



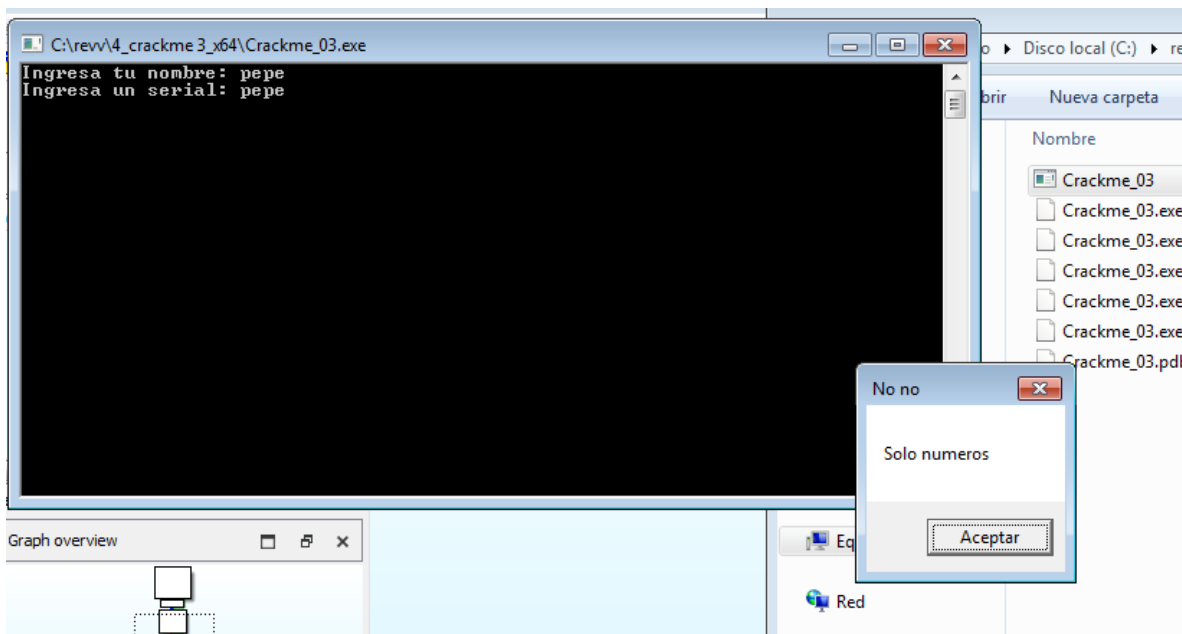
Tute – Crackme 3_64 – KarasuR00T – Fecha 12/08/2024

Este mini tutorial intenta, de alguna forma, explicar como llegar a resolver el crackme. “crackme 3_64”

¿Cómo es?

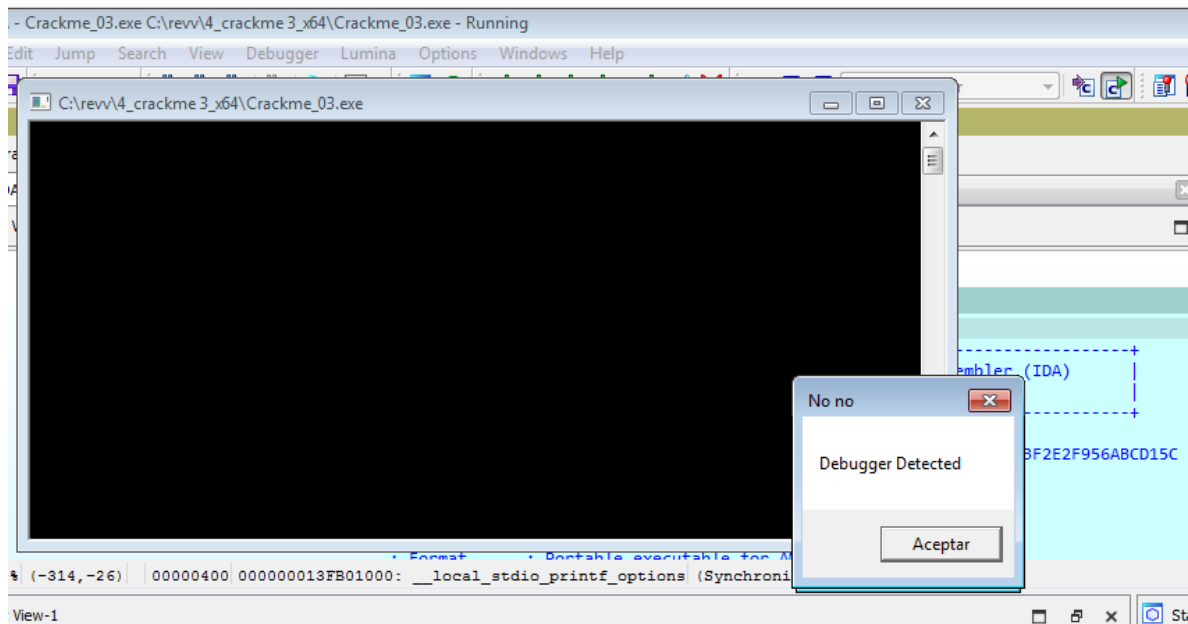
Bueno al iniciarlo ya sea haciendo doble click o por consola de comandos, se abre una ventana la cual solicita al usuario un nombre, luego solicita un serial.

Se verifica que, solo se pueden ingresar numeros como serial, ya que caso contrario indica un cartel “Solo Números”.



Comencé utilizando x64dbg – el ejecutable es de 64 bits

Si intento debuggearlo con x64dbg o bien con el debugger integrado en IDA recibo este mensaje:



Aunque aparece ese mensaje – en principio deja seguir operando.

Aun así, comencare en el IDA a realizar comentarios en el diagrama para darme una idea de lo que hace

```

; int __fastcall main(int argc, const char **argv, const char **envp)
main proc near

resultado= dword ptr -68h
x= dword ptr -64h
largo= dword ptr -60h
serial2= dword ptr -5Ch
debu= dword ptr -58h
cons= dword ptr -54h
serial= byte ptr -50h
nom= byte ptr -38h
var_18= qword ptr -18h

; __unwind { // __GSHandlerCheck
sub     rsp, 88h
mov     rax, cs:__security_cookie
xor     rax, rsp
mov     [rsp+88h+var_18], rax
mov     [rsp+88h+resultado], 0 ; Inicializa en 0 la variable resultado
mov     [rsp+88h+cons], 65 ; La constante vale 41h --> es el caracter A en hexa
call    cs:__imp_IsDebuggerPresent ; Llama a la funcion para detectar si hay un debugger o no
mov     [rsp+88h+debu], eax
cmp     [rsp+88h+debu], 1 ; Compara el resultado de debu - variable que es usada en la llamada para ver
jnz     short loc_7FF6CD9810B0 ; Salta SI NO ES 0

```

El detalle importante que noto en principio son dos cosas:

1ero -> tiene una llamada a una función para verificar presencia de un debugger –

2do -> s Se copia a la variable “cons” el valor 41h (carácter A en hexa)


```

xor     r9d, r9d      ; uType
lea     r8, Caption    ; "No no"
lea     rdx, Text      ; "Debugger Detected"
xor     ecx, ecx      ; hWnd
call    cs:__imp_MessageBoxA ; Cartel Debugger Detectado

```

```

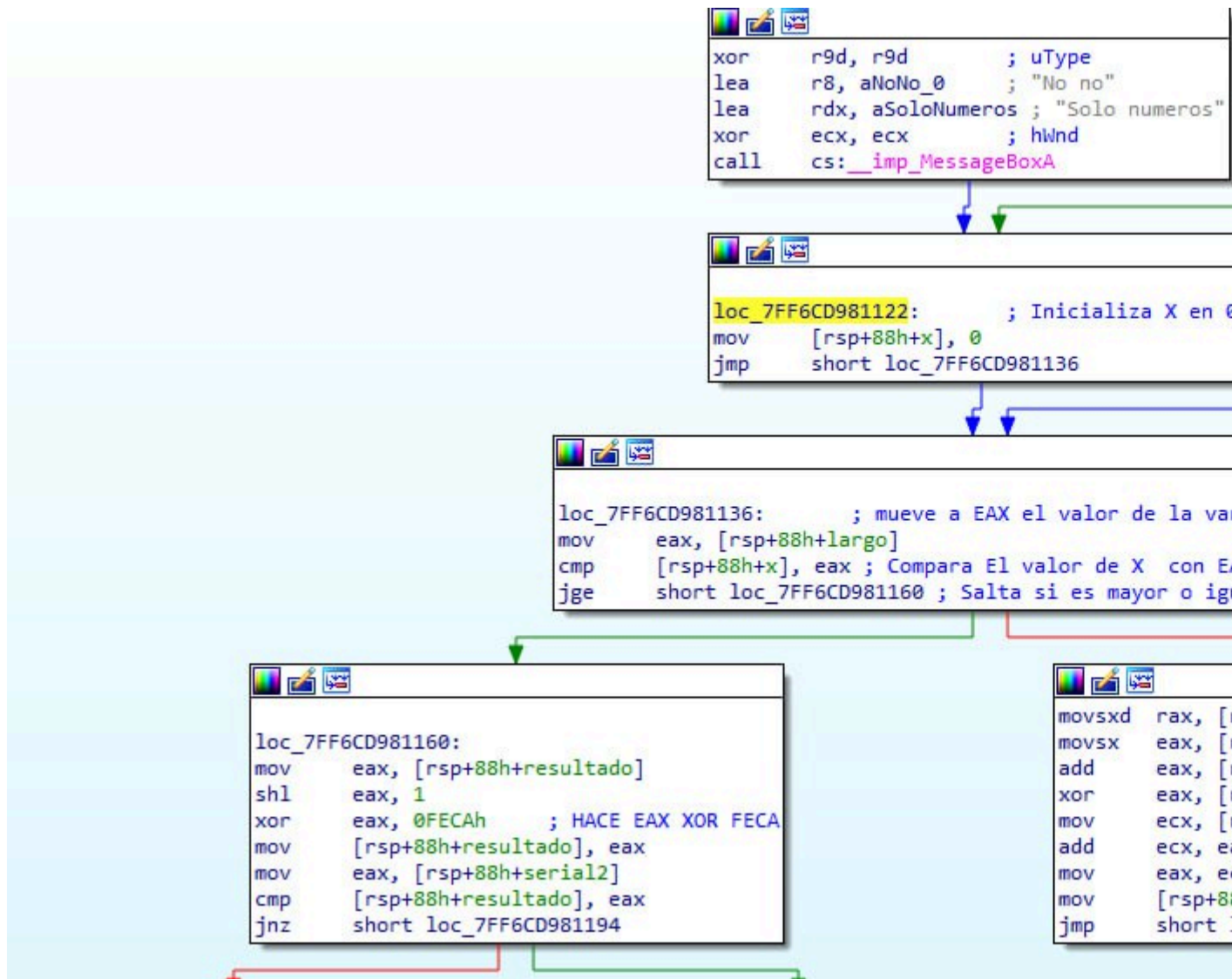
loc_7FF6CD9810B0:
lea     rcx, _Format    ; "Ingresa tu nombre: "
call    printf
mov     edx, 1Eh        ; Define el tamaño de caracteres para el nombre- 30 max
lea     rcx, [rsp+88h+nom] ; Resultado - guarda lo ingresado en la variable nom y se
call    gets_s          ; Almacena en el buffer (nombre)
lea     rcx, [rsp+88h+nom] ; ??
call    strlen          ; Determina el largo del string (nombre)
mov     [rsp+28h], eax   ; Guarda el largo determinado en EAX
lea     rcx, aIngresaUnSeria ; "Ingresa un serial: "
call    printf
mov     edx, 14h        ; Define el tamaño de caracteres para el nombre- 20 max
lea     rcx, [rsp+88h+serial] ; Resultado - guarda lo ingresado en la variable serial
call    gets_s          ; Almacena en el buffer (serial)
lea     rcx, [rsp+88h+serial] ; string
call    atoi            ; funcion para convertir una cadena en entero.
mov     [rsp+88h+serial2], eax ; Copia el valor de EAX en la variable Serial2
cmp     [rsp+88h+serial2], 0 ; Compara contra 0 -> si no es 0 es por que se introdujo o
jnz     short loc_7FF6CD981122 ; Salta SI NO ES 0

```

En el siguiente bloque básicamente toma por teclado (sea el nombre y el serial) y los aloja en un buffer cada uno (definiendo el tamaño 30 y 20 respectivamente) – luego los convierte de string a entero con la función ATOI

Notar que el serial ingresado por nosotros lo guarda en el registro RCX – también aquí compara y se da cuenta si el serial es vacío o no.

También determina el largo del nombre (por ejemplo pepe posee 4 caracteres)



Luego -> inicializa la variable x en 0 e ingresa -> luego mueve a EAX el valor correspondiente al largo del nombre (siguiendo con el ejemplo de usuario pepe EAX = 00000004)

Compara entre la variable x y EAX (4) como no es mayor igual entra en el bucle y aquí viene el “truco”

Primero va a tomar letra por letra del string ingresado (tener en cuenta que lo va a manejar como HEXA) ➔ Esto quiere decir que: pepe – en HEXA seria: 70 65 70 65

Entonces lo que va a hacer es a cada letra sumarle 4 (por lo anterior dicho ya que es el largo del string) a ese resultado le aplica XOR 41 (que es la variable cons)

Luego lo va a guardar en la variable “resultado” -> Lo copia a ECX

Y luego sigue así hasta finalizar y seguir por el contador el cual incrementa X en +1 -> y vuelve al inicio, para comparar – así recorre el string y va realizando los cálculos correspondientes

Al finalizar el bucle salta a la parte donde dice loc_7FF6CD981160 – es decir a la parte izquierda –

Luego toma la variable resultado y la multiplica por 2 → esto se debe a la línea shl eax,1 → Luego aplica XOR FECA (que en decimal es 65226) y lo guarda

Luego compara entre el serial ingresado inicialmente por nosotros contra el serial que armo según las operaciones realizadas sobre el nombre ingresado – Si son = muestra el cartel de chico bueno “Muy Bien” – caso contrario muestra el cartel de chico malo “Serial Incorrecto” como se muestra en la siguiente imagen



Siendo si, repasemos entonces de una forma más sencilla lo que hace este crackme internamente

1ero: Nos pide un nombre y un serial X

Nombre: pepe

Entonces lo que hace es que saca el largo del string – como dijimos es 4

Luego convierte a pepe en sus valores hexadecimales correspondientes: 70 65 70 65

Luego letra por letra realiza las operaciones:

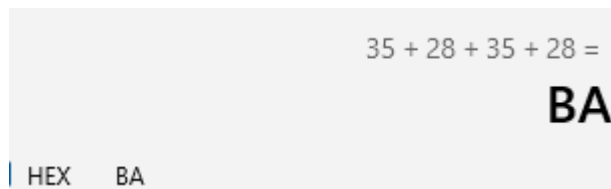
$$70 + 4 = 74 \rightarrow 74 \text{ XOR } 41 = 35$$

$$65 + 4 = 69 \text{ XOR } 41 = 28$$

$$70 + 4 = 74 \rightarrow 74 \text{ XOR } 41 = 35$$

$$65 + 4 = 69 \text{ XOR } 41 = 28$$

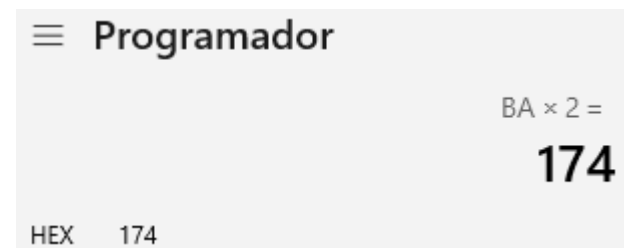
Si sumamos todos los resultados



35 + 28 + 35 + 28 =
BA
HEX BA

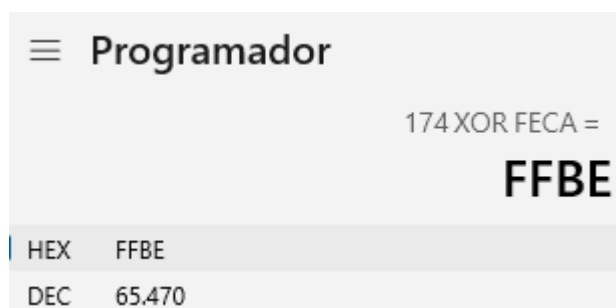
Entonces BA es lo que vale hasta ahora EAX para seguir realizándole operaciones ->

A “BA” se lo multiplica por 2 =



BA × 2 =
174
HEX 174

Luego se realiza 174 XOR FECA



174 XOR FECA =
FFBE
HEX FFBE
DEC 65,470

Es decir que nuestro serial para el nombre pepe es 65470 – Lo comprobamos:



```
Ingresa tu nombre: pepe  
Ingresa un serial: 65470
```

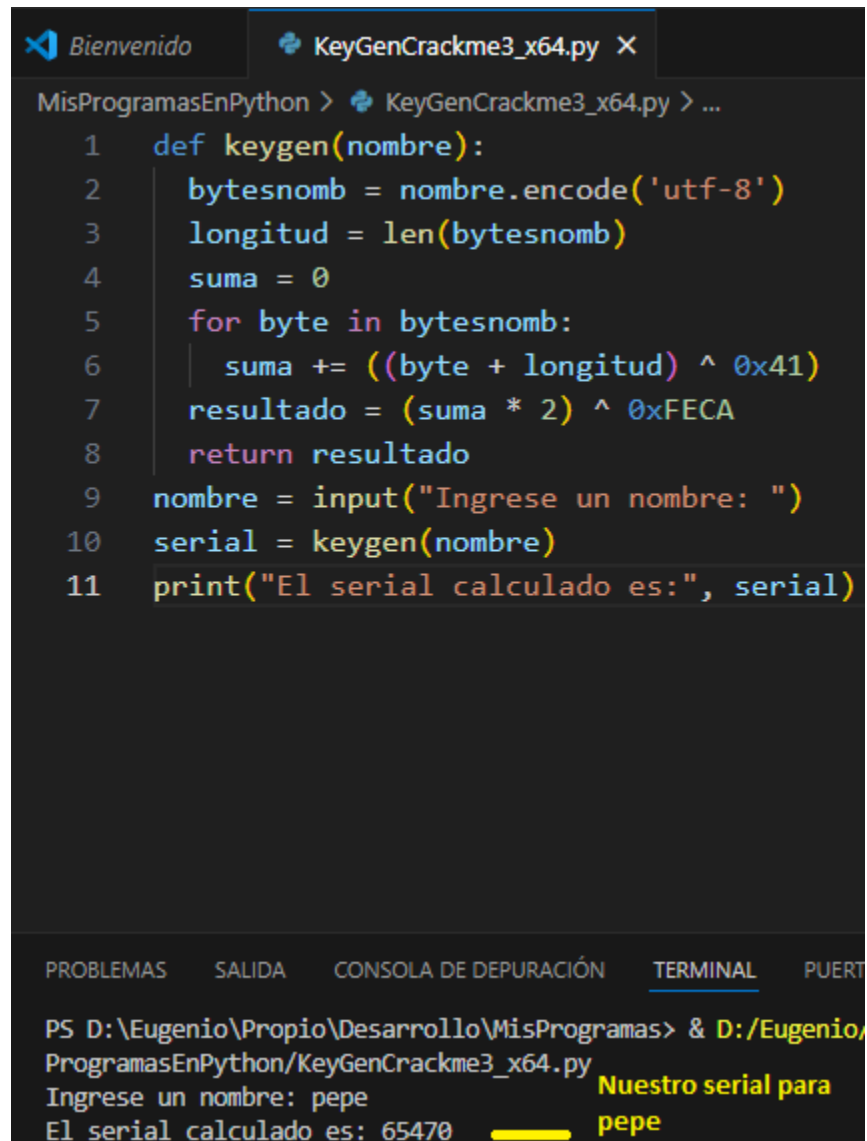
OK

Muy bien !

Aceptar

Genial – entonces con un poco de ayuda realizamos el script para poder usar cualquier nombre y nos dé el serial que corresponda:

El script esta realizado en Python:



```

MisProgramasEnPython > KeyGenCrackme3_x64.py X
1  def keygen(nombre):
2      bytesnomb = nombre.encode('utf-8')
3      longitud = len(bytesnomb)
4      suma = 0
5      for byte in bytesnomb:
6          suma += ((byte + longitud) ^ 0x41)
7      resultado = (suma * 2) ^ 0xFECA
8      return resultado
9  nombre = input("Ingrese un nombre: ")
10 serial = keygen(nombre)
11 print("El serial calculado es:", serial)

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```

PS D:\Eugenio\Propio\Desarrollo\MisProgramas> & D:/Eugenio/ProgramasEnPython/KeyGenCrackme3_x64.py
Ingrese un nombre: pepe
El serial calculado es: 65470

```

Nuestro serial para pepe

Es un script bastante sencillo el cual solicita al usuario el nombre que desea - y realiza todas las operaciones ya mencionadas devolviendo el serial para nuestro nombre indicado.