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

Software	**Crackme La Roca by Flamer**
Protección	**Serial.**
HERRAMIENTAS	Windows 7 Home Premium SP1 x32 Bits (SO donde trabajamos) x64DBG (Feb 14 2018) Detect It Easy (DIE) v1.01 DESCARGAR HERRAMIENTAS DESCARGAR TUTO+ARCHIVOS
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.

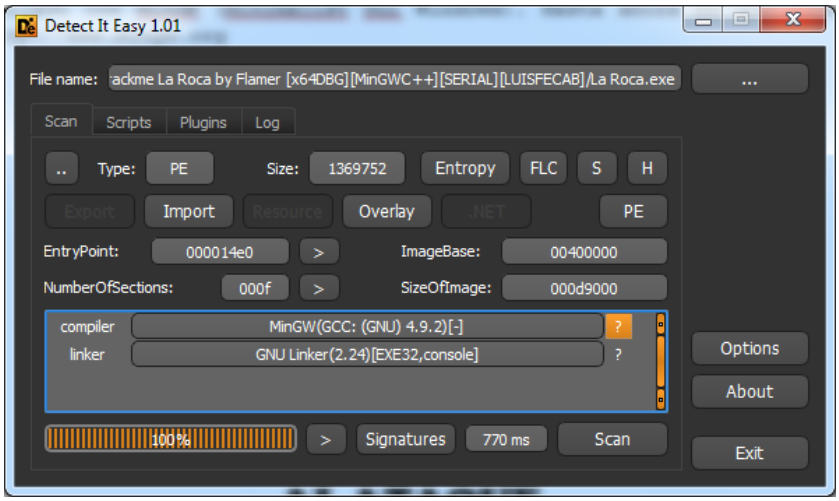
Hoy, Mayo 8, empezaré a escribir el tutorial del **Crackme** <**La.Roca**> hecho por **Flamer**. Este **Crackme** lo había resuelto ya hace unos meses atrás pero no había hecho el tutorial porque estaba obsesionado con poder completar el <**RandomWeird-Single.by.nextco**> y si leyeron mi tuto anterior 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 nuestra nuestras habilidades cuando se trata de buscar o hallar seriales. Este **Crackme** nos permitirá desarrollar nuestro "***ojo de cracker***".

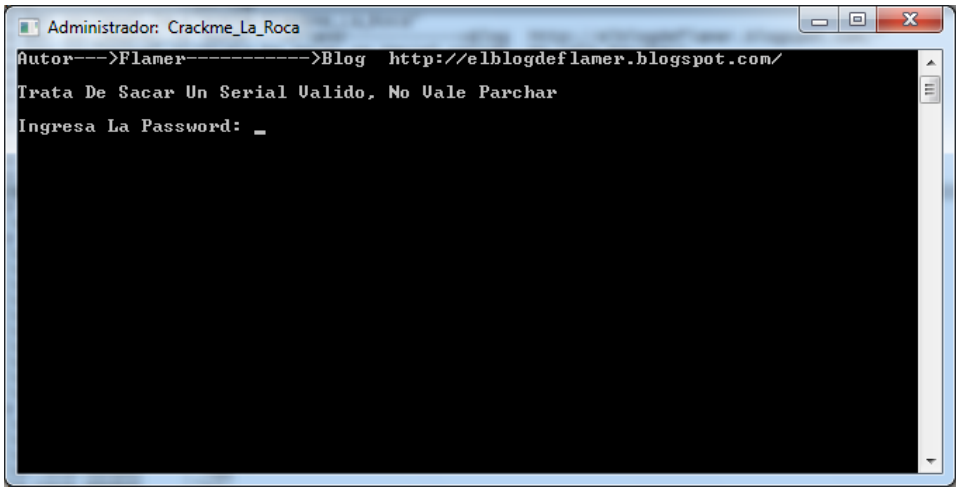
Para terminar esta pequeña introducción, quiero como siempre saludar a la lista de **CracksLatinoS** y por supuesto a mis amigos de **PeruCrackerS**.

ANALISIS INICAL

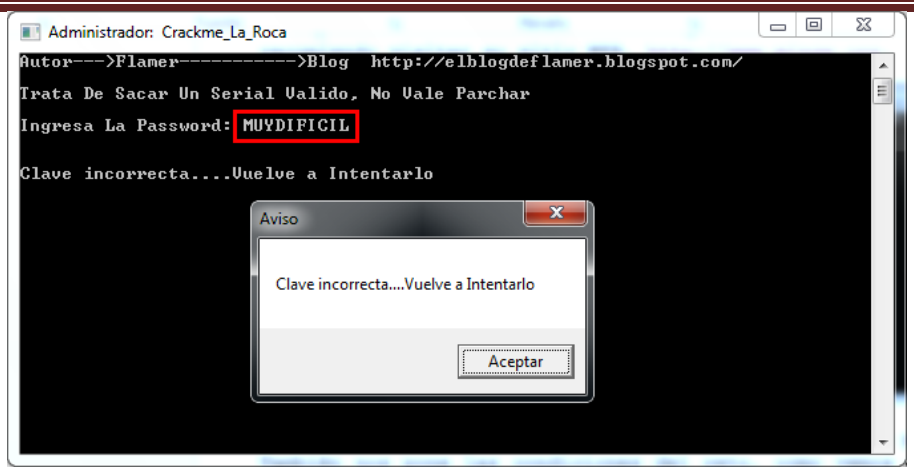
Para empezar lo analizaremos con el <**Detect It Easy (DIE) v1.01**> y si trae algo raro.



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**".

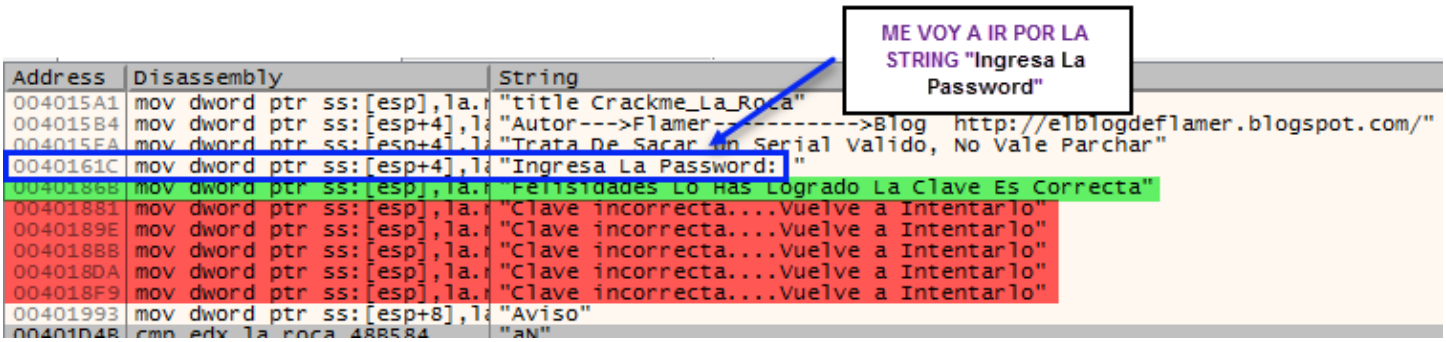


Era lo lógico, que nos diera a conocer al <**CHICO MALO**> que nos muestra un mensaje de "**Clave incorrecta…**". Y si vemos en la consola, también aparece "**Clave incorrecta....Vuelve a Intentarlo**". Eso nos indica que lo primero que podemos hacer es buscar las **Strings** y ver si por ahí podemos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

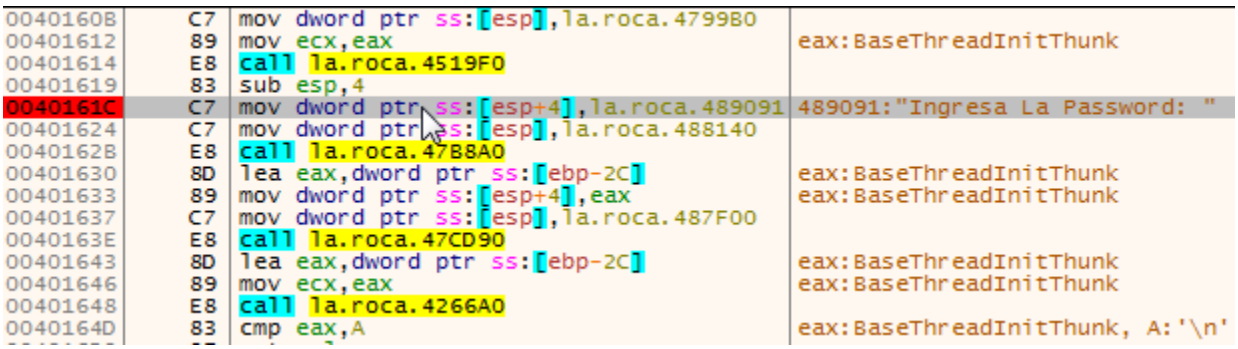
Ya con lo analizado podemos decir que no tiene nada raro y como vimos esta hecho en **C++** compilado con **MinGW**. Listo, y nos vamos al **AL ATAQUE**, que siempre es la mejor parte del tutorial para mi gusto.

AL ATAQUE

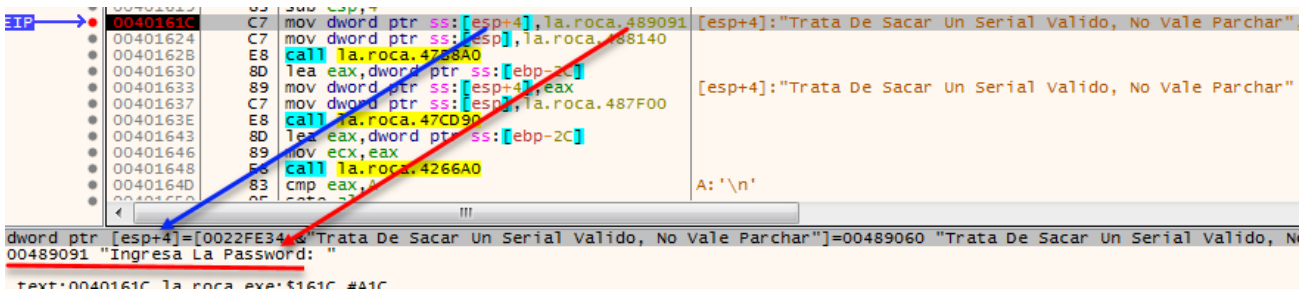
Carguemos el **Crackme** <La.Roca> en el **x64DBG** y buscamos en <**String references**> como ya lo habíamos planeado.



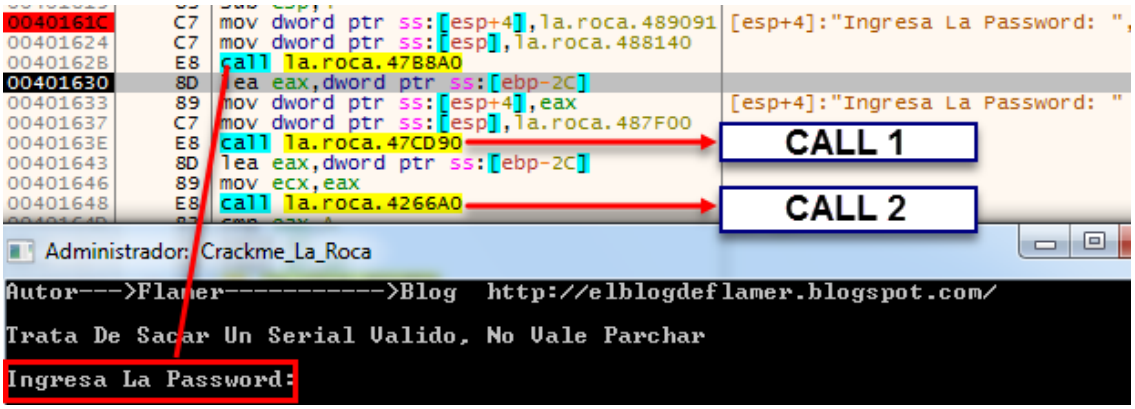
Por las **Strings** podemos ver que tenemos los mensajes para el <**CHICO BUENO**> y <**CHICO MALO**>. Si vemos se tienen cinco **Strings** para el <**CHICO MALO**> **RESALTADO EN ROJO**, eso puede indicar que pueden existir diferentes comprobaciones que nos pueden llevar por mal camino. Veremos más adelante cuáles son esas condiciones donde se decide si vamos bien o mal, por ahora decido irme por la **String "Ingresa La Password"** y así me permite quedar en el inicio del procedimiento y ver qué tengo a partir de ahí. Vayamos a la dirección **0040161C** que carga la **String**.



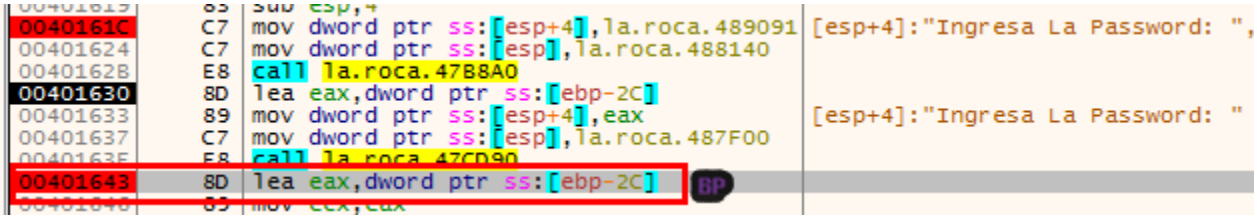
Ahí estamos en **0040161C** y le ponemos un <**BREAKPOINT**>. Analicemos primero de forma estática esta parte de código. Lo que pasará con este <**BREAKPOINT**>; nos detendremos cuando se valla a cargar la **String**, la cual será mostrada en consola, así que ahora que corramos el **Crackme** con <**F9**> este no se ejecutará por completo debido al <**BREAKPOINT**> pero gracias a eso podremos tracear y ver los siguientes **CALL** y darnos cuenta si sucede algo interesante. Dicho esto ejecutemos con <**F9**>.



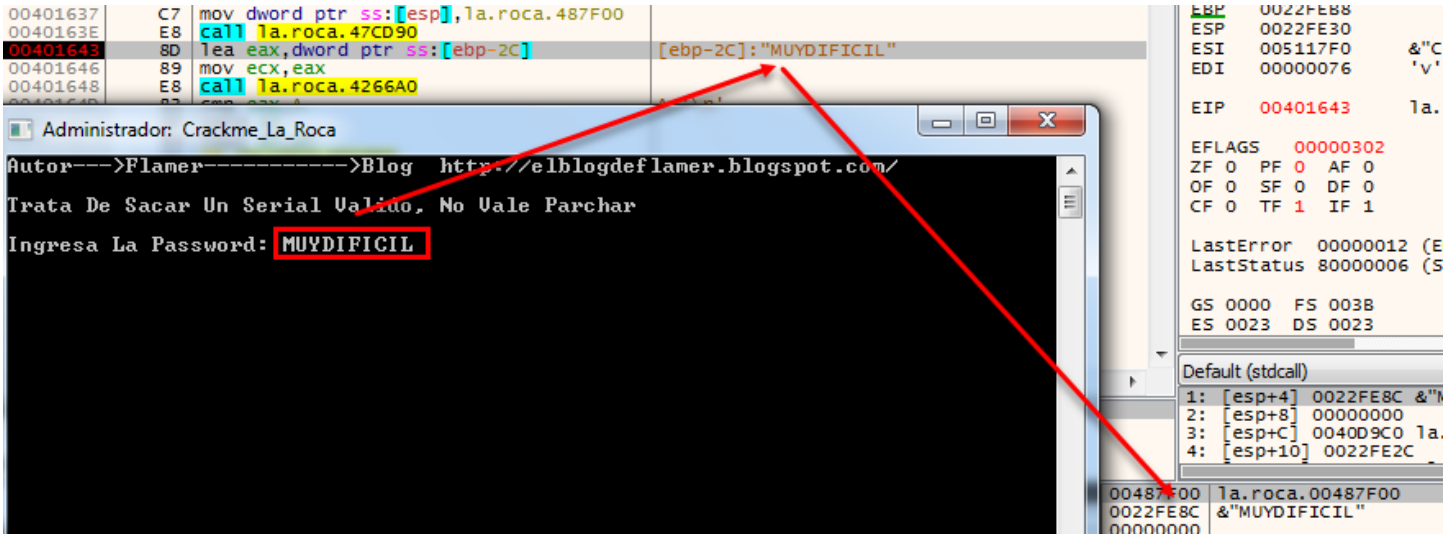
Nos detuvimos en **0040161C** y vemos que en [esp+4] tiene la **String "Trata De Sacar Un Serial Valido"**, no es la que inicialmente vimos, pero si vemos en la.roca.489091 tiene la **String "Ingresa La Password:"** que será cargada y que en **0040162B** **call** **la.roca.47B8A0** se cargara la **String**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



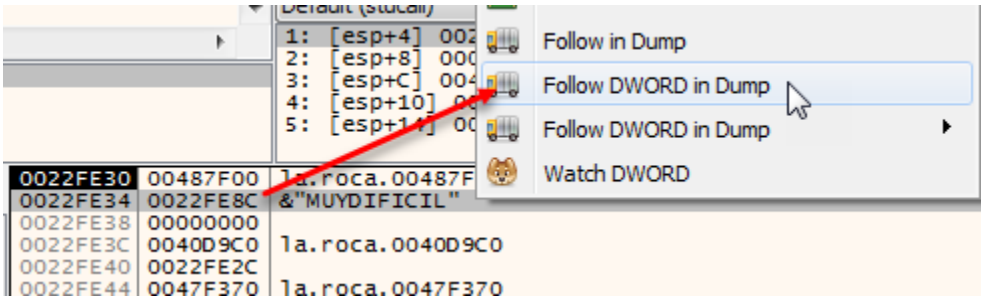
Estamos parados en **00401630**, hemos pasado ese **CALL** y podemos ver que ya en consola se cargó la **String "Ingresa La Password:"**. Nos quedan los otros **CALL** y que por lo que explicamos en el tutorial anterior, 1682 podemos decir que el **0040163E** **call** **la.roca.47CD90** se terminará 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**.



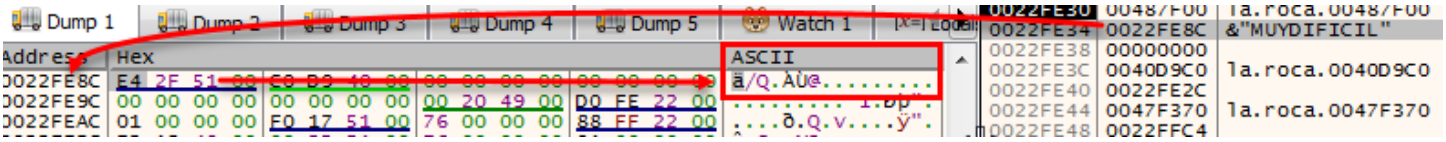
Ahí vemos puesto nuestro <**BREAKPOINT**>. Continuemos con la ejecución del programa hasta que nos detengamos en ese último <**BREAKPOINT**>. Para que no se pierdan, el **Crackme** se ejecuta completamente y nosotros debemos ingresar nuestro **SERIAL** y probarlo para llegar al <**BREAKPOINT**>, mi **SERIAL**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



Nos detuvimos donde queríamos después del call la.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**.

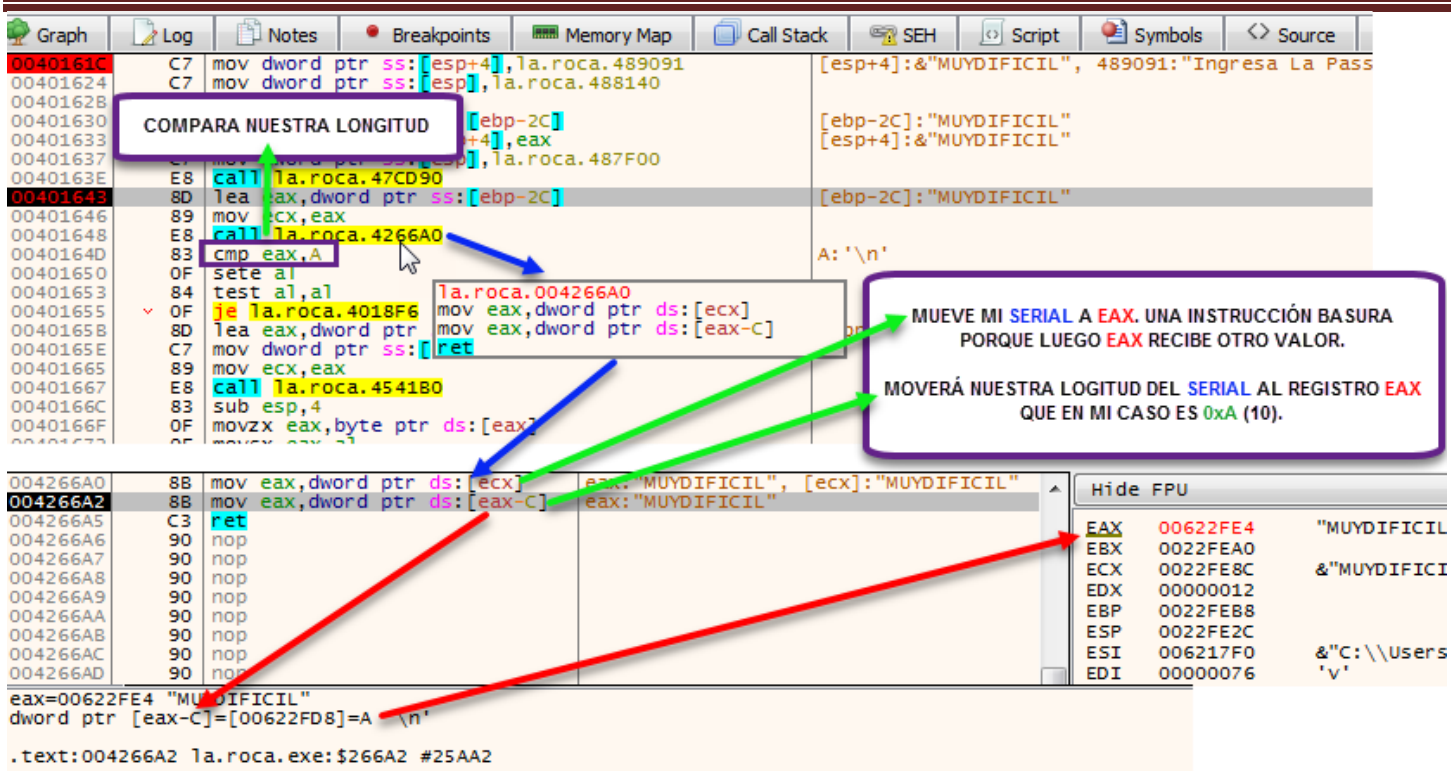


Con <**Clic Derecho->Follow DWORD in Dump**>. Veamos qué nos muestra el **DUMP**.

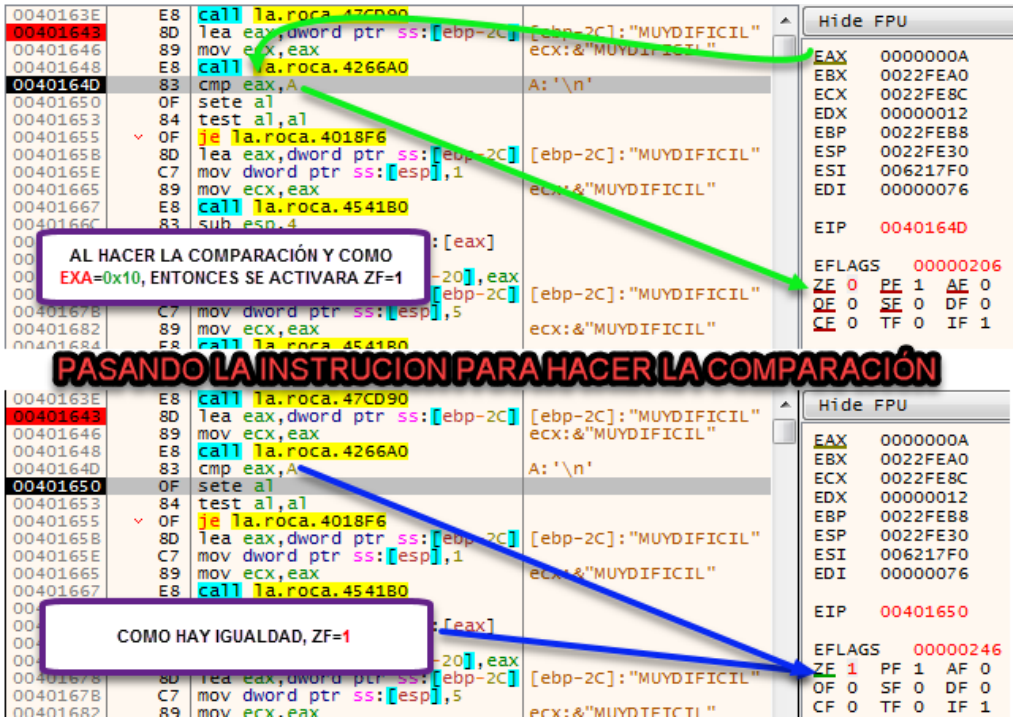


No esta nuestro **SERIAL**, en la columna **ASCII** no hay nada parecido a "**MUYDIFICIL**", pero si en la **PILA** si aparece y entonces como lo entendemos. Bueno, bueno, ya mis avezados **Crackers** sabrán la respuesta, bueno en realidad no tengo una respuesta para eso, solo puedo compartir lo que 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 abandono ya casi un mes porque me enfrasqué 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 la.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.

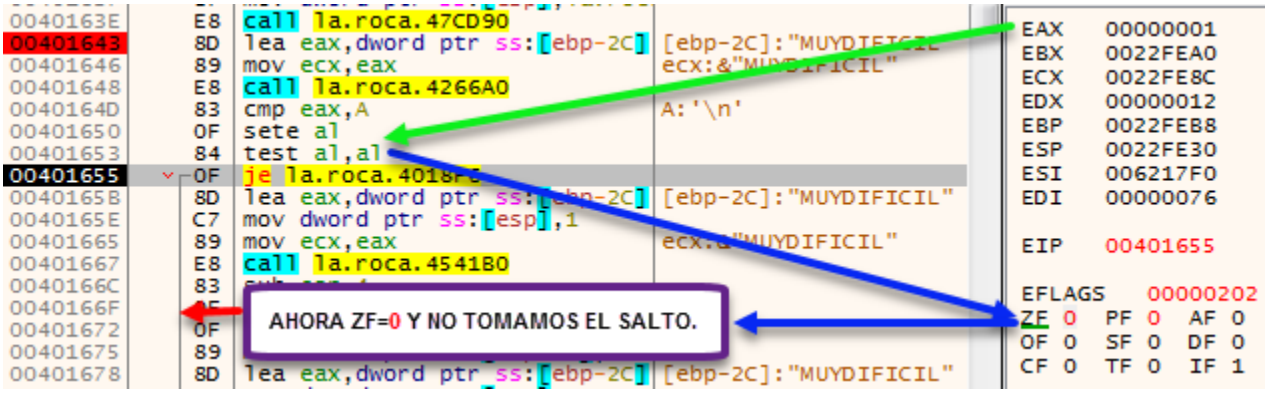


Bien, "***una imagen vale que mil palabras***". Sigan las flechas en la captura de arriba que resume lo que hace ese **CALL**, que en realidad no es nada solo cargar nuestra longitud. Observemos las siguientes instrucciones en donde compara nuestra longitud y después tomará o no un salto.

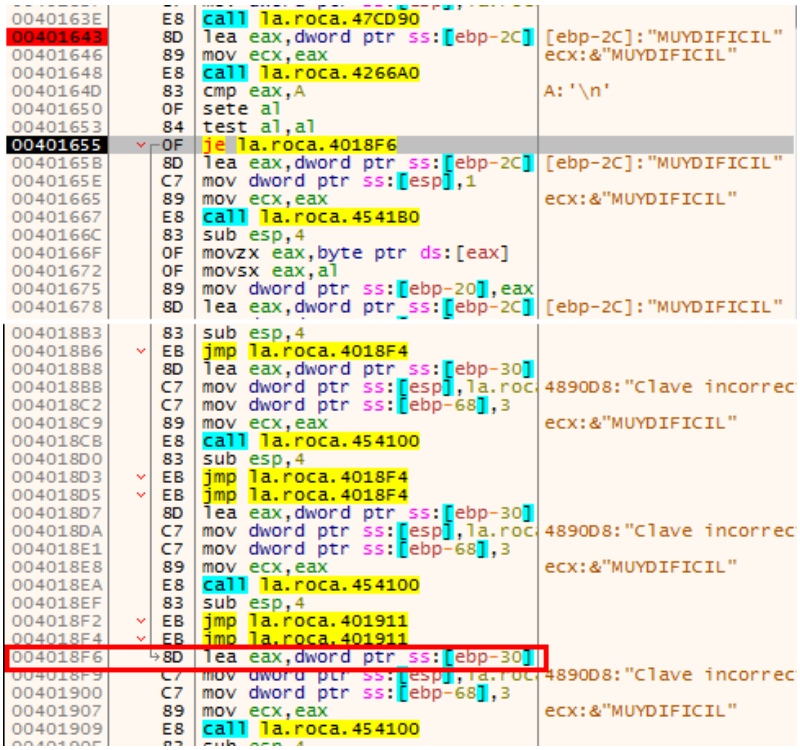


Como vemos se nos activó la **FLAG-Z=1**, y estamos parados en **00401650 sete al**, la instrucción **SETE** lo que hace cargar el valor que tiene **FLAG-Z** a un registro y para este caso es **EAX** pero en la posición **AL** que es el último **BYTE** de **EAX**. Si hemos

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.



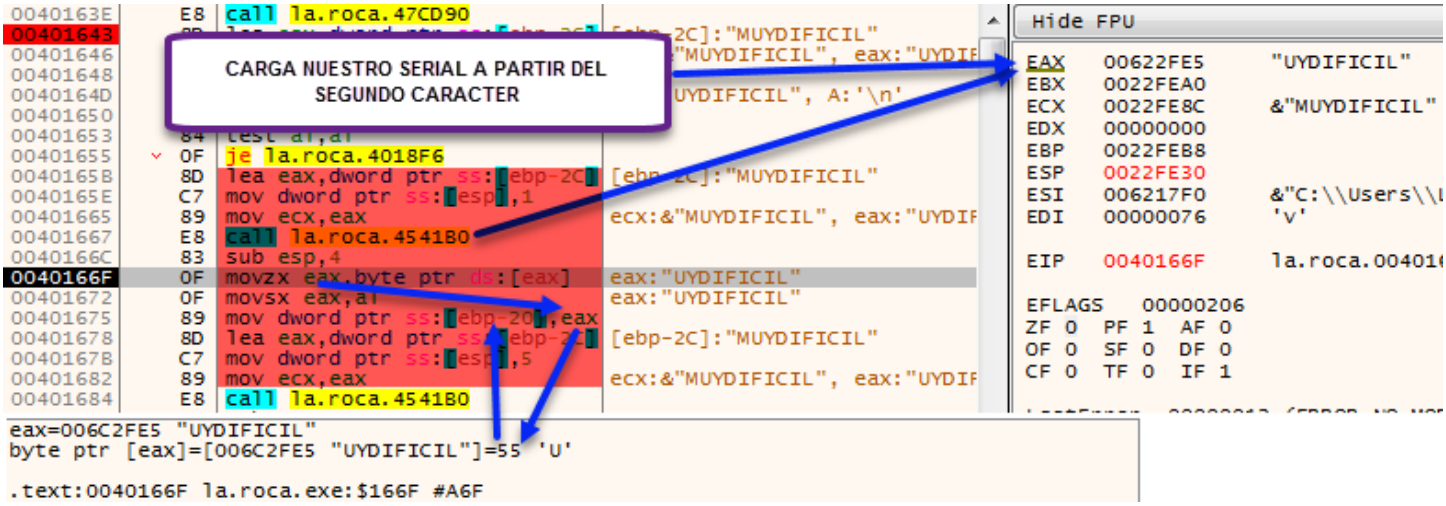
Pasando la instrucción **TEST** que activa nuestra **FLAG-Z=0**, y con eso no tomamos el salto **00401655** **je la.roca.4018F6** que nos enviará al <**CHICO MALO**>. Sigamos el salto para corroborar lo que dijimos.



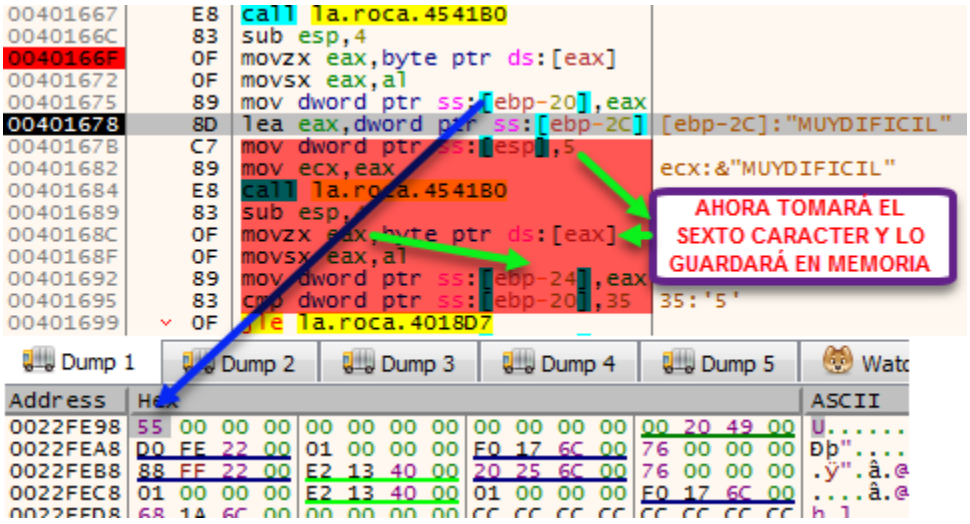
Podemos observar que hay varios <**CHICOS MALOS**> y como lo dijimos al inicio de esta sección cuando buscamos las **Strings** supusimos que debían de haber varias comprobaciones y con este lo podemos comprobar. Me imagino que ya se dieron cuenta que mi **SERIAL** le acertó a la longitud, primera vez 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

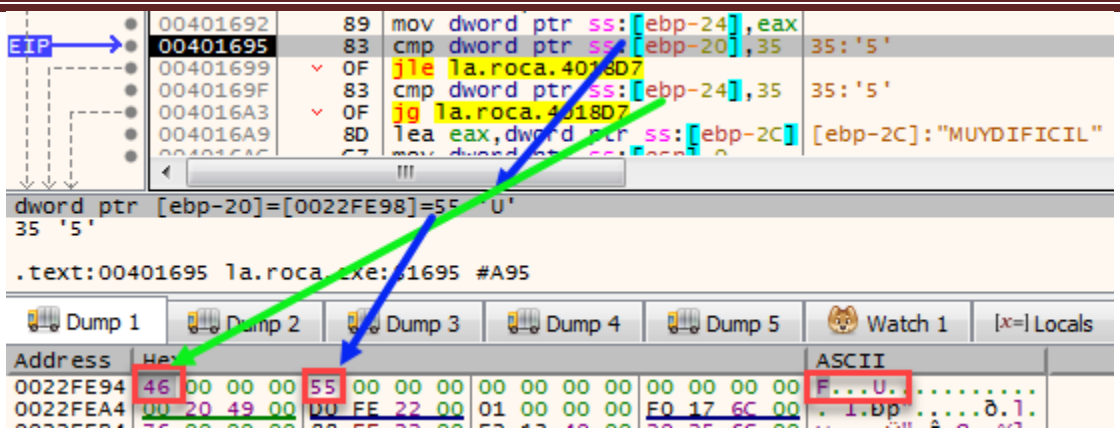
parte de nuestro **SERIAL** para luego trabajar con él. Lleguemos hasta la dirección **0040166F**.



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.



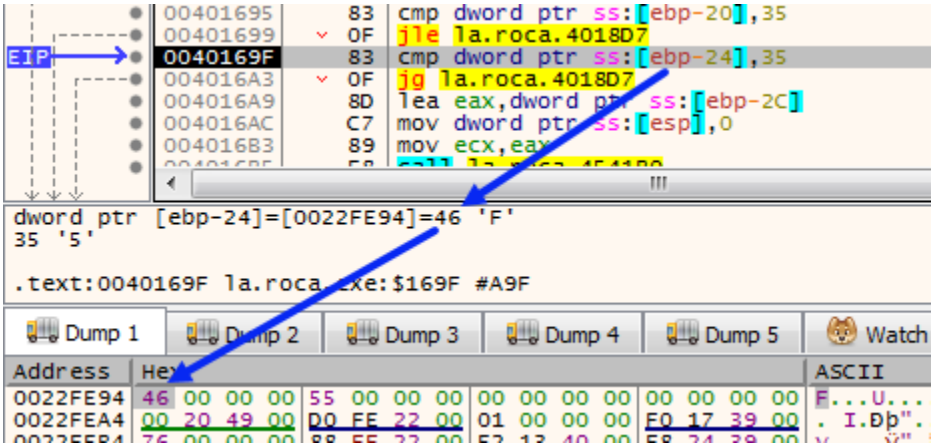
Estamos detenidos en **00101678** y podemos ver en el **DUMP** que en **0022FE98** guardó **0x55**, "**U**". Miremos lo **RESOLTADO EN ROJO**, vemos que ahora volverá a hacer el mismo procedimiento pero para tomar el Sexto carácter de nuestro "**MUYDIFICIL**", "**F**" y guardarlo en memoria. Entonces,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



Estando detenidos en **00401695**, vemos que va 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 **JNG**	Jump if less or equal Jump if not greater	signed	ZF = 1 or SF <> OF

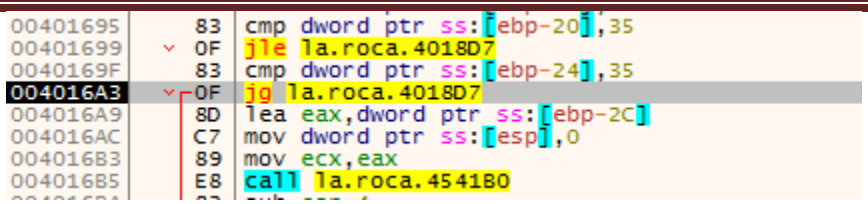
La idea es no tomar los saltos porque todos nos llevan al <**CHICO MALO**>. Si observamos **JLE** no salta si el valor comparado es mayor, y para nuestro caso, 0x55 **>** 0x35, entonces cumplimos la condición y no saltaremos, con eso podemos decir que el segundo carácter, "**U**", es correcto en el **SERIAL**. 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.



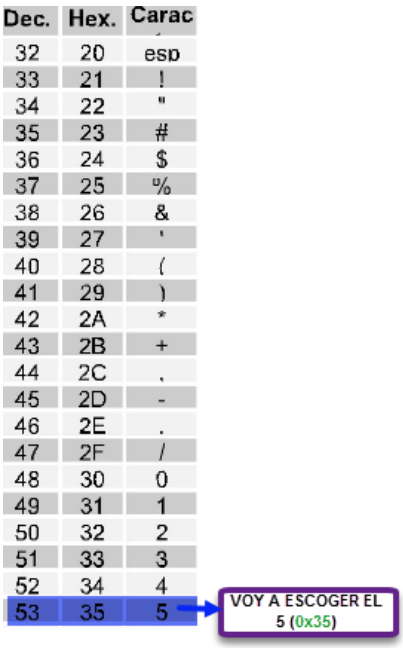
Aquí hay otro salto, y es un **JG**. Miremos qué condición activa este salto.

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

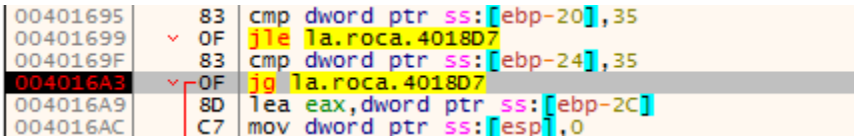
Se activa cuando nuestro valor a comparar es mayor. Hasta aquí me llegó la suerte con nuestro **SERIAL** porque el valor **HEXA** de "**F**" es mayor, miremos, 0x46 **>** 0x35. Con eso se activará el salto y vamos al <**CHICO MALO**>.



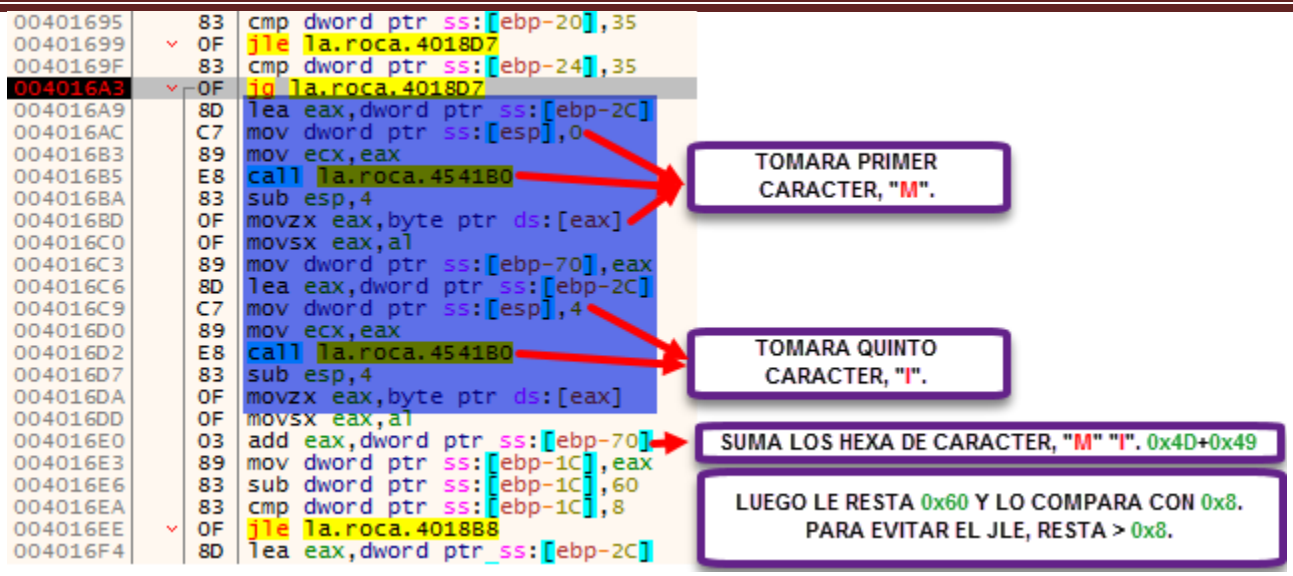
Clarito se ve que tomamos el salto. Entonces, debemos escoger un carácter que tenga un valor **HEXA** menor o igual a 0x35.



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.



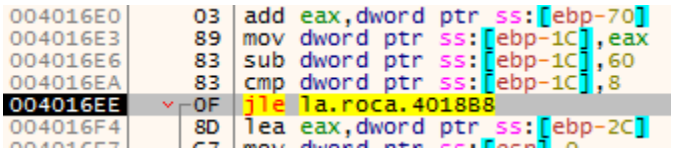
Quitemos los <**BREAKPOINT**> que habíamos puesto al inicio para evitar parar en lugares que ya no nos interesan. Reiniciemos todo e ingresamos nuestro nuevo **SERIAL** y lo probamos.



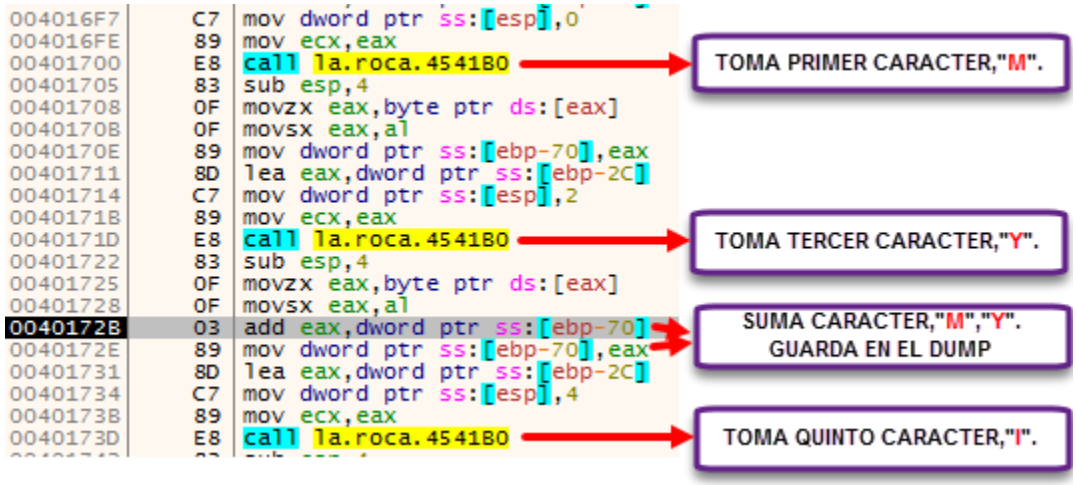
Perfecto, el salto no se toma y seguimos por buen camino. Miremos lo **RESALTADO EN AZUL**, vamos a tomar el Primer y Quinto carácter de "**MUYDI5ICIL**", "**M**", "**I**". Los sumará, **0x4D+0x49=0x69** y a esa suma le restará **0x60**, **0x96-60=0x36**. Recordemos la condición que activa el salto **JLE**.

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

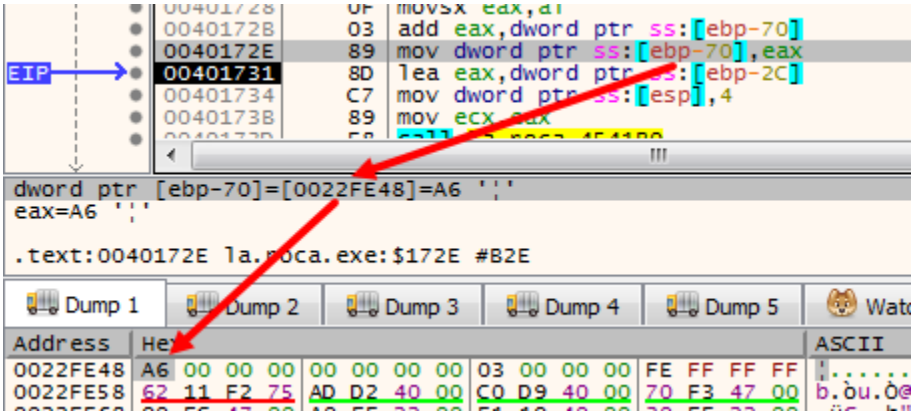
El resultado debe ser mayor para no saltar, **0x36 > 0x8**. Perfecto, no saltaremos y seguimos por buen camino. Nuestro **SERIAL** va quedando así, "**MUYDI5ICIL**".



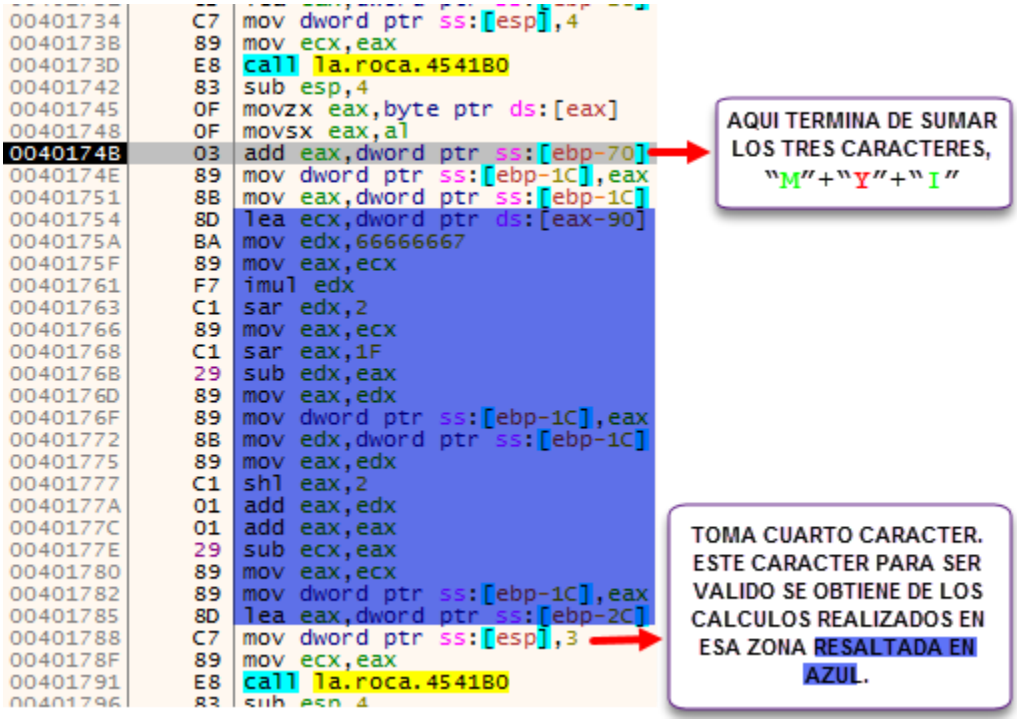
De lo anterior deducimos que la suma del Primer y Quinto carácter debe ser mayor o igual a **0x69**. Ahora se tomarán otros caracteres y veremos qué hace con ellos.



Sigue juagando 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. Traciemos hasta cuando guarda el valor sumado.

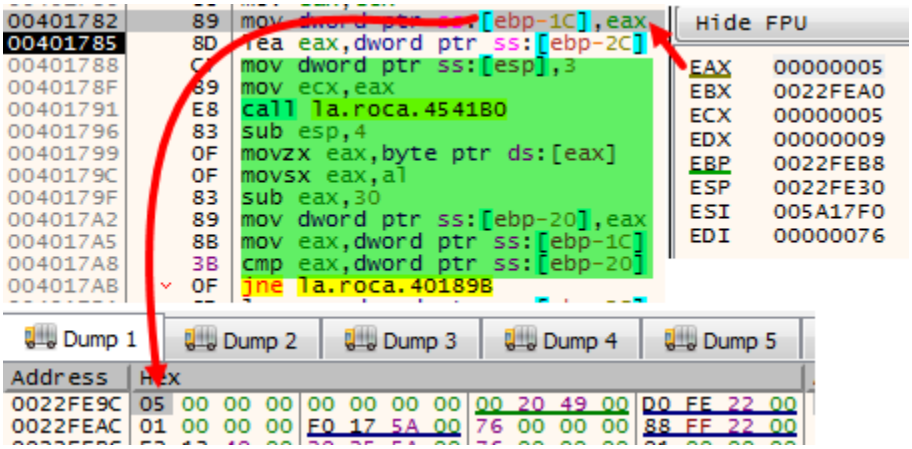


Ahí lo guardó, seguiremos traceando con los ojos bien abiertos para ver qué hace con ese valor y con el Quinto carácter.

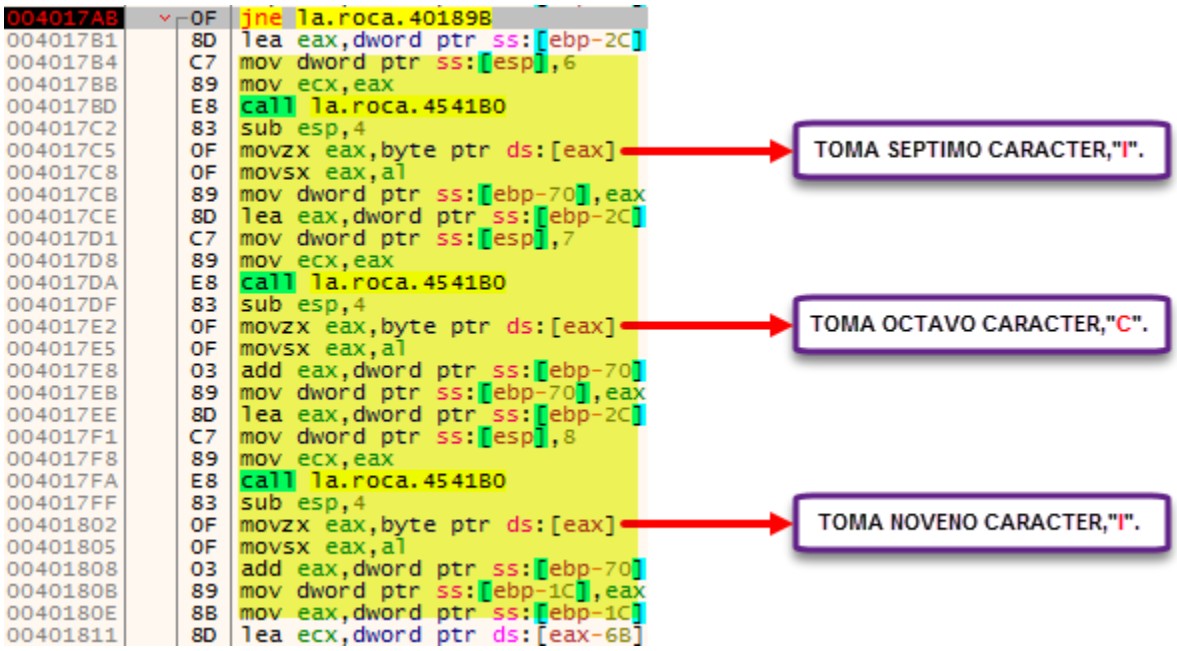


Pues lo que hace es sumarle a **0xA6** el valor **HEXA** de nuestro Quinto carácter, **0x49**, y guardarlo en memoria, **0xA6+0x49=0xEF**. En otras palabras, "**M**"+"**Y**"+"**I**". Se puso marullero el reto, observemos la zona **RESALTADA EN AZUL**. Ahora tenemos una serie de cálculos que toma como 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 valido 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

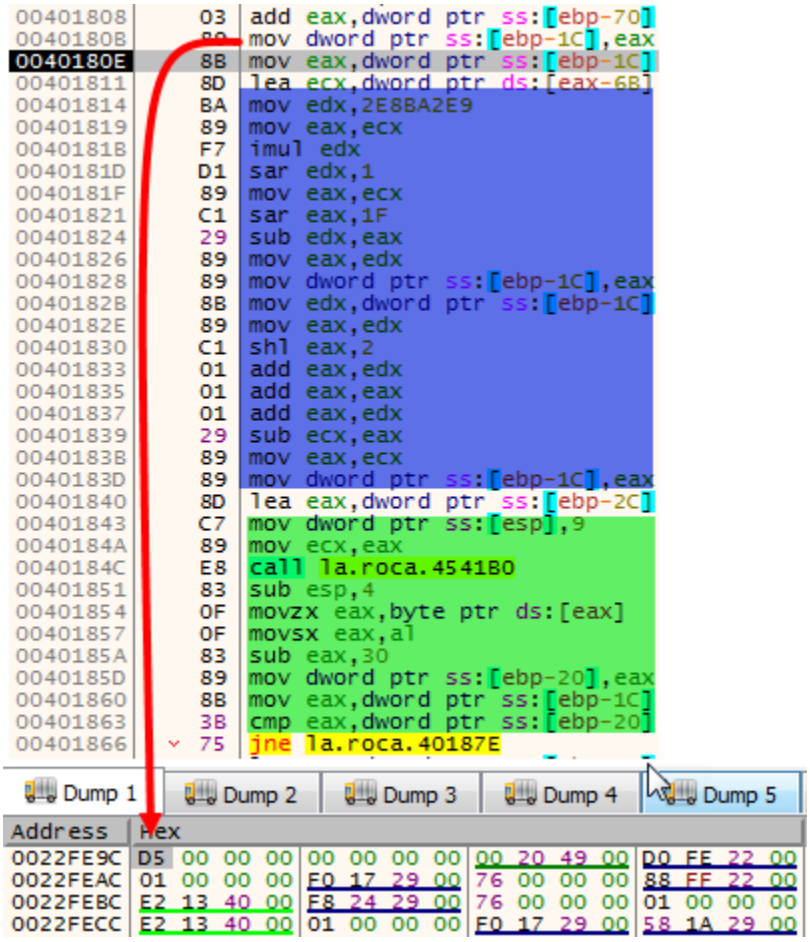
poco más abajo. La zona **RESALTADA EN AZUL** es importante si deseas hacer un **KeyGen**, que en este mismo instante que escribo estas palabras no creo que lo vaya a hacer, es que me está ganando la pereza, además **Flamer** nos pide hallar un **SERIAL VÁLIDO**. Vamos a tracear hasta donde guardemos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



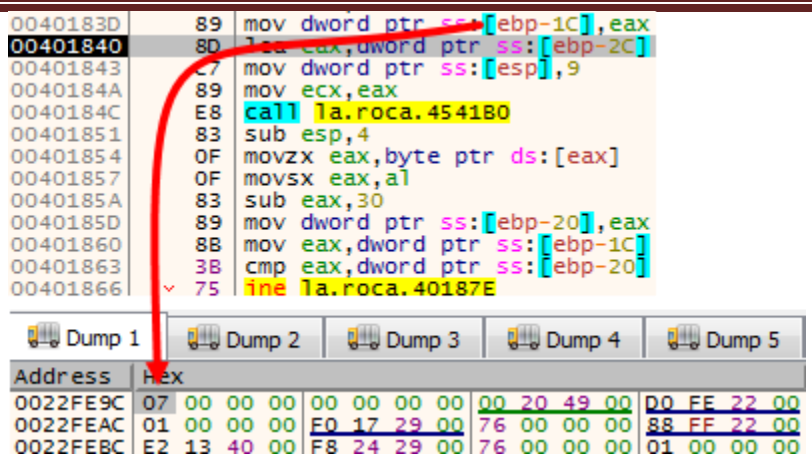
El valor que obtenemos es 0x05. Luego seguimos con la zona **RESALTADA EN VERDE**, ya sabemos que toma el Cuarto carácter de "**MUYDI5ICIL**", "**D**". Lo que hará es restarle 0x30 al valor **HEXA** de "**D**", 0x44-0x30=0x14. Lo que hay es una comparación entre el valor obtenido de la zona **RESALTADA 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.



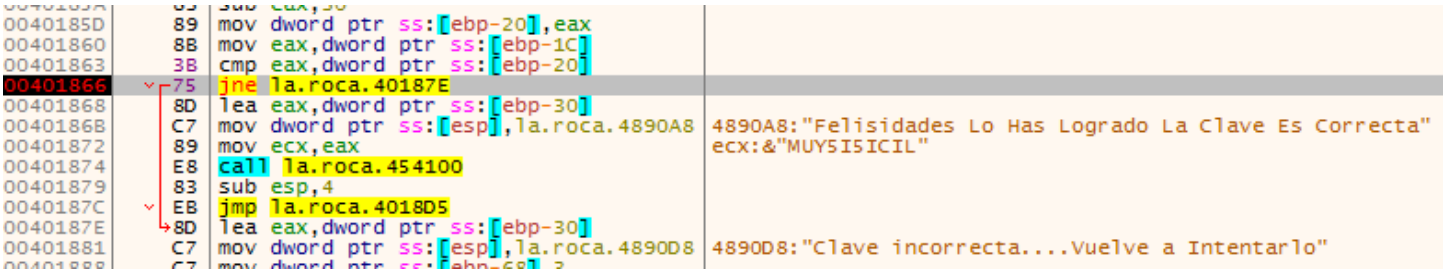
Estamos parados en **004017AB** y como vemos no tomamos el salto, así que vamos por buen camino. En la zona **RESALTADA EN AMARILLO**, prácticamente se va a realizar lo mismo que hicimos hace rato, solo que ahora se tomarán el Séptimo, Octavo y Noveno carácter de "**MUY5I5ICIL**", y los sumará, "**I**"+"**C**"+"**I**".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



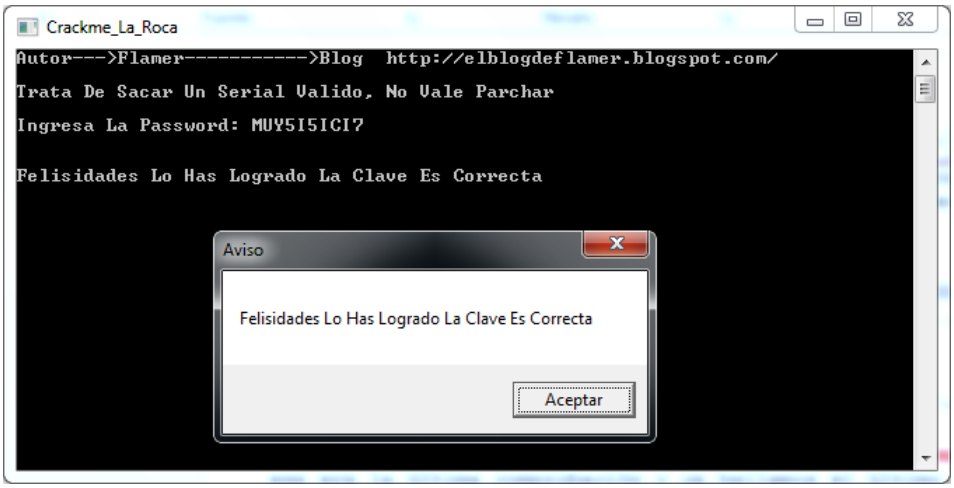
Ahí está, como podemos ver en la imagen de arriba el valor que obtuvimos de sumar los tres caracteres es **0xD5**, que se guarda en memoria, y en nuestra nueva zona **RESALTADA EN AZUL** se obtendrá un nuevo valor que nos servirá como pista para hallar el Décimo carácter válido. En la zona **RESALTADA 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**.



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 Decimo 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**".



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