



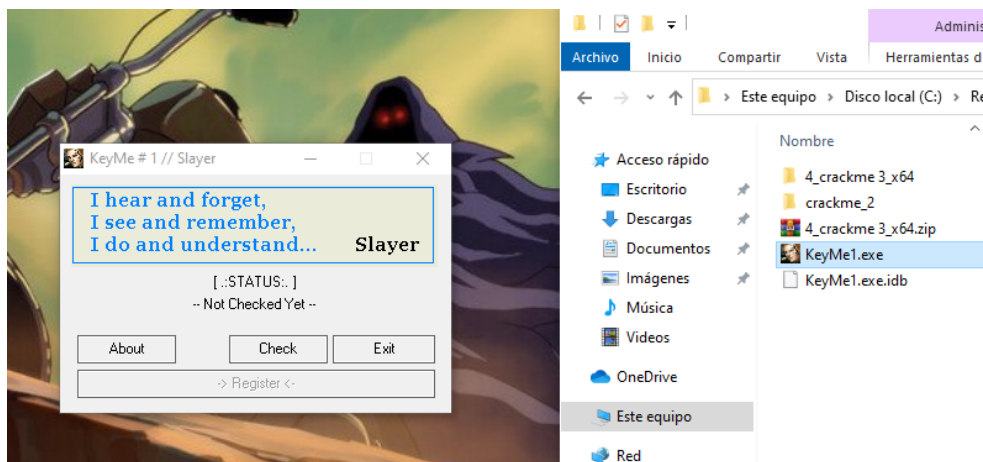
Crackme KeyMe #1 – Slayer

19/09/2024 – KarasuRØØT

¡Buenas nuevamente! Aquí intentando seguir aprendiendo algo de reversing. Bueno

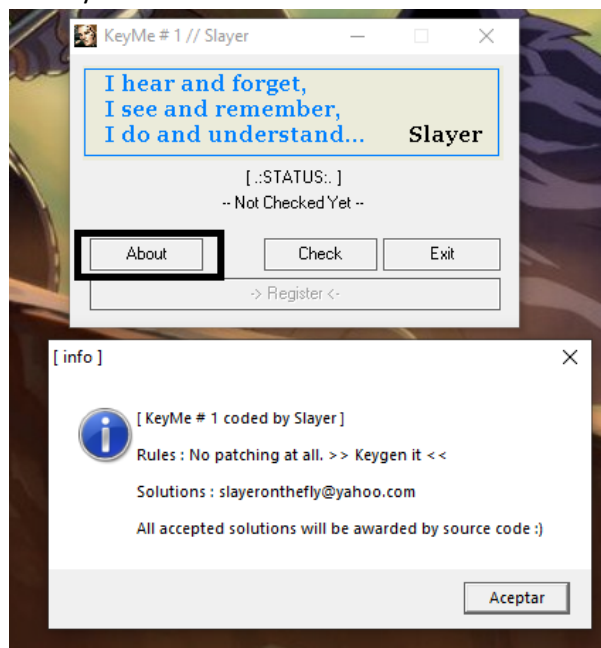
Comencemos a ver un poco el crackme “KeyMe #1 – Autor: Slayer

Veamos con que nos encontramos:



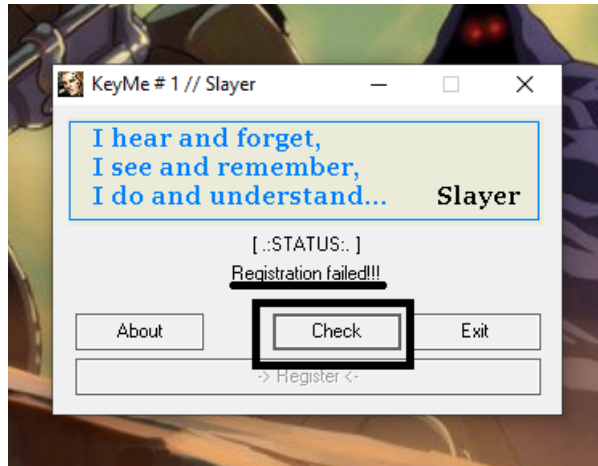
Ok al ejecutarlo se abre una ventana, con 3 botones disponibles (About, Check, Exit), y uno griseado (Register).

Si doy en “About” nos muestra esta información:



Básicamente, indica que no todo se resuelve parcheando – que generemos un Keygen y que manden la solución a la casilla de mail.

Si doy en el botón “Check”:



Ohh que sorpresa – nos dice que la registración falló – que bien... Y como era de esperarse, el botón Exit – sale del programa

Debo confesar que lo que indica el cartel. Me llamo la atención, en realidad una de mis esperanzas era que me de alguna pista.

“i hear and forget,

i see and remember,

i do and understand”

Traducido sería algo así:

“escucho y olvido,

veo y recuerdo,

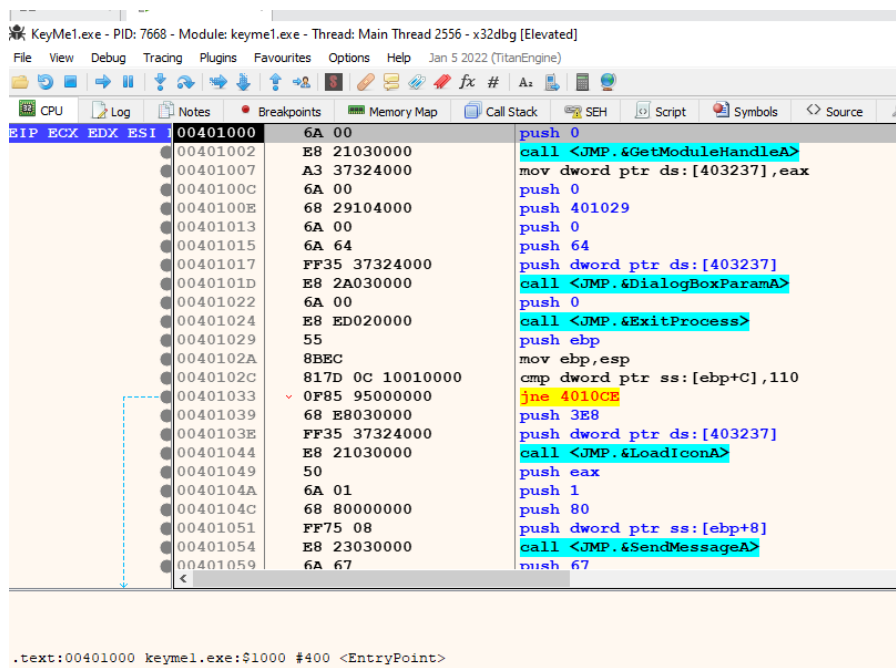
lo hago y lo entiendo”

Particularmente yo, no encontré nada. Por eso prosigo por otro lado.

Si lo abro en IDA para revisarlo... la verdad no se me ocurre ahora como verlo. Seguire con x64DBG –

```
;-----  
; Input SHA256 : A16C0548267208831031425A474A43ED5E43D744C40D681A2CFC362B24098299  
; Input MD5    : D3A282BF64B2932C2318096E38E040D7  
; Input CRC32  : 192CD00B  
  
; File Name    : C:\Reversing\HerramientasREVERSING\Ejercicios-SOLID\KeyMe1.exe  
; Format       : Portable executable for 80386 (PE)  
; Imagebase    : 400000  
; Timestamp    : 41F6DF8B (Wed Jan 26 00:08:43 2005)  
; Section 1. (virtual address 00001000)  
; Virtual size : 00000388 ( 904.)  
; Section size in file : 00000400 ( 1024.)  
; Offset to raw data for section: 00000400  
; Flags 60000020: Text Executable Readable  
; Alignment    : default  
  
.686p  
.mmx  
.model flat  
  
; Segment type: Pure code  
; Segment permissions: Read/Execute  
_text segment para public 'CODE' use32  
assume cs:_text  
;org 401000h  
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing  
  
; Attributes: noreturn  
  
public start  
start proc near  
push 0 ; lpModuleName  
call GetModuleHandleA  
mov hInstance, eax  
push 0 ; dwInitParam  
push offset DialogFunc ; lpDialogFunc  
push 0 ; hInstParent  
push 64h ; 'd' ; lpTemplateName  
push hInstance ; hInstance  
call DialogBoxParamA  
push 0 ; uExitCode  
call ExitProcess  
start endp
```

En principio podemos decir que ahí comienza el programita – el x64 sugiere que es el entrypoint:



```
KeyMe1.exe - PID: 7668 - Module: keyme1.exe - Thread: Main Thread 2556 - x32dbg [Elevated]  
File View Debug Tracing Plugins Favourites Options Help Jan 5 2022 (TitanEngine)  
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source  
EIP ECX EDX ESI 00401000 6A 00 push 0  
00401002 E8 21030000 call <JMP.&GetModuleHandleA>  
00401007 A3 37324000 mov dword ptr ds:[403237],eax  
0040100C 6A 00 push 0  
0040100E 68 29104000 push 401029  
00401013 6A 00 push 0  
00401015 6A 64 push 64  
00401017 FF35 37324000 push dword ptr ds:[403237]  
0040101D E8 2A030000 call <JMP.&DialogBoxParamA>  
00401022 6A 00 push 0  
00401024 E8 ED020000 call <JMP.&ExitProcess>  
00401029 55 push ebp  
0040102A 8BEC mov ebp,esp  
0040102C 817D 0C 10010000 cmp dword ptr ss:[ebp+C],110  
00401033 0F85 95000000 jne 4010CE  
00401039 68 E8030000 push 3E8  
0040103E FF35 37324000 push dword ptr ds:[403237]  
00401044 E8 21030000 call <JMP.&LoadIconA>  
00401049 50 push eax  
0040104A 6A 01 push 1  
0040104C 68 80000000 push 80  
00401051 FF75 08 push dword ptr ss:[ebp+8]  
00401054 E8 23030000 call <JMP.&SendMessageA>  
00401059 6A 67 push 67  
  
.text:00401000 keyme1.exe:$1000 #400 <EntryPoint>
```

Intento buscar que Sting tiene:

String
"[info]"
"reg.key"
"Memory allocation error!"
"Good work. You have done it!"
"Registration failed!!!"
"Step 1 ok -> now Register it!"
"Registration failed!!!"
L"t-ms-win-firewallapi-webproxy-11-1-1"
L"\\/"
L"win.ini"
L"Nation"
L"Control Panel\\International\\Geo"
L"\\Software\\Microsoft\\Windows NT\\CurrentVersion"
L"AppCompatFlags\\Layers"
L"\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders"
L"SIGN.MEDIA="

Chico BUENO

Chico MALO

Chico casi bueno (?)

¿?

Si voy al desensamblador veo y traceo, comienzo a ver que hay varias call que llaman a API's

Address	Disassembly	Comment
004011E0	EB 4C	jmp 40122E
004011E2	FF35 4A334000	push dword ptr ds:[40334A]
004011E8	E8 1D010000	call <JMP.&CloseHandle>
004011ED	56	push esi
004011EE	52	push edx
004011EF	8B35 4E334000	mov esi,dword ptr ds:[40334E]
004011F5	8B06	mov eax,dword ptr ds:[esi]
004011F7	83C6 04	add esi,4
004011FA	8B16	mov edx,dword ptr ds:[esi]
004011FC	33C2	xor eax,edx
004011FE	8B15 42334000	mov edx,dword ptr ds:[403342]
00401204	03C2	add eax,edx
00401206	5A	pop edx
00401207	5E	pop esi
00401208	3B05 3E334000	cmp eax,dword ptr ds:[40333E]
0040120E	75 1E	jne 40122E
00401210	68 D9304000	push keymel.4030D9
00401215	6A 66	push 66
00401217	FF75 08	push dword ptr ss:[ebp+8]
0040121A	E8 63010000	call <JMP.&SetDlgItemTextA>
0040121F	6A 00	push 0
00401221	FF35 5A334000	push dword ptr ds:[40335A]
00401227	E8 26010000	call <JMP.&EnableWindow>
0040122C	EB 29	jmp 401257
0040122E	68 F6304000	push keymel.4030F6
00401233	6A 66	push 66

esi:EntryPoint
edx:EntryPoint
esi:EntryPoint
esi:EntryPoint
esi:EntryPoint
edx:EntryPoint, esi:EntryPoint
edx:EntryPoint
edx:EntryPoint
edx:EntryPoint
edx:EntryPoint
esi:EntryPoint

4030D9:"Good work. You have done i

4030F6:"Registration failed!!!"

Hay varias API's ¹de interés, no pondré todas, pero sí creo yo las más importantes en este crackme -

Destination
<kernel32.GetModuleHandleA>
<user32.DialogBoxParamA> ←
<kernel32.ExitProcess>
<user32.LoadIconA>
<user32.SendMessageA>
<user32.GetDlgItem>
<user32.GetDlgItem>
<kernel32.GetComputerNameA> ←
<kernel32.lstrlen>
<user32.EndDialog>
<user32.EndDialog>
<user32.MessageBoxA>
<kernel32.CreateFileA> ←
<kernel32.GetFileSize>
<kernel32.CloseHandle>
<kernel32.GlobalAlloc>
<user32.SetDlgItemTextA>
<kernel32.CloseHandle>
<kernel32.ReadFile> ←
<kernel32.CloseHandle>
<kernel32.GlobalFree>
<kernel32.CloseHandle>
<user32.SetDlgItemTextA>

GetComputerName: Recupera solo el nombre NetBIOS del equipo local.

CreateFileA: Crea o abre un archivo o dispositivo de E/S. Los dispositivos de E/S más usados son los siguientes: archivo, secuencia de archivos, directorio, disco físico, volumen, búfer de consola, unidad de cinta, recurso de comunicaciones, mailslot y canalización

ReadFile: Lee datos del archivo o dispositivo de entrada/salida (E/S) especificado. Las lecturas se realizan en la posición especificada por el puntero del archivo si el dispositivo lo admite. Esta función está diseñada tanto para operaciones sincrónicas como asincrónicas.

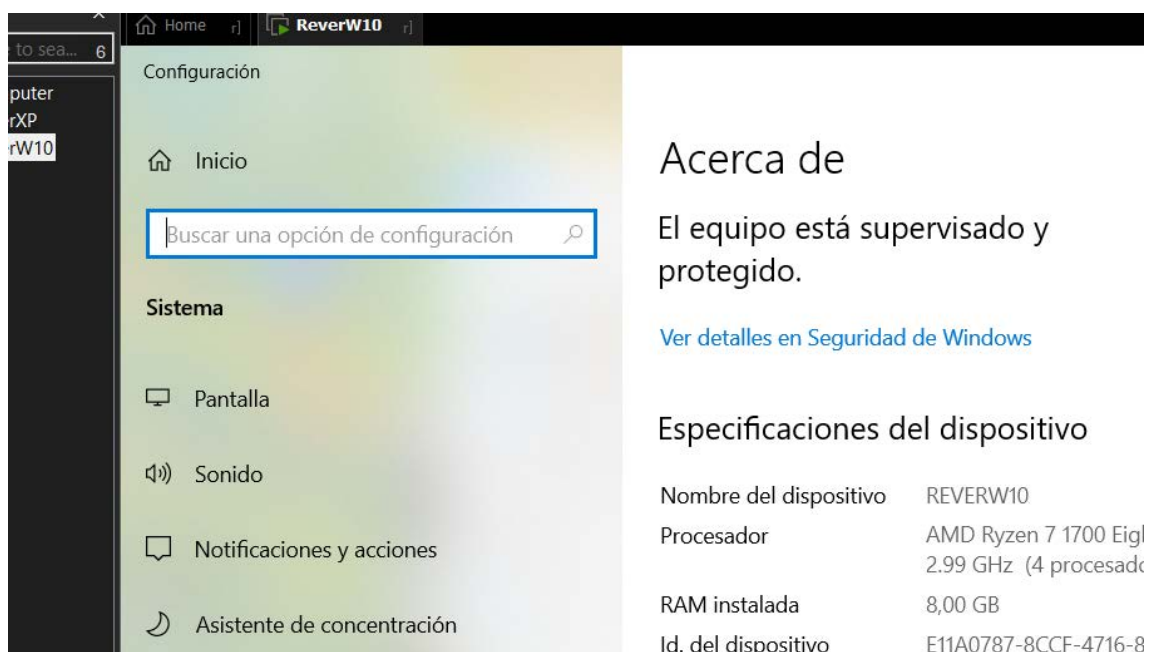
DialogBoxParam: La función DialogBoxParam usa la función CreateWindowEx para crear el cuadro de diálogo. A continuación, DialogBoxParam envía un mensaje de WM_INITDIALOG (y un mensaje de WM_SETFONT si la plantilla especifica el estilo DS_SETFONT o DS_SHELLFONT) al procedimiento del cuadro de diálogo. La función muestra el cuadro de diálogo (independientemente de si la plantilla especifica el estilo WS_VISIBLE), deshabilita la ventana del propietario e inicia su propio bucle de mensajes para recuperar y enviar mensajes para el cuadro de diálogo. Cuando el procedimiento del cuadro de diálogo llama a la función EndDialog, DialogBoxParam destruye el cuadro de diálogo, finaliza el bucle de mensajes, habilita la ventana del propietario (si se ha habilitado previamente) – al margen de esto, creo que esto es lo que nos deshabilita el botón “Register”

¹ Recordar que toda la documentación sobre API's – se encuentra buscando por Google “msdn + NombreApi” -> y lo importante es ¿Qué hace? - ¿Qué Parametros Usa? - ¿Que devuelve?

Que cosa curiosa que es esto del cracking, si recordamos unas hojas más arriba yo deje una imagen de cómo se veía la búsqueda de strings la cual tenía varias cadenas. Pero luego de realizar la búsqueda de las API's y revisar que era cada una, se me ocurrió volver a buscar las strings, luego de haber ejecutado el crackme con F9 y encontré esto:

```
String
"REVERW10"
"REVERW10"
"REVERW10"
"REVERW10"
"[ info ]"
"reg.key"
"Memory allocation error!"
"Good work. You have done it!"
"Registration failed!!!"
"Step 1 ok -> now Register it!"
"Registration failed!!!"
"\t\f\t\f"
L"CURRENT_USER"
L"MACHINE"
L"USERS"
L"USER"
L"\\\\"
L"\\REGISTRY\\"
L"CLASSES_ROOT"
L"CURRENT_USER"
L"CONFIG"
L"MACHINE"
L"USERS"
L"CLASSES_ROOT"
```

Recordemos la API "GetComputerName: Recupera solo el nombre NetBIOS del equipo local."
→ y como ese nombre es como suelo nombrar más o menos las VM que creo, me llamo la atención:



Como vi eso, bueno quería saber qué hacía con el nombre del host donde corre el crackme. Por lo que se me ocurrió poner un breakpoint en el call que hace referencia a la API antes mencionada -

The screenshot shows a debugger window with the following assembly code:

```

00401096 68 3B324000 call <JMP.&GetComputerNameA>
0040109B E8 A0020000 push keyme1.40323B
004010A5 A3 46334000 call <JMP.&Istrlen>
004010A8 59 mov dword ptr ds:[403346],eax
004010AB 58 pop ecx
004010AD 58 pop eax
004010AF 60 pushad
004010B1 9C pushfd
004010B3 BE 3B324000 mov esi, keyme1.40323B
004010B6 C705 3E334000 00000000 mov dword ptr ds:[40333E],0
004010BB AC lodsb
004010BD 0FB6C0 movzx eax,al
004010C0 0105 3E334000 add dword ptr ds:[40333E],eax
004010C3 3C 00 cmp al,0
004010C5 75 F2 jne 4010B8
004010C7 9D popfd
004010C9 61 popad
004010CB 33C0 xor eax,eax
004010CD C9 leave
004010CF C2 1000 ret 10
004010D2 837D 0C 10 cmp dword ptr ss:[ebp+C],10
004010D5 75 0A jne 4010DB
004010D7 6A 00 push 0

```

The right pane shows the CPU registers and the string "REVERW10".

Vemos que luego de tomar el nombre del host – está el call <JMP.&Istrlen> -> Según lo que entendemos es que Istrlen “Determina la longitud de la cadena especificada (sin incluir el carácter nulo final).”

Ok seré sincro, me puse a tracear a puro F8 y no llegue a nada más que al Chico MALO, resignado a que solo “perdía el tiempo” ya iba a reiniciar nuevamente para ver si no se me había pasado algo por alto, pero es aquí donde, encontré algo interesante traceando en el código me encontré con esto:

The screenshot shows a debugger window with the following assembly code:

```

0040125D E8 D2000000 push dword ptr ds:[403346]
00401262 66:83F8 67 call <JMP.&GlobalFree>
00401266 0F85 98000000 cmp ax,67
0040126C 6A 00 jne 401304
0040126E E8 03010000 push 0
00401273 6A 01 call <JMP.&OpenClipboard>
00401275 E8 E4000000 push 1
0040127A 0BC0 call <JMP.&GetClipboardData>
0040127C 74 72 or eax,eax
0040127E 74 72 je 4012F0
00401283 A3 33324000 mov dword ptr ds:[403233],eax
00401285 60 pushad

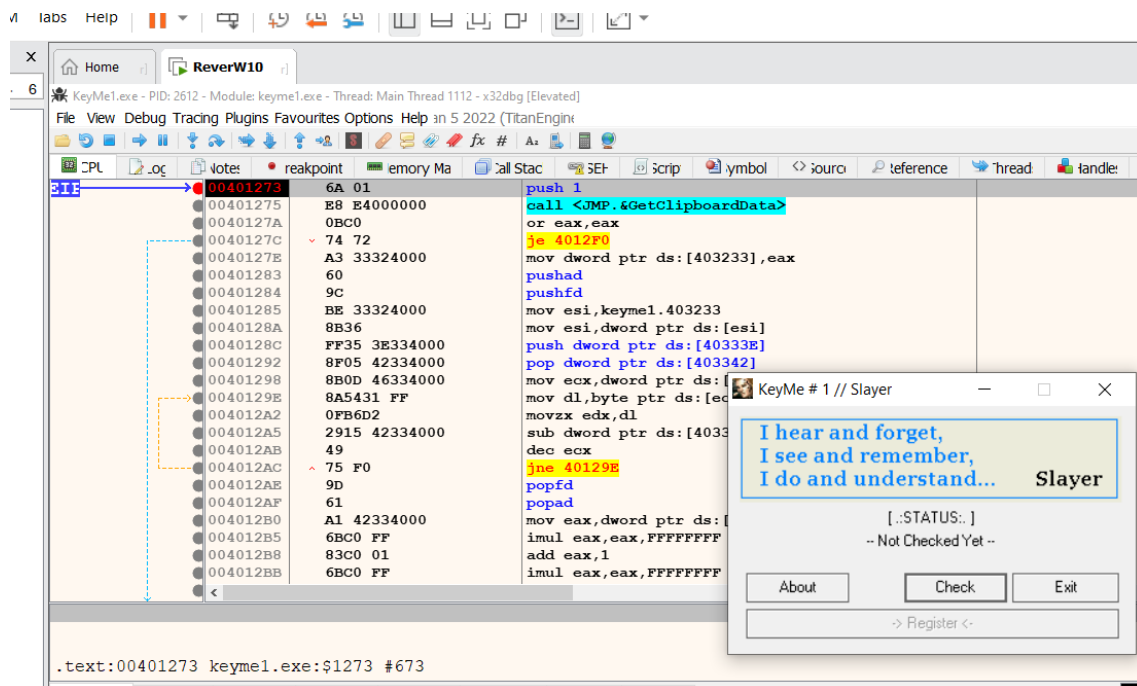
```

OpenClipboard: Abre el portapapeles para su examen y evita que otras aplicaciones modifiquen el contenido del portapapeles.

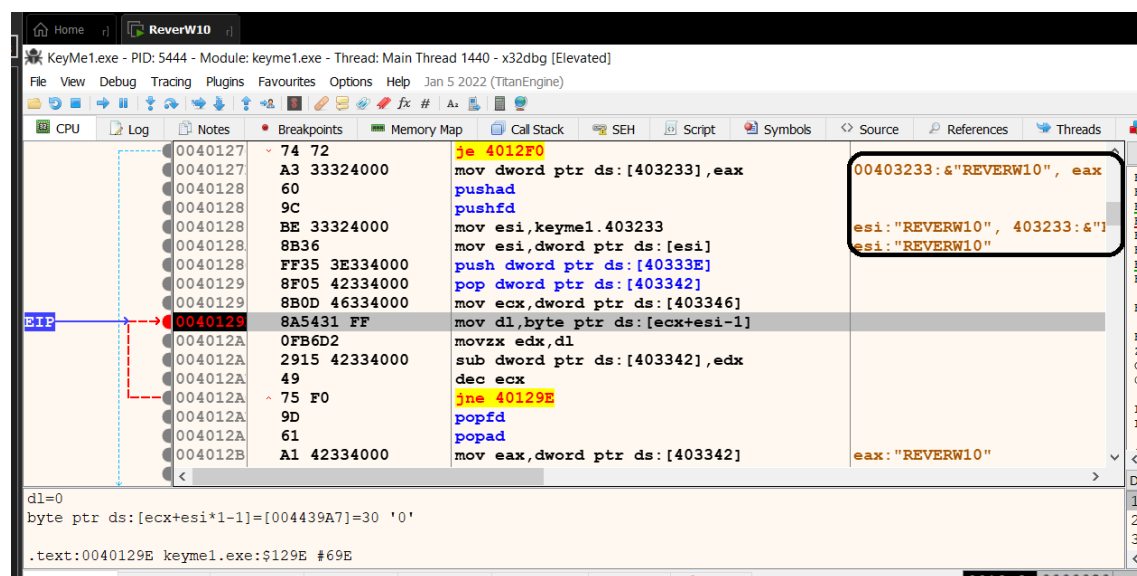
GetClipboardData: Recupera datos del portapapeles en un formato específico. El portapapeles debe haberse abierto previamente.

Esto me hizo pensar entonces que algo debería tener en el portapapeles, y luego el Get recupera lo que tengo ahí así que voy a hacer algo de “trampa” voy a copiarme el nombre del host “ReverW10” y voy a poner un Breackpoint en el primer call, en donde abre el portapapeles a ver qué pasa, reinicio y lo pongo a correr con F9.

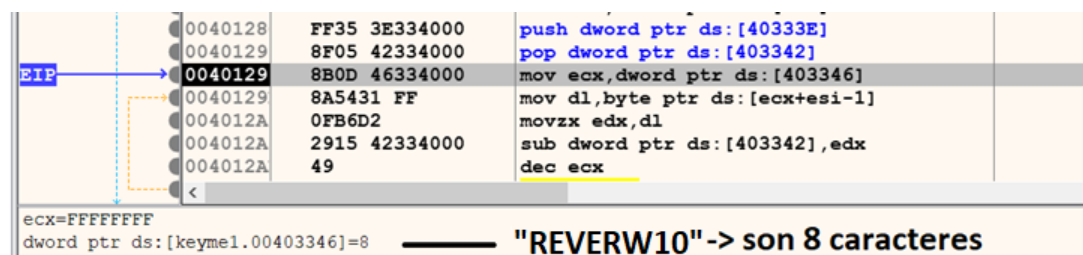
Como no hacía nada, seleccione el botón “Check” y aquí se detuvo:

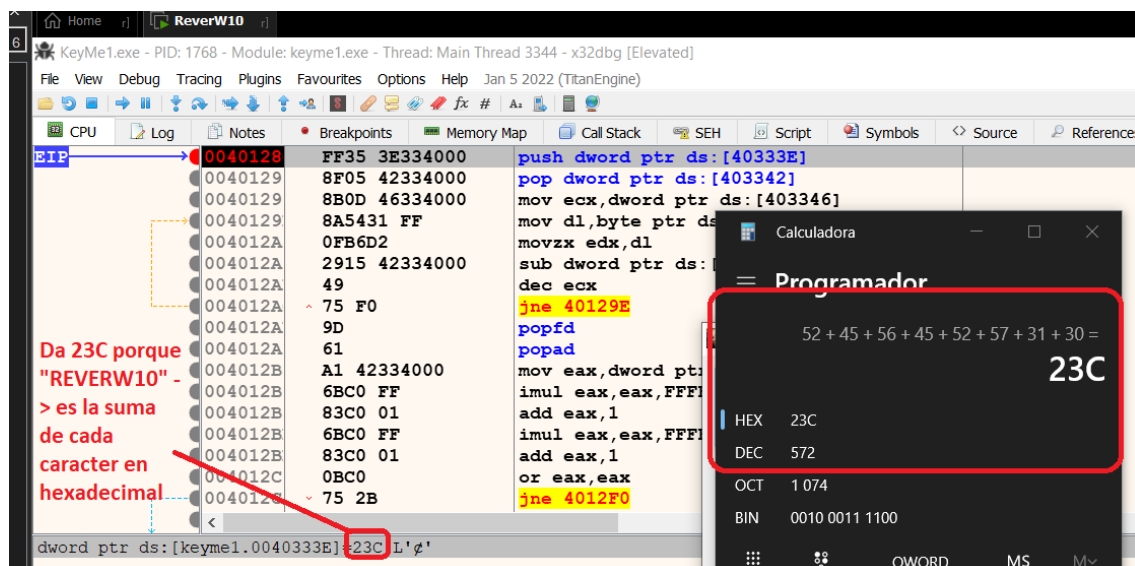


Luego comienza a ir y venir y hacer un montón de vueltas, así que intentare ir simplificando un poco (nota: si tuve que hacerlo varias veces por que me perdía)

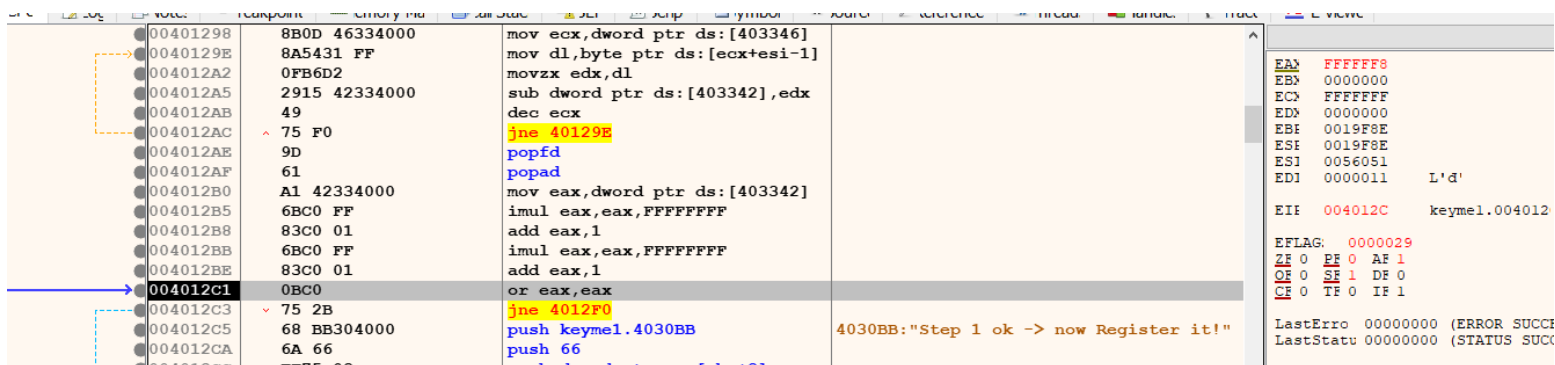


Aquí por ejemplo entra a recorrer la string “REVERW10” y veo que hace varias operaciones



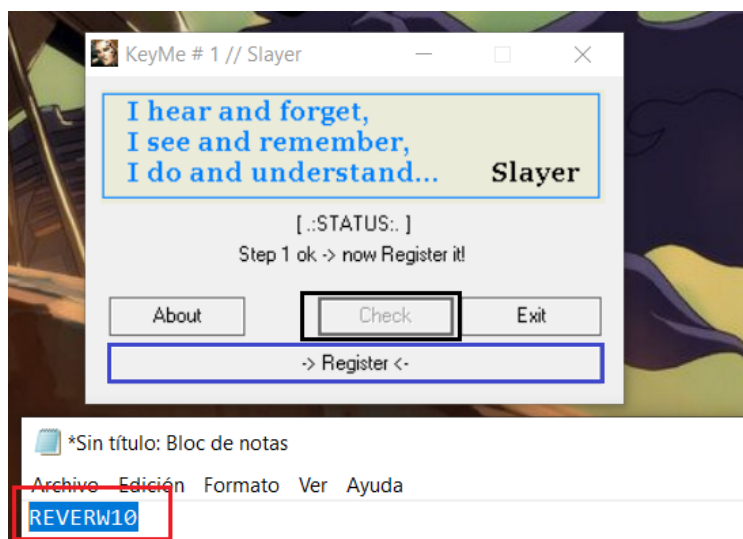


Por lo que entonces lo que tiene [40333E] seria la sumatoria de los caracteres en hexadecimal del nombre del host. Y fin...



Como esto me llevó mucho tiempo y.... hay cosas que me exceden en cuestiones de solo copiar y pegar una imagen, deberán confiar o probar por ustedes mismos.

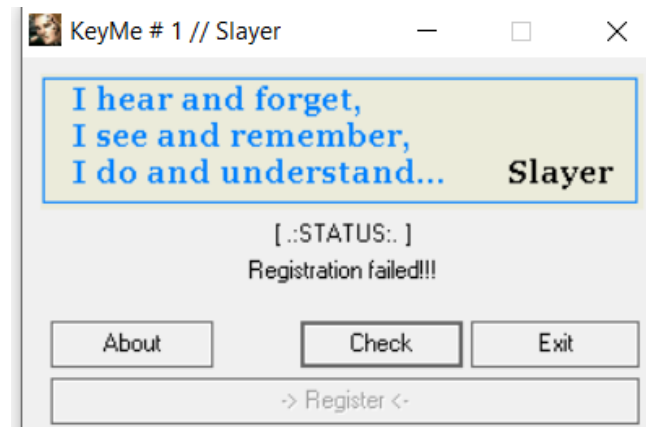
Ok luego de tanta cosa de ir y venir y revisar varias veces las anotaciones, en un momento breve de lucides, comencé a penar que en algún lado usaba la api **GetClipboardDat** o al menos era una de las posibilidades. Así que decidí probar copiando el nombre de mi host en el portapapeles y darle a "Check"



Como el nombre de mi host es "REVERW10" (sin comillas) lo copie en un bloc de notas. Luego abrí el KeyMe #1 de Slayer -> le di al botón Check y me habilitó el botón "-> Register <-"

Genial un paso más y no hemos roto la regla la cual "impide" que parchemos el programa.

La felicidad duró muy poco, el tiempo que me tomo darle a Register y darme cuenta que volvemos al principio... o algo así:



Es aquí donde me no me quedaron otras ideas que seguir traceando y encontré lo siguiente:

00401112	6A 03	push 3	
00401112	6A 00	push 0	
00401112	6A 01	push 1	
00401112	68 00000080	push 80000000	
00401113	68 26314000	push keyme1.403126	
00401113	E8 D3010000	call <JMP.&CreateFileA>	403126: "reg.key"
00401113	83F8 FF	cmp eax,FFFFFFFF	
00401114	75 05	jne 401117	

Ya había visto la API **CreateFileA** – pero para tenerlo a mano, básicamente lo que hace es:

Crea o abre un archivo o un dispositivo de E/S. Los dispositivos de E/S más utilizados son los siguientes: archivo, secuencia de archivos, directorios, etc.

Bien, pero crear archivos, al menos este KeyMe, no lo hace, por lo que decidí inicialmente crearlo yo. ¿Y por qué no? Entonces me dispuse y cree un reg.key en la misma ubicación que el KeyME1.exe (en el escritorio) - En cuanto al contenido de dicho archivo, se me ocurrió usar el mismo nombre que la pc, en mi caso "REVERW10" pero con la consideración de tener ese nombre también en el portapapeles. Reg.key lo manipule con Notepad en principio. Ahora bien, luego de revisar llegamos a esta línea:

4011F	8B06	mov eax,dword ptr ds:[esi]	
4011F	83C6 04	add esi,4	
4011F	8B16	mov edx,dword ptr ds:[esi]	
4011F	33C2	xor eax,edx	
4011F	8B15 42334000	mov edx,dword ptr ds:[403342]	
40120	03C2	add eax,edx	
40120	5A	pop edx	
40120	5E	pop esi	
40120	3B05 3E334000	cmp eax,dword ptr ds:[40333E]	
40120	75 1E	jne 40122E	
40121	68 D9304000	push keyme1.4030D9	
40121	6A 66	push 66	

Expression: 45564552

Bytes: 52455645

Signed: 1163281746

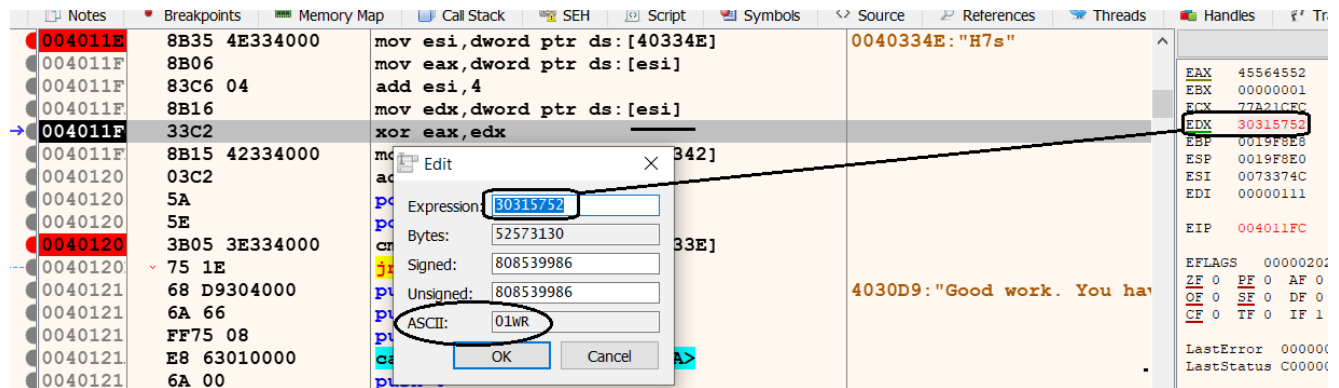
Unsigned: 1163281746

ASCII: EVER

EAX	45564552
EBX	00000001
ECX	77A21CFC
EDX	00000000
EBP	0019F8E8
ESP	0019F8E0
ESI	00733748
EDI	00000111
EIP	004011F7
EFLAGS	00000246
ZF	1
PF	1
AF	0
OF	0
SF	0
DF	0
CF	0
TF	0
IF	1

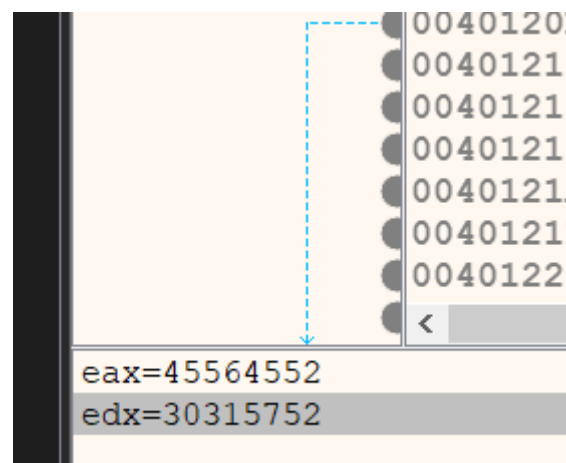
Observé que en EAX estaba el valor 45564552 – que en ASCII como indica, se representa como EVER –

Pero, en realidad, si recordamos, aquí lo muestra como “Little Endian” -> por lo que en realidad lo que estamos viendo ahí es “REVE” Y luego también pude verificarlo en EDX que tiene el valor 30315752 que representa en ASCII “01WR” – también en Little Endian.

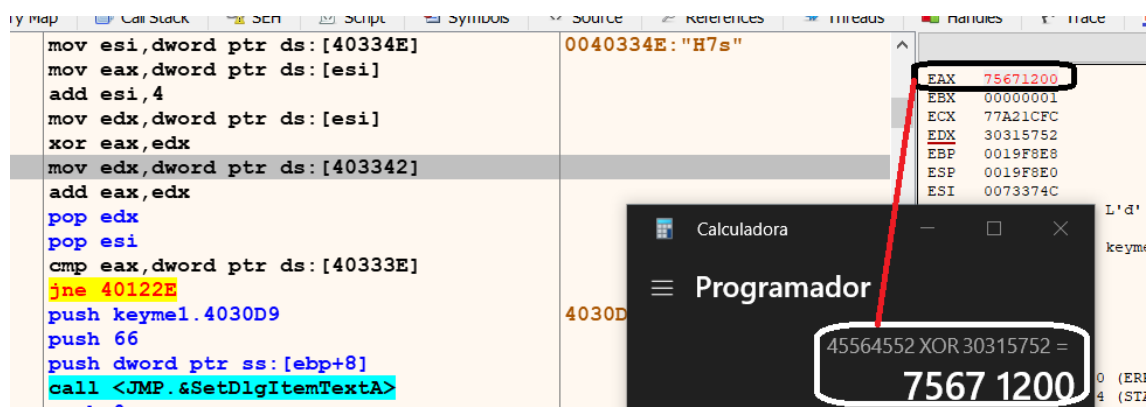


Es decir que toma en principio los primeros 4 caracteres del nombre de mi PC – y los pasa a EAX en Little Endian como mencionamos antes, luego toma los siguientes 4 caracteres también y los pone en EDX – y si vemos bien la imagen luego realiza un XOR EAX, EDX –

Esto me costó muchísimo entenderlo, por eso aprovecho a agradecer aquella buena persona que me ayudo a comprenderlo jeje – “Lo más importante es siempre creer en uno mismo, pero una pequeña ayuda de los demás es una gran bendición” – para ponerlo un poco más claro quizás, sigamos el ejemplo de mi PC – el nombre es REVERW10 – bien lo que hace es lo siguiente: a EAX le pasa “EVER” (que es REVE) y a EDX le pasa 01WR(que sería RW10) – PERO ATENCION: cuando hace XOR EAX, EDX -> lo hace con los valores que tenemos en EAX Y EDX correspondientes – es decir EAX = 45564552 y EDX = 30315752 – De hecho esto lo vemos un poco mas abajo en la aclaración del desensamblador:



Aquí vemos el resultado y lo deja en EAX



En esta línea veo que eax entonces tiene el resultado xoreado y lo que contiene [40333E] Es 23C – que si recordamos 23C es el resultado de la suma de cada carácter en HEXA del nombre de mi host “REVERW10”

```

00401200  3A          pop     eax
00401201  5E          pop     esi
00401202  3B05 3E334000  cmp     eax, dword ptr ds:[40333E]
00401203  75 1E       jne     40120B
00401204  68 D9304000  push    keyme1.4030D9
00401205  6A 66       push    66
00401206  FF75 08     push    dword ptr ss:[ebp+8]
00401207  E8 63010000  call    <JMP.&SetDlgItemTextA>
00401208  6A 00       push    0
00401209  FF35 5A334000  push    dword ptr ds:[40335A]

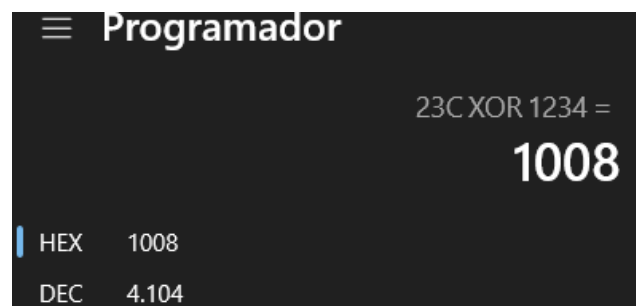
```

eax=75671200
dword ptr ds:[keyme1.0040333E]=23C L'ç'

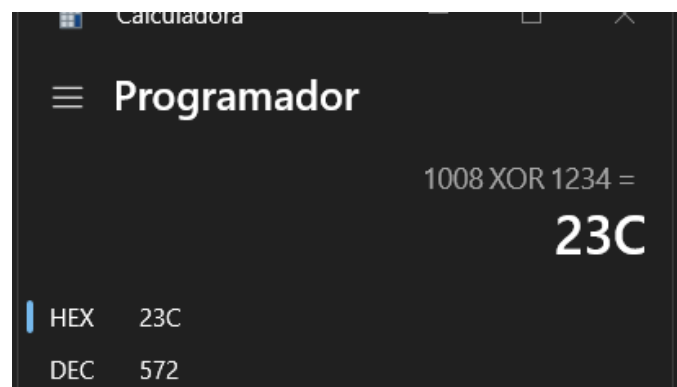
Entonces, lo que compara es básicamente lo que indica la imagen – compara el resultado xoreado entre EAX y EDX (que redoramos que toman los primeros 4 bytes en uno y en el otro los siguientes 4, al menos en mi ejemplo) y eso lo hace contra la suma de cada carácter del nombre de mi PC en hexadecimal. Entonces en reg.key tengo que tener los valores que al xorearse, en mi caso lleguen a 23C – Gracias Solid por indicarme lo siguiente, quizás a otros les sirva – Si pensamos que $A \text{ XOR } B = 23C$ – A y B son mis incógnitas que haciendo XOR da como resultado 23C

“Si $A \text{ XOR } B = C \rightarrow C \text{ XOR } B = A \rightarrow A \text{ XOR } C = B$ ”

Y aquí viene el gran detalle que me exploto el cerebro ja. Es que no soy muy bueno en matemáticas y demás. Pero técnicamente, en nuestra situación con el KeyMe en realidad tenemos una sola incógnita. - Pero ¿cómo? ¿Por qué? – bueno ... yo lo entendí así: $C = 23C$ y A o B debería ser una constante o un numero fijo. Por ejemplo, yo hice así:



¿Es decir que en Hexa 1008 o en Decimal 4104 – sería nuestra incógnita – por qué? Por la formula de antes- Si yo hago: 10008 XOR 1234 Debería dar 23C – Tal y como nos muestra la calculadora:



Tambien lo podemos probar con la consola de Python

```
Símbolo del sistema - python
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados

C:\Users\Home>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023)
Type "help", "copyright", "credits" or "license()" for more
>>> 10 ^ 20
30
>>> 30 ^ 20
10
>>> 10 ^ 30
20
>>>
```

```
Símbolo del sistema - python
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados

C:\Users\Home>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023)
Type "help", "copyright", "credits" or "license()" for more
>>> 10 ^ 20
30
>>> 30 ^ 20
10
>>> 10 ^ 30
20
>>> 0x23c ^ 1234
1774
>>> 1774 ^ 1234
572
>>> hex(572)
'0x23c'
>>>
```

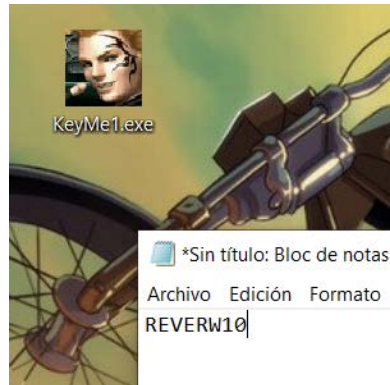
Repasemos entonces más o menos a lo que llegamos:

- 1 – Para activar el botón de “Register” – necesitamos tener en el portapapeles el nombre de la PC
- 2 – Luego debemos tener (al menos hasta donde probe el reg.key) con el KeyMe1.exe – el cual el valor que tenga allí, deba ser un valor no mas de 8 bytes (4 para el primero y 4 para el segundo) Ahhh pero aca hay un detalle, y eso es que debería ser en Little Endian.

Para resolverlo:

Bueno entonces vamos a hacer caso a consejos de sabios, probemos primero hacerlo “a papel” luego veremso como hacer el código en Python- siguiendo con mi ejemplo- el nombre de mi Host es REVERW10

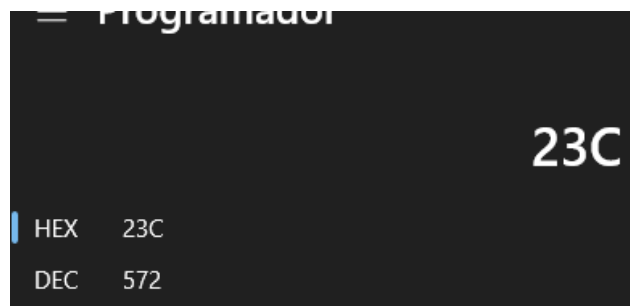
Entonces si mal no entendimos y tracemos e hicimos todo bien – Primero asegurarnos de tener dicho nombre en el portapapeles. Ok eso esta



Bien ahora, si recordamos lo anterior dicho sabemos que REVERW10 = 23C en Hexa – es decir que mi reg.key cuando haga las operaciones debe ser igual a 23C

Es decir, $23C \text{ XOR } 1234 = 1008 \text{ (Hexa)}$ o 4104 (Decimal) – tomo 1234 por que un numero cualquiera (obviamente caundo hagan los cálculos deberían hacerse con los números correspondientes)

En real hay una pequeña trampa, yo cuando lo hice inicialmente, con la calculadora, tome $23C \text{ (HEXA)} \text{ XOR } 1234 \text{ (HEXA)}$ pero no me da los valores correctos en si – si bien luego i el resultado lo xoreo me vuelve a dar 23C – en realidad la cuenta debería ser asi:
 $23C \rightarrow$ En decimal es 572

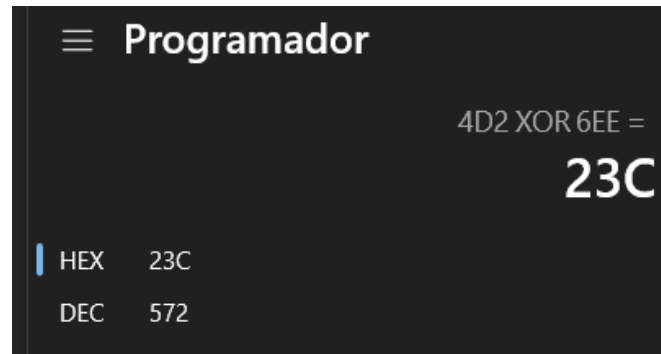


Y 1234 en decimal- es decir hice $572 \text{ XOR } 1234$



Es decir que nos da 6EE en Hexa o 1774 en decimal – Genial

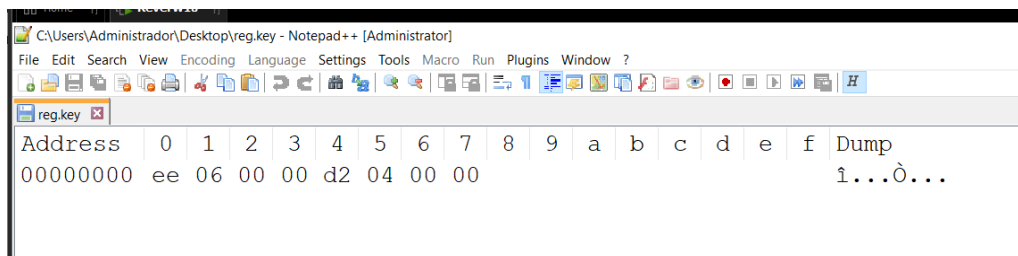
Entonces básicamente si hago 6EE XOR 4D2 debería dar 23C



4D2(hexa) = a 1234 en decimal

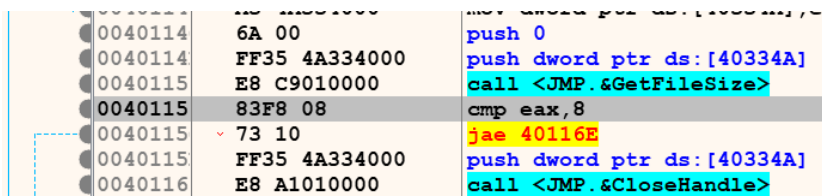
Bien entonces me encontré con otro problema – se supone que deberíamos hacer que en bloc de notas – en formato ASCII y en little endian se guarde el valor correspondiente a 4D26EE – para ello, además, deberemos completar los bytes faltantes

Con la ayuda del Notepad++ nos quedaría así:

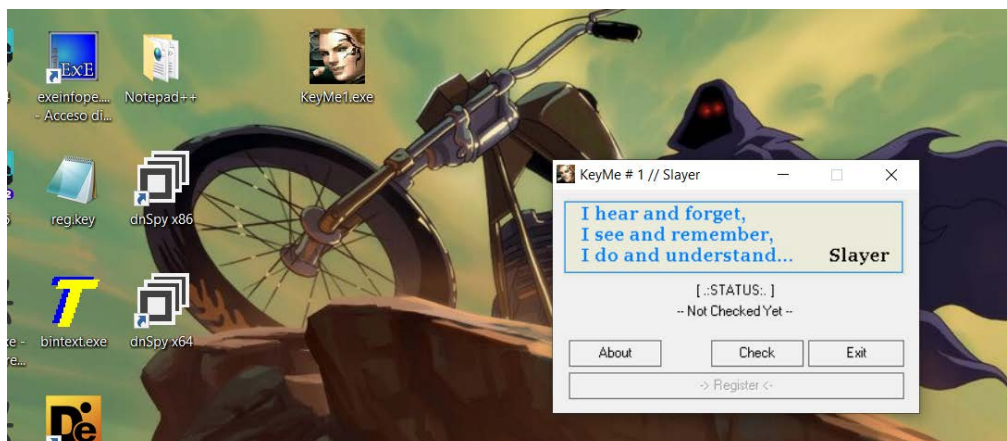


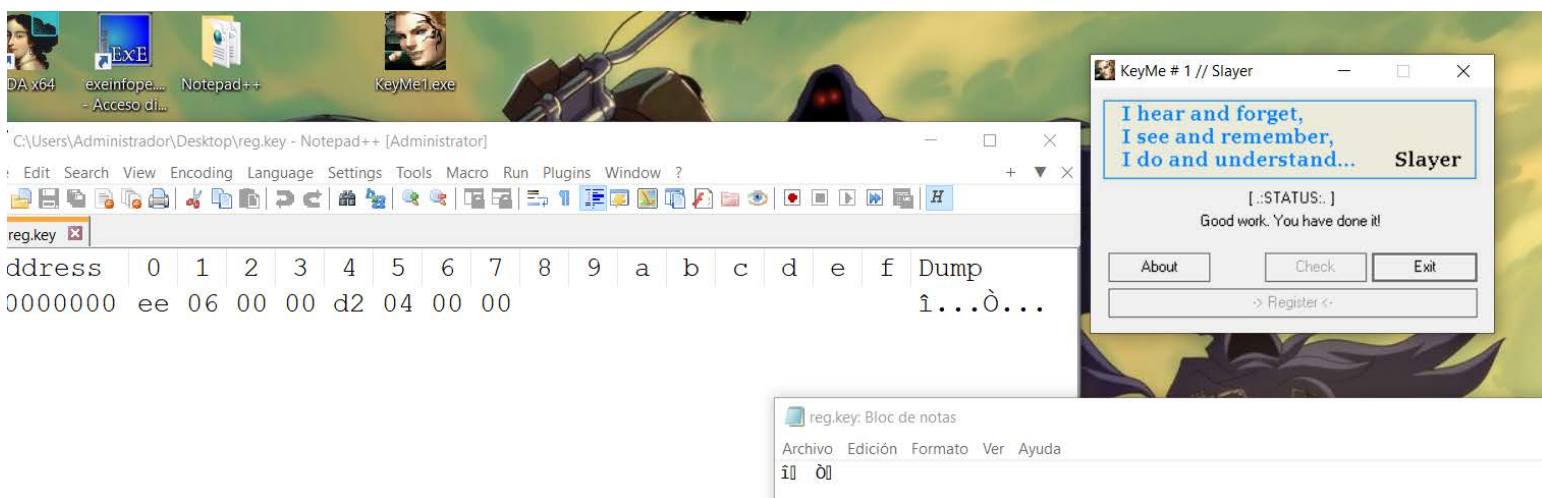
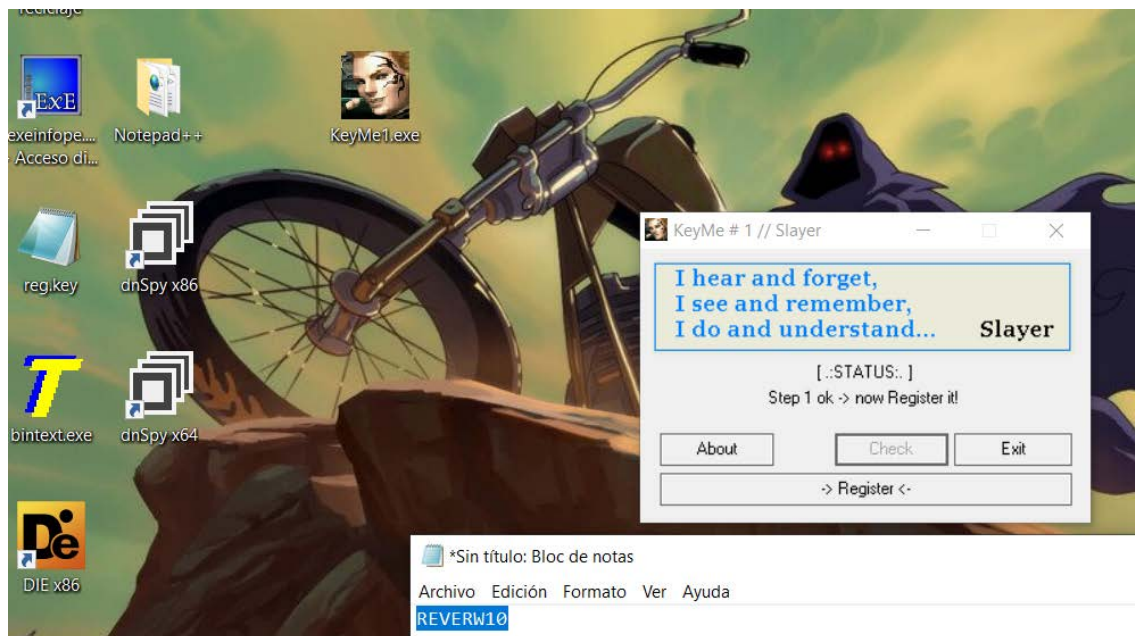
Es decir, EE060000D2040000 (Así queda representado en ASCII y completando los 8 bytes que en alguna parte del programa lo evalúa)

Si mal no recuerdo era por aquí:



Bien entonces – si usara ese reg.key con ese nombre de host y ejecuto el KeyMe de Slayer debería funcionar: veamos:





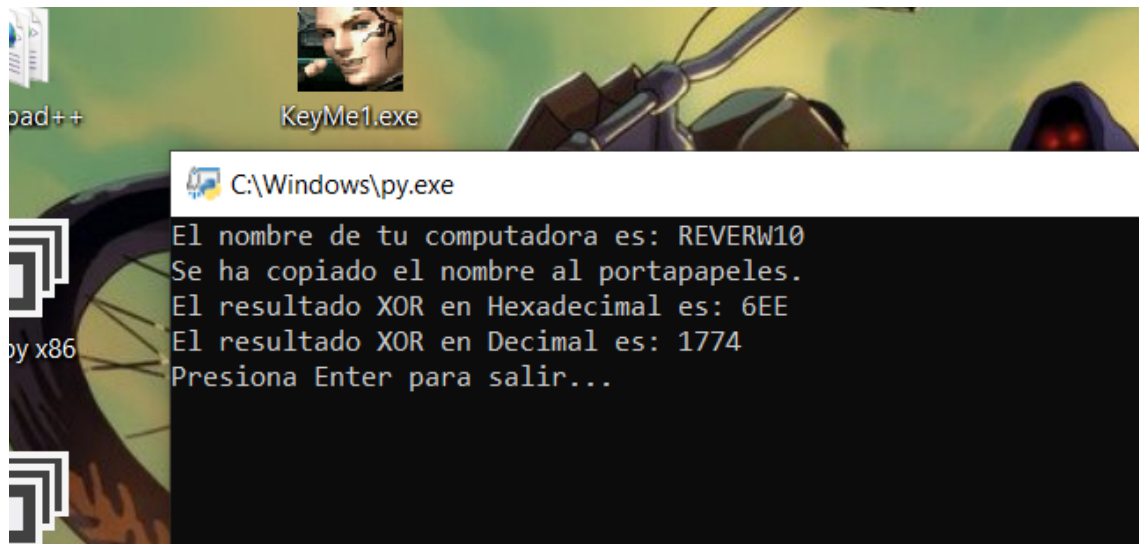
¡Bien! Lo hemos logrado :D notar que en el Notepad común es lo mismo que dice el Notepad++ pero este último, con el plugin HEX EDITOR -> View in HEX nos muestra como sería en hexadecimal

Por último les dejo como sería un script en Python:

```
import socket
import subprocess
import time
import os
import struct

nombre_pc = socket.gethostname() #Tomo el nombre de la PC
constante_xor = 1234 #Defino una constante para luego operar
contador = 0
# Calculo la suma hexadecimal directamente
for caracter in nombre_pc:
    valor_decimal = ord(caracter)
    valor_hexadecimal = hex(valor_decimal)
    contador += valor_decimal
sumaHEX = contador ^ constante_xor #Aca realizo la operacion de hacer la CONSTANTE XOR el resultado
estru0 = struct.pack("<I", constante_xor) #Converto datos de Python en una cadena de bytes, siguiendo un formato específico.
estru1 = struct.pack("<I", sumaHEX) #<: Indica que se usará el orden de bytes "little-endian"
final = estru1 + estru0 #Sumo las dos variables donde converti cada cadena de bytes
# Copiamos solo el nombre de la PC al portapapeles
subprocess.run(["clip"], input=nombre_pc.encode(), check=True)
# Imprimimos los resultados por consola
print("El nombre de tu computadora es:", nombre_pc)
print("Se ha copiado el nombre al portapapeles.")
print(f"El resultado XOR en Hexadecimal es: {sumaHEX:X}")
print(f"El resultado XOR en Decimal es: {sumaHEX}")
# Creo el archivo
ruta_escritorio = os.path.join(os.environ['USERPROFILE'], 'Desktop')
ruta_archivo = os.path.join(ruta_escritorio, "reg.key")
with open(ruta_archivo, "wb") as archivo:
    archivo.write(final)
# Mantener la consola abierta hasta que el usuario presione Enter
input("Presiona Enter para salir...")
```

El resultado como se vería si lo ejecuto en el hshot con nombre "REVERW10"



Aclaraciones:

- Si la verdad que este crackme... en lo personal fue bastante difícil, pero con un poco de ayuda, paciencia y pensar, sale. Además, que está muy bueno porque los detalles que tiene, como por ejemplo que APIS usa.
- Esto funciona sin problemas en pc de 64 bits (no cuento con alguna vm que sea de 32 bits).
- Todo va bien si el nombre esta en mayúsculas, pero si el nombre del host fuera ReverW10, por ejemplo – a la hora de generar el reg.key lo genera todo, pero al querer poner “Check” no me lo valida. Esto se debe a un error que tuve cunado realicé el código en Python.

¿Como lo resolví? Sencillo, cambiando la línea donde toma el nombre, que lo tome en mayúsculas.

```
import socket
import subprocess
import time
import os
import struct

nombre_pc = socket.gethostname().upper()#Tomo el nombre de la PC en mayus
constante_xor = 1234 #Defino una constante para luego operar
contador = 0
for caracter in nombre_pc: # Calculo la suma hexadecimal directamente
    valor_decimal = ord(caracter)
    valor_hexadecimal = hex(valor_decimal)
    contador += valor_decimal
sumaHEX = contador ^ constante_xor #Aca realizo la operacion de hacer la CONSTANTE XOR el resultado
estru0 = struct.pack("<I",constante_xor) #Converto datos de Python en una cadena de bytes, siguiendo un formato específico.
estru1 = struct.pack("<I", sumaHEX) #<: Indica que se usará el orden de bytes "little-endian"
final = estru1 + estru0 #Sumo las dos variables donde converti cada cadena de bytes
subprocess.run(["clip"], input=nombre_pc.encode(), check=True)# Copiamos solo el nombre de la PC al portapapeles
print("El nombre de tu computadora es:", nombre_pc)# Imprimimos los resultados por consola
print("Se ha copiado el nombre al portapapeles.")
print(f"El resultado XOR en Hexadecimal es: {sumaHEX:X}")
print(f"El resultado XOR en Decimal es: {sumaHEX}")
ruta_escritorio = os.path.join(os.environ['USERPROFILE'], 'Desktop')# Creo el archivo
ruta_archivo = os.path.join(ruta_escritorio, "reg.key")
with open(ruta_archivo, "wb") as archivo:
    archivo.write(final)
# Mantener la consola abierta hasta que el usuario presione Enter
input("Presiona Enter para salir...")
```

Toma ahora el nombre
en Mayusculas