-30]
004016A3 C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
004016A6 C7 mov dword ptr ss:[ebp-68],3
004016A9 89 mov ecx,eax
004016AC E8 call la.roca.454100
004016AF 83 sub esp,4
004016B2 8B jmp la.roca.401911
004016B5 8B jmp la.roca.401911
004016B8 8D lea eax,dword ptr ss:[ebp-30]
004016BB C7 mov dword ptr ss:[esp],la.roca.4890D8 "Clave incorrecta"
004016BE C7 mov dword ptr ss:[ebp-68],3
004016C1 89 mov ecx,eax
004016C4 E8 call la.roca.454100
004016C7 83 sub esp,4
```

Podemos observar que hay varios **<CHICOS MALOS>** y como lo dijimos al inicio de esta sección cuando buscamos las **Strings** supusimos que debían de haber varias comprobaciones y con este lo podemos comprobar. Me imagino que ya se dieron cuenta que mi **SERIAL** le acertó a la longitud, primera vez que le acierto a algo cuando crackeo desde un inicio. Entonces, la longitud de nuestro **SERIAL VÁLIDO** es de **0xA=10**.

Vamos bien hasta ahora, ya pasamos la primera comprobación. Sigamos traceando con **<F8>** para evitar entrar **00401667 call la.roca.4541B0** que lo que hará es colocar una

# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

parte de nuestro **SERIAL** para luego trabajar con él. Lleguemos hasta la dirección **0040166F**.

```
0040163E E8 call la.roca.47CD90
00401643 8B mov ebx,dword ptr ss:[ebp-2C] [ebp-2C]:"MUYDIFICIL"
00401646 8B mov ebx,dword ptr ss:[ebp-2C] [ebp-2C]:"MUYDIFICIL", eax:"UYDIF"
00401648 8B mov ebx,dword ptr ss:[ebp-2C] [ebp-2C]:"MUYDIFICIL", A:'\n'
00401650 8B mov ebx,dword ptr ss:[ebp-2C] [ebp-2C]:"MUYDIFICIL"
00401653 84 test al,al
00401655 0F je la.roca.4018F6
00401658 8D lea eax,dword ptr ss:[ebp-20] [ebp-20]:"MUYDIFICIL"
0040165E C7 mov dword ptr ss:[esp],1
00401665 89 mov ecx,eax
00401667 E8 call la.roca.454180
0040166C 83 sub esp,4
0040166F 0F movzx eax,byte ptr ds:[eax] eax:"UYDIFICIL"
00401672 0F movsx eax,al eax:"UYDIFICIL"
00401675 89 mov dword ptr ss:[ebp-20],eax [ebp-20]:"MUYDIFICIL"
00401678 8D lea eax,dword ptr ss:[ebp-20] [ebp-20]:"MUYDIFICIL"
00401678 C7 mov dword ptr ss:[esp],5
00401682 89 mov ecx,eax
00401684 E8 call la.roca.454180
00401689 83 sub esp,4
0040168C 0F movzx eax,byte ptr ds:[eax] eax:"UYDIFICIL"
0040168F 0F movsx eax,al eax:"UYDIFICIL"
00401692 89 mov dword ptr ss:[ebp-24],eax [ebp-24]:"MUYDIFICIL"
00401695 83 cmp dword ptr ss:[ebp-20],35 35:'5'
00401699 0F je la.roca.4018D7
```

Hide FPU

EAX	00622FE5	"UYDIFICIL"
EBX	0022FEA0	
ECX	0022FE8C	&"MUYDIFICIL"
EDX	00000000	
EBP	0022FEB8	
ESP	0022FE30	
ESI	006217F0	&"C:\\Users\\I
EDI	00000076	'v'
EIP	0040166F	la.roca.004016

EFLAGS 00000206  
ZF 0 PF 1 AF 0  
OF 0 SF 0 DF 0  
CF 0 TF 0 IF 1

eax=006C2FE5 "UYDIFICIL"  
byte ptr [eax]=[006C2FE5 "UYDIFICIL"]='U'  
.text:0040166F la.roca.exe:\$166F #A6F

Como dijimos en la imagen, se cargará el **SERIAL** desde el Segundo carácter y por eso tenemos en **0040165E** `mov dword ptr ss:[esp],1`, que será la posición del carácter que no se mostrará de ahí para atrás. Y todo eso solo para coger el segundo carácter **"MUYDIFICIL"**, que es **"U"** y guardarlo en la dirección **[ebp-20]** y esto lo hace en **00401675** `mov dword ptr ss:[ebp-20],eax`. Vamos a tracear hasta que guarde el carácter **"U"** en memoria.

```
00401667 E8 call la.roca.454180
0040166C 83 sub esp,4
0040166F 0F movzx eax,byte ptr ds:[eax]
00401672 0F movsx eax,al
00401675 89 mov dword ptr ss:[ebp-20],eax [ebp-20]:"MUYDIFICIL"
00401678 8D lea eax,dword ptr ss:[ebp-20] [ebp-20]:"MUYDIFICIL"
00401678 C7 mov dword ptr ss:[esp],5
00401682 89 mov ecx,eax
00401684 E8 call la.roca.454180
00401689 83 sub esp,4
0040168C 0F movzx eax,byte ptr ds:[eax]
0040168F 0F movsx eax,al
00401692 89 mov dword ptr ss:[ebp-24],eax [ebp-24]:"MUYDIFICIL"
00401695 83 cmp dword ptr ss:[ebp-20],35 35:'5'
00401699 0F je la.roca.4018D7
```

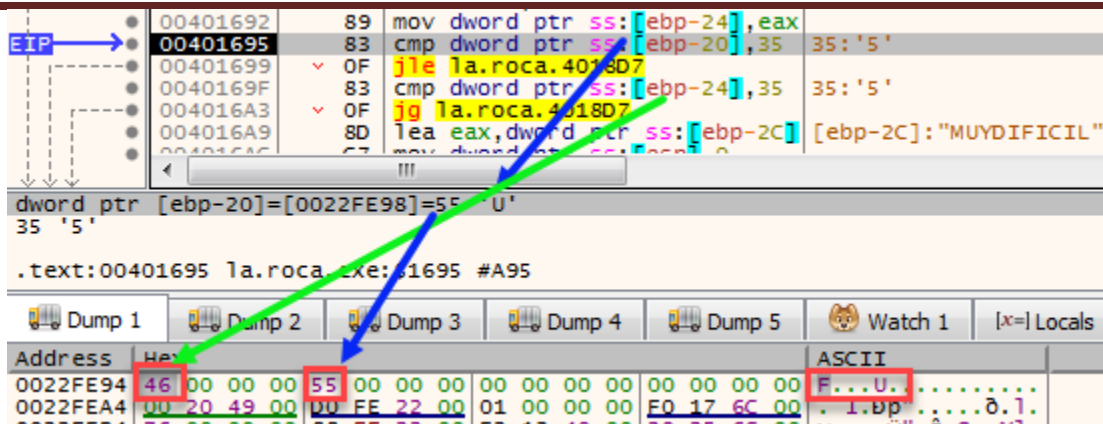
AHORA TOMARÁ EL SEXTO CARACTER Y LO GUARDARÁ EN MEMORIA

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Wat

Address	Hex	ASCII
0022FE98	55 00 00 00	U.....
0022FEA8	D0 FE 22 00	Db"....
0022FEB8	88 FF 22 00	.y".â.@
0022FEC8	01 00 00 00	....â.@
0022FED8	68 14 6C 00	h 1

Estamos detenidos en **00101678** y podemos ver en el **DUMP** que en **0022FE98** guardó **0x55**, **"U"**. Miremos lo **RESOLTADO EN ROJO**, vemos que ahora volverá a hacer el mismo procedimiento pero para tomar el Sexto carácter de nuestro **"MUYDIFICIL"**, **"F"** y guardarlo en memoria. Entonces, seguiremos traceando con **<F8>** para no entrar a ese **CALL**. Lleguemos hasta la comparación en **00401695** `cmp dword ptr ss:[ebp-20],35`.

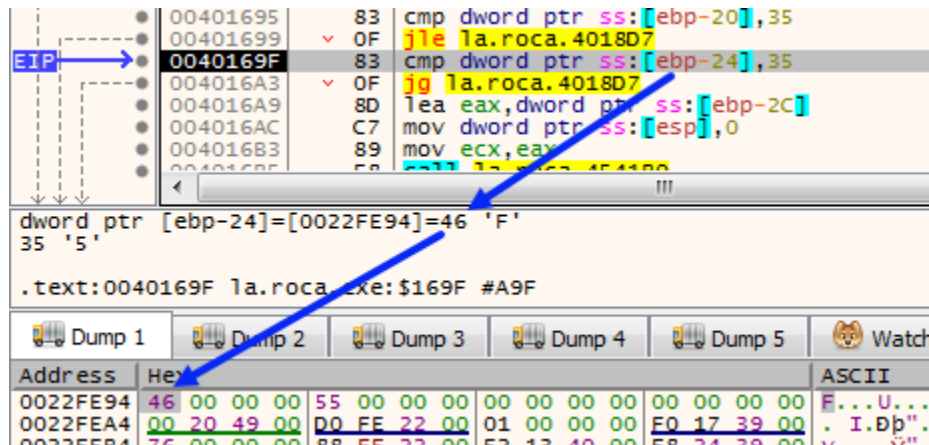
# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]



Estando detenidos en **00401695**, vemos que va a comparar el valor **HEXA** de nuestro segundo carácter, "U" (0x55) con 0x35, para ver si toma el salto o no en **00401699 jle la.roca.4018D7**.

JLE	Jump if less or equal	signed	ZF = 1 or SF <> OF
JNG	Jump if not greater		

La idea es no tomar los saltos porque todos nos llevan al <CHICO MALO>. Si observamos **JLE** no salta si el valor comparado es mayor, y para nuestro caso, **0x55 > 0x35**, entonces cumplimos la condición y no saltaremos, con eso podemos decir que el segundo carácter, "U", es correcto en el **SERIAL**. Eso es un segundo acierto, ya le atiné a la longitud y al Segundo carácter. Luego, en **0040169F** se hace otra comparación pero con el valor **HEXA** de nuestro sexto carácter, "F" (0x46). Sigamos traceando hasta llegar a ese lugar.



Aquí hay otro salto, y es un **JG**. Miremos qué condición activa este salto.

JG	Jump if greater	signed	ZF = 0 and SF = OF
JNLE	Jump if not less or equal		

Se activa cuando nuestro valor a comparar es mayor. Hasta aquí me llegó la suerte con nuestro **SERIAL** porque el valor **HEXA** de "F" es mayor, miremos, **0x46 > 0x35**. Con eso se activará el salto y vamos al <CHICO MALO>.

# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

```
00401695 83 cmp dword ptr ss:[ebp-20],35
00401699 v 0F jle la.roca.4018D7
0040169F 83 cmp dword ptr ss:[ebp-24],35
004016A3 v 0F jg la.roca.4018D7
004016A9 8D lea eax,dword ptr ss:[ebp-2C]
004016AC C7 mov dword ptr ss:[esp],0
004016B3 89 mov ecx,eax
004016B5 E8 call la.roca.4541B0
```

Clarito se ve que tomamos el salto. Entonces, debemos escoger un carácter que tenga un valor **HEXA** menor o igual a **0x35**.

Dec.	Hex.	Carac
32	20	esp
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	.
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5

VOY A ESCOGER EL  
5 (0x35)

Esos son los caracteres que cumplen la condición. Vamos a escoger el "5" (0x35), quedando nuestro nuevo serial, **MUYDI5ICIL**. Coloquemos un <BREAKPOINT> en ese salto.

```
00401695 83 cmp dword ptr ss:[ebp-20],35
00401699 v 0F jle la.roca.4018D7
0040169F 83 cmp dword ptr ss:[ebp-24],35
004016A3 v 0F jg la.roca.4018D7
004016A9 8D lea eax,dword ptr ss:[ebp-2C]
004016AC C7 mov dword ptr ss:[esp],0
```

Quitemos los <BREAKPOINT> que habíamos puesto al inicio para evitar parar en lugares que ya no nos interesan. Reiniciemos todo e ingresamos nuestro nuevo **SERIAL** y lo probamos.

# Crackme La Roca by Flamer [x64DBG][MinGWC++][SERIAL][LUISFECAB]

00401695	83	cmp dword ptr ss:[ebp-20],35	
00401699	0F	jle la.roca.4018D7	
0040169F	83	cmp dword ptr ss:[ebp-24],35	
004016A3	0F	je la.roca.4018D7	
004016A9	8D	lea eax,dword ptr ss:[ebp-2C]	
004016AC	C7	mov dword ptr ss:[esp],0	
004016B3	89	mov ecx,eax	
004016B5	E8	call la.roca.454180	TOMARA PRIMER CARACTER, "M".
004016BA	83	sub esp,4	
004016BD	0F	movzx eax,byte ptr ds:[eax]	
004016C0	0F	movsx eax,al	
004016C3	89	mov dword ptr ss:[ebp-70],eax	
004016C6	8D	lea eax,dword ptr ss:[ebp-2C]	
004016C9	C7	mov dword ptr ss:[esp],4	
004016D0	89	mov ecx,eax	
004016D2	E8	call la.roca.454180	TOMARA QUINTO CARACTER, "I".
004016D7	83	sub esp,4	
004016DA	0F	movzx eax,byte ptr ds:[eax]	
004016DD	0F	movsx eax,al	
004016E0	03	add eax,dword ptr ss:[ebp-70]	SUMA LOS HEXA DE CARACTER, "M" "I". 0x4D+0x49
004016E3	89	mov dword ptr ss:[ebp-1C],eax	
004016E6	83	sub dword ptr ss:[ebp-1C],60	
004016EA	83	cmp dword ptr ss:[ebp-1C],8	LUEGO LE RESTA 0x60 Y LO COMPARA CON 0x8. PARA EVITAR EL JLE, RESTA >0x8.
004016EE	0F	jle la.roca.4018B8	
004016F4	8D	lea eax,dword ptr ss:[ebp-2C]	

Perfecto, el salto no se toma y seguimos por buen camino. Miremos lo **RESALTADO EN AZUL**, vamos a tomar el Primer y Quinto carácter de "MUYDI5ICIL", "M", "I". Los sumará,  $0x4D+0x49=0x96$  y a esa suma le restará  $0x60$ ,  $0x96-60=0x36$ . Recordemos la condición que activa el salto **JLE**.

<b>JLE</b>	Jump if less or equal	signed	ZF = 1 or SF <> OF
<b>JNG</b>	Jump if not greater		

El resultado debe ser mayor para no saltar,  $0x36 > 0x8$ . Perfecto, no saltaremos y seguimos por buen camino. Nuestro **SERIAL** va quedando así, "MUYDI5ICIL".

```

004016E0 03 add eax,dword ptr ss:[ebp-70]
004016E3 89 mov dword ptr ss:[ebp-1C],eax
004016E6 83 sub dword ptr ss:[ebp-1C],60
004016EA 83 cmp dword ptr ss:[ebp-1C],8
004016EE 0F jle la.roca.4018B8
004016F4 8D lea eax,dword ptr ss:[ebp-2C]
004016F7 C7 mov dword ptr ss:[esp],0

```

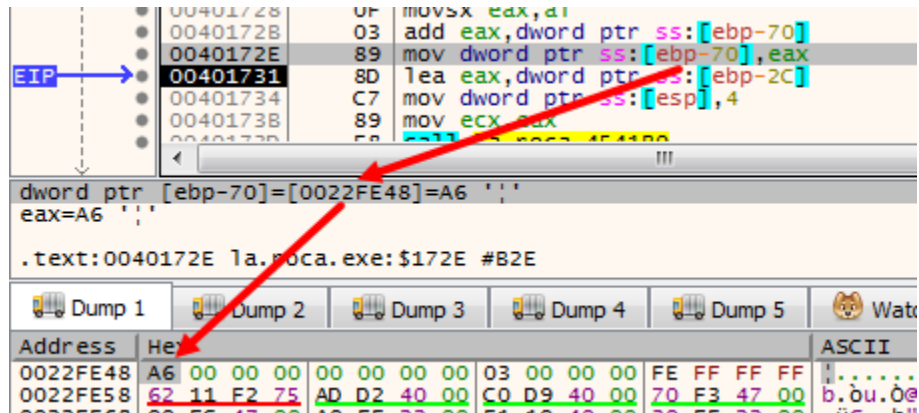
De lo anterior deducimos que la suma del Primer y Quinto carácter debe ser mayor o igual a  $0x69$ . Ahora se tomarán otros caracteres y veremos qué hace con ellos.

004016F7	C7	mov dword ptr ss:[esp],0	
004016FE	89	mov ecx,eax	
00401700	E8	call la.roca.454180	TOMA PRIMER CARACTER, "M".
00401705	83	sub esp,4	
00401708	0F	movzx eax,byte ptr ds:[eax]	
0040170B	0F	movsx eax,al	
0040170E	89	mov dword ptr ss:[ebp-70],eax	
00401711	8D	lea eax,dword ptr ss:[ebp-2C]	
00401714	C7	mov dword ptr ss:[esp],2	
0040171B	89	mov ecx,eax	
0040171D	E8	call la.roca.454180	TOMA TERCER CARACTER, "Y".
00401722	83	sub esp,4	
00401725	0F	movzx eax,byte ptr ds:[eax]	
00401728	0F	movsx eax,al	
0040172B	03	add eax,dword ptr ss:[ebp-70]	SUMA CARACTER, "M", "Y". GUARDA EN EL DUMP
0040172E	89	mov dword ptr ss:[ebp-70],eax	
00401731	8D	lea eax,dword ptr ss:[ebp-2C]	
00401734	C7	mov dword ptr ss:[esp],4	
0040173B	89	mov ecx,eax	
0040173D	E8	call la.roca.454180	TOMA QUINTO CARACTER, "I".

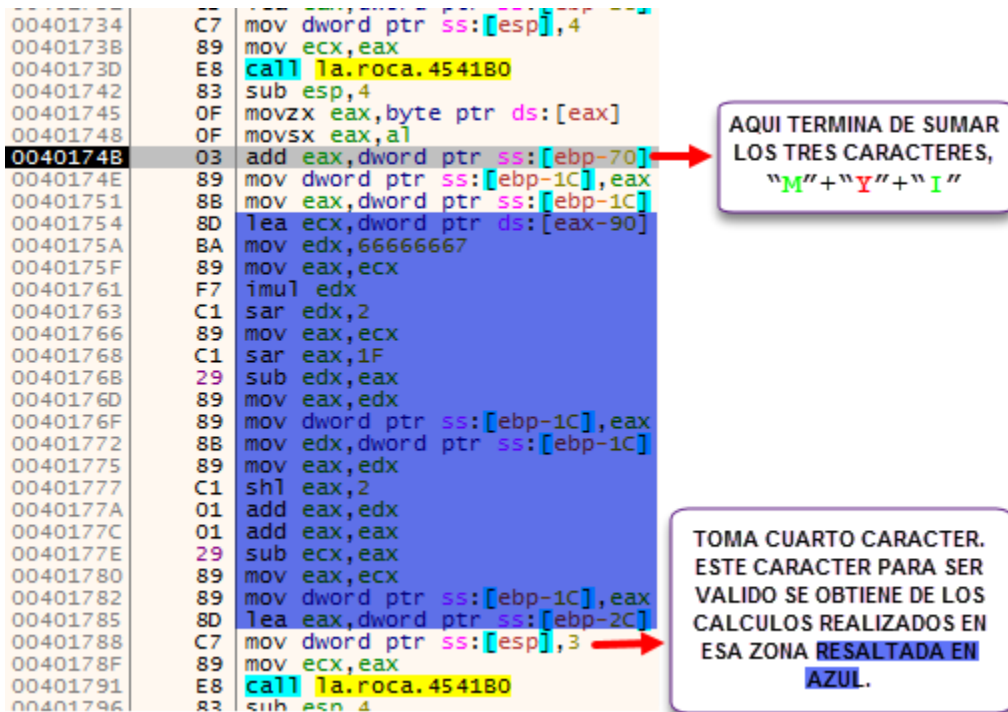


## Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

Sigue jugando con nuestro **SERIAL**. Se puso más extremo, ahora toma tres caracteres, y son en su orden Primero, Tercero y Quinto carácter que serían "MUYDI5ICIL", "M", "Y", "I". Suma "M"+"Y",  $0x4D+0x59=0xA6$  y lo guarda en memoria. Tracemos hasta cuando guarda el valor sumado.



Ahí lo guardó, seguiremos traceando con los ojos bien abiertos para ver qué hace con ese valor y con el Quinto carácter.



Pues lo que hace es sumarle a  $0xA6$  el valor **HEXA** de nuestro Quinto carácter,  $0x49$ , y guardarlo en memoria,  $0xA6+0x49=0xEF$ . En otras palabras, "M"+"Y"+"I". Se puso marullero el reto, observemos la zona **RESALTADA EN AZUL**. Ahora tenemos una serie de cálculos que toma como base el valor obtenido de sumar nuestros tres caracteres de arriba. Puede confundirnos al inicio pero lo importante es el resultado final que obtengamos de esas instrucciones porque ese valor nos da la pista para obtener el Cuarto carácter válido en nuestro **SERIAL** que estamos hallando. Si miramos, podemos darnos cuenta que después de la zona **RESALTADA EN AZUL** se tomará el cuarto carácter para más adelante realizar algún tipo de validación, eso lo veremos un

## Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

poco más abajo. La zona **RESALTADA EN AZUL** es importante si deseas hacer un **KeyGen**, que en este mismo instante que escribo estas palabras no creo que lo vaya a hacer, es que me está ganando la pereza, además **Flamer** nos pide hallar un **SERIAL VÁLIDO**. Vamos a tracear hasta donde guardemos el valor calculado en la zona **RESALTADA EN AZUL** y eso es hasta la dirección **00401785**.

```
00401782 89 mov dword ptr ss:[ebp-1C],eax
00401785 8D lea eax,dword ptr ss:[ebp-2C]
00401788 C7 mov dword ptr ss:[esp],3
0040178F 89 mov ecx,eax
00401791 E8 call la.roca.454180
00401796 83 sub esp,4
00401799 0F movzx eax,byte ptr ds:[eax]
0040179C 0F movsx eax,al
0040179F 83 sub eax,30
004017A2 89 mov dword ptr ss:[ebp-20],eax
004017A5 8B mov eax,dword ptr ss:[ebp-1C]
004017A8 3B cmp eax,dword ptr ss:[ebp-20]
004017AB 0F jne la.roca.401898
```

Register	Value
EAX	00000005
EBX	0022FEA0
ECX	00000005
EDX	00000009
EBP	0022FE88
ESP	0022FE30
ESI	005A17F0
EDI	00000076

Address	Hex
0022FE9C	05 00 00 00
0022FEAC	01 00 00 00

El valor que obtenemos es **0x05**. Luego seguimos con la zona **RESALTADA EN VERDE**, ya sabemos que toma el Cuarto carácter de **"MUYDI5ICIL"**, **"D"**. Lo que hará es restarle **0x30** al valor **HEXA** de **"D"**, **0x44-0x30=0x14**. Lo que hay es una comparación entre el valor obtenido de la zona **RESALTADA EN AZUL** y el de la zona **RESALTADA EN VERDE**. En nuestro caso sería **CMP 5,14**. Y después un salto **JNE** y que no se debe activar, y para no tomarlo, los valores de la comparación deben ser iguales, y nosotros desgraciadamente tenemos valores diferentes, así que tomaremos el salto e iremos al **<CHICO MALO>**. Pero no hay problema, debemos hacer que la resta del Cuarto carácter nos dé como resultado **0x05** y para que eso ocurra el cuarto carácter debe valer **0x35** y ese carácter es el **"5"**. Así que nuestro nuevo **SERIAL** va quedando así, **"MUY5I5ICIL"**. Probemos este nuevo **SERIAL**, pongamos un **<BREAKPOINT>** en **004017AB** **jne la.roca.40189B**.

```
004017AB 0F jne la.roca.40189B
004017B1 8D lea eax,dword ptr ss:[ebp-2C]
004017B4 C7 mov dword ptr ss:[esp],6
004017BB 89 mov ecx,eax
004017BD E8 call la.roca.454180
004017C2 83 sub esp,4
004017C5 0F movzx eax,byte ptr ds:[eax]
004017C8 0F movsx eax,al
004017CB 89 mov dword ptr ss:[ebp-70],eax
004017CE 8D lea eax,dword ptr ss:[ebp-2C]
004017D1 C7 mov dword ptr ss:[esp],7
004017D8 89 mov ecx,eax
004017DA E8 call la.roca.454180
004017DF 83 sub esp,4
004017E2 0F movzx eax,byte ptr ds:[eax]
004017E5 0F movsx eax,al
004017E8 03 add eax,dword ptr ss:[ebp-70]
004017EB 89 mov dword ptr ss:[ebp-70],eax
004017EE 8D lea eax,dword ptr ss:[ebp-2C]
004017F1 C7 mov dword ptr ss:[esp],8
004017F8 89 mov ecx,eax
004017FA E8 call la.roca.454180
004017FF 83 sub esp,4
00401802 0F movzx eax,byte ptr ds:[eax]
00401805 0F movsx eax,al
00401808 03 add eax,dword ptr ss:[ebp-70]
0040180B 89 mov dword ptr ss:[ebp-1C],eax
0040180E 8B mov eax,dword ptr ss:[ebp-1C]
00401811 8D lea ecx,dword ptr ds:[eax-6B]
```

TOMA SEPTIMO CARACTER, "I".

TOMA OCTAVO CARACTER, "C".

TOMA NOVENO CARACTER, "I".

## Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

Estamos parados en **004017AE** y como vemos no tomamos el salto, así que vamos por buen camino. En la zona **RESALTADA EN AMARILLO**, prácticamente se va a realizar lo mismo que hicimos hace rato, solo que ahora se tomarán el Séptimo, Octavo y Noveno carácter de "MUY5I5ICIL", y los sumará, "I"+"C"+"I". En valores HEXA nos quedaría,  $0x49+0x43+0x49=0xD5$ .

The screenshot shows a debugger window with assembly code on the left and a memory dump on the right. A red arrow points from the instruction `jne la.roca.40187E` at address `00401866` to the memory dump at address `0022FE9C`, which contains the value `D5 00 00 00`.

Address	Hex
0022FE9C	D5 00 00 00
0022FEAC	01 00 00 00
0022FEB0	E2 13 40 00
0022FECC	E2 13 40 00

Ahí está, como podemos ver en la imagen de arriba el valor que obtuvimos de sumar los tres caracteres es  $0xD5$ , que se guarda en memoria, y en nuestra nueva zona **RESALTADA EN AZUL** se obtendrá un nuevo valor que nos servirá como pista para hallar el Décimo carácter válido. En la zona **RESALTADA EN VERDE**, trabajará con nuestro Décimo carácter de nuestro "MUY5I5ICIL", "L" ( $0x4C$ ), le restará  $0x30$  y ese valor de la resta será utilizado para hacer la comparación con el valor calculado en la zona **RESALTADA EN AZUL**.

# Crackme La Roca by Flamer [x64DBG][MinGWC++] [SERIAL][LUISFECAB]

```
0040183D 89 mov dword ptr ss:[ebp-1C],eax
00401840 8D lea eax,dword ptr ss:[ebp-2C]
00401843 C7 mov dword ptr ss:[esp],9
0040184A 89 mov ecx,eax
0040184C E8 call la.roca.454180
00401851 83 sub esp,4
00401854 0F movzx eax,byte ptr ds:[eax]
00401857 0F movsx eax,al
0040185A 83 sub eax,30
0040185D 89 mov dword ptr ss:[ebp-20],eax
00401860 8B mov eax,dword ptr ss:[ebp-1C]
00401863 3B cmp eax,dword ptr ss:[ebp-20]
00401866 75 jne la.roca.40187E
```

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5
Address	Hex			
0022FE9C	07 00 00 00	00 00 00 00	00 20 49 00	D0 FE 22 00
0022FEAC	01 00 00 00	F0 17 29 00	76 00 00 00	88 FF 22 00
0022FEB0	E2 13 40 00	F8 24 29 00	76 00 00 00	01 00 00 00

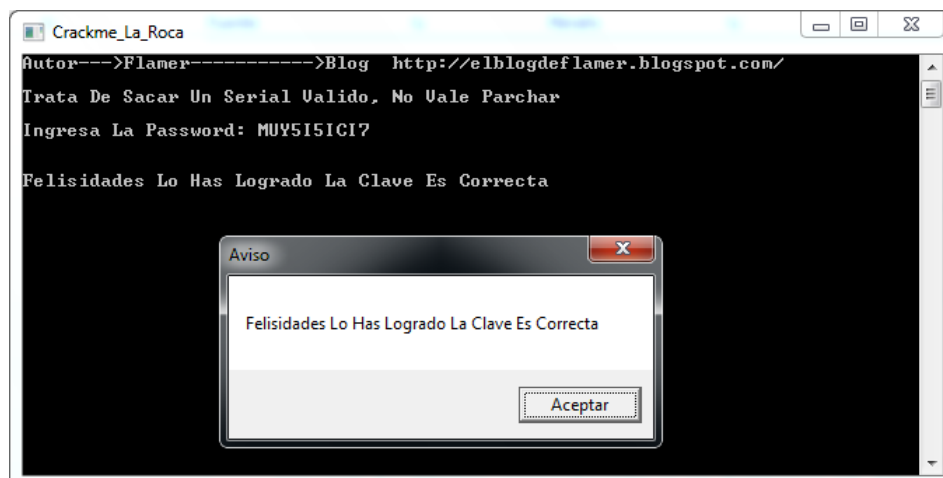
Pasamos la zona **RESALTADA EN AZUL** para saber el primer valor a comparar y es **0x07**, y el otro valor es, "L", **0x4C-0x30=0x1C**. La comparación sería **CMP 7,1C**; en realidad no importa el orden porque el salto **JNE** se activa si no hay igualdad y como vemos igualdad no tenemos, pero ya con estos datos podemos hallar nuestro Décimo carácter válido. El Décimo carácter al restarle **0x30** debe darnos **0x07** para nuestro caso sería "7", que es **0x37**. Listo, nuestro nuevo **SERIAL** queda así: "MUY5I5ICI7".

```
0040185D 89 mov dword ptr ss:[ebp-20],eax
00401860 8B mov eax,dword ptr ss:[ebp-1C]
00401863 3B cmp eax,dword ptr ss:[ebp-20]
00401866 75 jne la.roca.40187E
00401868 8D lea eax,dword ptr ss:[ebp-30]
0040186B C7 mov dword ptr ss:[esp],la.roca.4890A8
00401872 89 mov ecx,eax
00401874 E8 call la.roca.454100
00401877 83 sub esp,4
0040187C EB jmp la.roca.4018D5
0040187E 8D lea eax,dword ptr ss:[ebp-30]
00401881 C7 mov dword ptr ss:[esp],la.roca.4890D8
00401884 C7 mov dword ptr ss:[ebp-60],9
```

4890A8:"Felisidades Lo Has Logrado La Clave Es Correcta"  
ecx:&"MUY5I5ICIL"

4890D8:"Clave incorrecta...Vuelve a Intentarlo"

Tenga pa' que lleve, tomamos el salto hacia el <**CHICO MALO**> pero no importa porque esa era la última comprobación y ya hallamos el último carácter válido que era el Décimo. Entonces nuestro **SERIAL VÁLIDO** es "MUY5I5ICI7". Entonces los caracteres de **COLOR AZUL** son utilizados para comprobar la validez del serial y si queremos generar un **KeyGen** estos serán la base para sacar un **SERIAL VÁLIDO**. Probemos nuestro flamante **SERIAL**.



Perfecto, ya logramos sacar un **SERIAL VÁLIDO**. Terminamos por aquí.

## PARA TERMINAR

Un reto ideal para practicar con seriales. Como comenté, no voy a hacer el **KeyGen**. Espero que si alguno lee este pequeño tutorial se anime y haga un **KeyGen**, creo que este escrito le puede servir de guía para programarlo.

No hay más que decir, si apenas estás entrándole al Cracking y quieres mejorar tus habilidades este reto y este tutorial te pueden ser de mucha ayuda.

Me despido de todos mis amigos de **PeruCrackers** y **CracksLatinos**, no sin antes agradecerte a ti, mi respetado lector que ha sacado algo de tiempo para hacerlo.

*@LUISFECAB*