



Crackme	Enigma Group - App Challenge 2
Misión	-Activar button -Encontrar el Password
Compilado	Visual Basic 6.0 Código Nativo
Empacado	No
Tools	RDG v0.7.6 - IDA v6.8
Sistema Operativo	Windows Xp SP3 o Superior (32 y 64 bits)
Reverser	QwErTy
Dedicado a	Enigma Group - CracksLatinoS - Ricnar - tincopasan
Descargar Crackme	https://mega.nz/#!3IN1ISgT!8KAQPMs-sAubXCRU3yvuiLxJ3d5nBgKMtsonEU6vAss

El Sistema Operativo Windows XP SP3 no es el más adecuado para trabajar con IDA. En este caso me he visto obligado a utilizarlo por problemas con mi otro PC.

Con este tute quisiera animar a los que todavía no se han iniciado con IDA a que lo hagan, a los que ya lo están utilizando, pueda servirles de alguna ayuda en sus avances, y a los más experimentados simplemente pasen un rato entretenido con su lectura, y sepan disculparme si he cometido errores.

*Descargamos el crackme siguiendo el enlace arriba indicado y lo descomprimimos.
(Esta comprimido con el 7-Zip y no requiere contraseña alguna para ello)*



ESTUDIANDO LA VÍCTIMA

Lo ejecutamos



Nos aparece un caja de texto para entrar un Password, dos buttons; "Submit" y "Cancel", un texto y el nombre del crackme.

Bien, vamos a introducir un Password cualquiera en la caja de texto para probar suerte, yo tipeo "QwErTy CLS", y a medida que lo voy entrando me lo esconde detrás de asteriscos.



Ahora le damos al button "Submit" para que compruebe la validez del Password, y por más que lo intentemos no hay manera....., a la que nos posicionamos sobre él con el cursor, descaradamente y de forma automática se desactiva.



El autor del crackme nos acaba de mostrar la primera misión a solucionar, salimos del mismo dándole a "Cancel"

CONTINUAMOS ESTUDIANDO LA VÍCTIMA

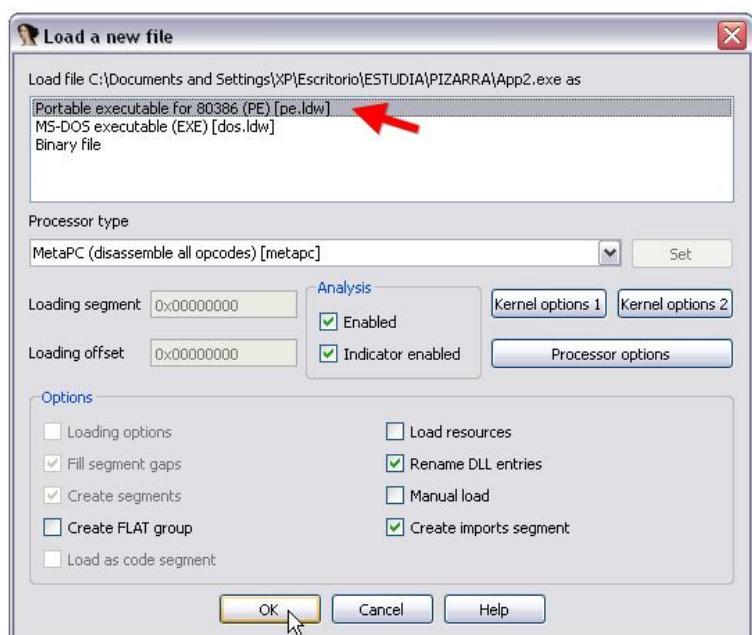
Le pasamos el **RDG Packer Detector**, para saber a lo que nos enfrentamos, y nos informa en el modo escaneo profundo “**M-B**”, que está compilado en “**Visual Basic 6.0 Código Nativo**”, de que no está empacado, y de que es de **32 bits**.



VAMOS A POR ELLA

PRIMERA MISIÓN - Activar el Button “Submit”

Vamos a buscar como activarlo, para ello cargamos el Crackme arrastrándolo y soltándolo sobre el ícono “**IDA Pro (32-bit)**”, Nos detecta el ejecutable, y dejamos todas las tildes por defecto



Le damos a “**OK**”, IDA lo abre/analiza correctamente y aparecemos en el punto de entrada, address “**0040122C**”

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Functions window Strings window Program Segmentation Hex View-1

Function name ThunRTMain

```

.text:0040122C          public start
.text:0040122C
.text:0040122C start:
.text:0040122C     push    offset dword_401E80
.text:00401231     call    ThunRTMain
.text:00401231 ; -----
.text:00401236         dw     0
.text:00401238         dd     0
.text:0040123C         dd     30h, 38h, 0
.text:00401248         dd     3B144994h, 4384219Bh, 543F3
.text:00401250         dd     1000h, 2 dup(0)
.text:00401268 aApp2
.text:0040126D
.text:00401270
.text:00401274 dword_401274 dd 31CCFFh, 1AA10A05h, 6A0AC74
.text:00401274 ; DATA
.text:00401274 dd 0F8883A2Bh, 8B448BCEh, 30AC
.text:00401274 dd 0CF669933h, 0CB711h, 0D3600
.text:004012CC         dd 0AE200h, 94000h, 0B0000h, 5
.text:004012CC         dd 1E010Dh, 67696E45h, 4720616
.text:004012CC         dd 43207070h, 6C6C6168h, 65676
.text:004012CC         dd 19h, 2 dup(0)
.text:00401318         dd 3F8000h, 43F8000h, 2301220
.text:00401318         dd 20200001h, 10000h, 8A80008h
.text:00401318         dd 400000h, 10000h, 8, 6 dup(0)
.text:00401370         dd 99330000h, 99330066h, 0CC66
.text:00401370         dd 0F9900CCh, 0CC990099h, 996
.text:00401370         dd 66330033h, 5F5F0066h, 0FFCC
.text:00401370         dd 0CCC0066h, 4D400066h, 6600
.text:00401370         dd 0CC330055h, 0CC330066h, 0C0
.text:00401370         dd 66330000h, 0FF660000h, 0CC6
.text:00401370         dd 86h, 499h, 330004CCh, 33000
.text:00401370         dd 33000499h, 330004CCh, 66000

```

0000122C 0040122C: .text:start (Synchronized with Hex View-1)

Output window

The initial autoanalysis has been finished.

Python

AU: idle | Down | Disk: 249GB

Bien, como siempre seguro que hay más métodos para solucionarlo pero yo lo he hecho de la siguiente forma; nos vamos a la pestaña "Strings window" para ver si hay algo interesante y después de muchos intentos con otras APIs del listado que nos muestra IDA, me quedo con "__vbaVarTstEq", que compara o chequea strings, y "__vbaEnd" que como también sabemos llama a fin de una función, fin de una instrucción, o a finalización de programa.

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Functions window Strings window Program Segmentation

Function name ThunRTMain

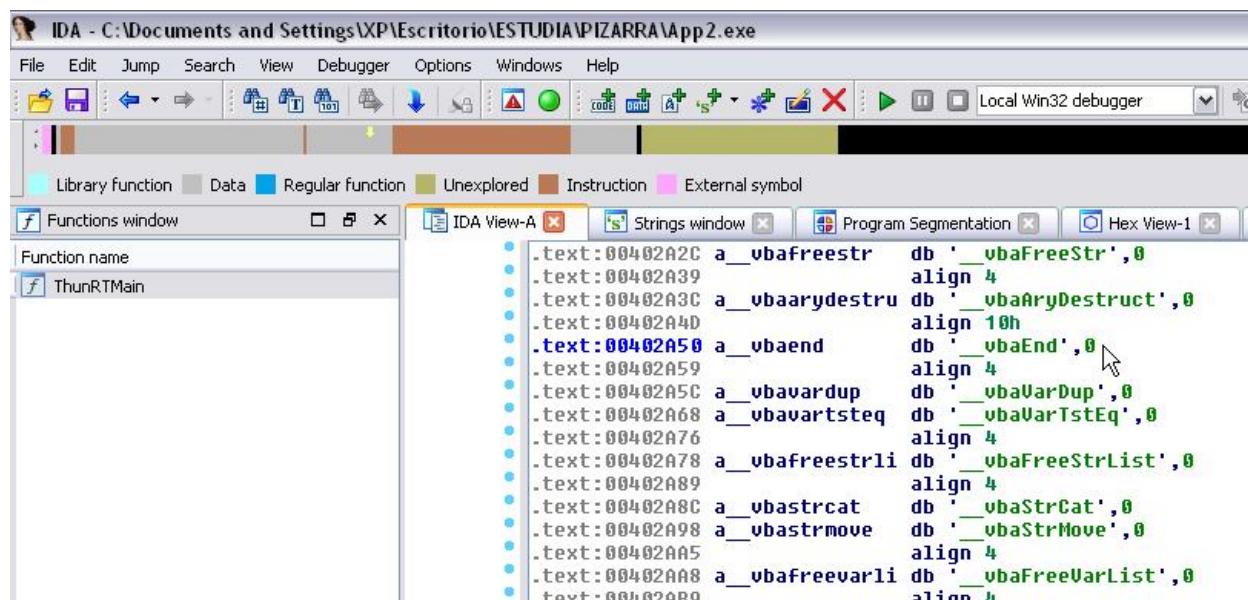
Address	Length	Type	String
'\$'.text:00401...	00000005	C	App2
'\$'.text:00401...	00000005	C	App2
'\$'.text:00401...	00000005	C	App1
'\$'.text:00402...	0000000C	C	frmPassword
'\$'.text:00402...	00000005	C	App2
'\$'.text:00402...	0000000C	C	lblPassword
'\$'.text:00402...	0000000A	C	cmdCancel
'\$'.text:00402...	00000009	C	VBA6.DLL
'\$'.text:00402...	00000013	C	__vbaErrorOverflow
'\$'.text:00402...	0000000D	C	__vbaFreeStr
'\$'.text:00402...	00000011	C	__vbaAryDestruct
'\$'.text:00402...	00000009	C	vbaEnd
'\$'.text:00402...	0000000C	C	__vbaVarDup
'\$'.text:00402...	0000000E	C	__vbaVarTstEq
'\$'.text:00402...	00000011	C	__vbarreestrList
'\$'.text:00402...	0000000C	C	__vbaStrCat
'\$'.text:00402...	0000000D	C	__vbaStrMove
'\$'.text:00402...	00000011	C	__vbaFreeVarList
'\$'.text:00402...	00000011	C	__vbaFreeObjList
'\$'.text:00402...	00000019	C	__vbaGenerateBoundsError
'\$'.text:00402...	0000000D	C	__vbaStrCopy
'\$'.text:00402...	00000013	C	__vbaAryConstruct2
'\$'.text:00402...	0000000D	C	__vbaFreeObj
'\$'.text:00402...	00000015	C	__vbaResultCheckObj
'\$'.text:00402...	0000000C	C	__vbaObjSet
'\$'.text:00403...	0000000D	C	MSVBVM60.DLL

Empezamos con “_vbaEnd” ya que mi instinto de cracker me dice que si localizo el punto donde finaliza una función determinada, (en nuestro caso el button “Submit” si no cumple una determinada regla, queda desactivado y por tanto función finalizada y no nos deja continuar), igual encuentro algo interesante por los alrededores...

Vamos allá. Nos posicionamos con el cursor sobre la misma APIs “_vbaEnd”

'S'	.text:00402...	00000005	C	App2
'S'	.text:00402...	0000000C	C	lblPassword
'S'	.text:00402...	0000000A	C	cmdCancel
'S'	.text:00402...	00000009	C	VBA6.DLL
'S'	.text:00402...	00000013	C	_vbaErrorOverflow
'S'	.text:00402...	0000000D	C	_vbaFreeStr
'S'	.text:00402...	00000011	C	_vbaAryDestruct
'S'	.text:00402...	00000009	C	<u>_vbaEnd</u>
'S'	.text:00402...	0000000C	C	_vbaVarDup
'S'	.text:00402...	0000000E	C	_vbaVarTstEq
'S'	.text:00402...	00000011	C	_vbaFreeStrList

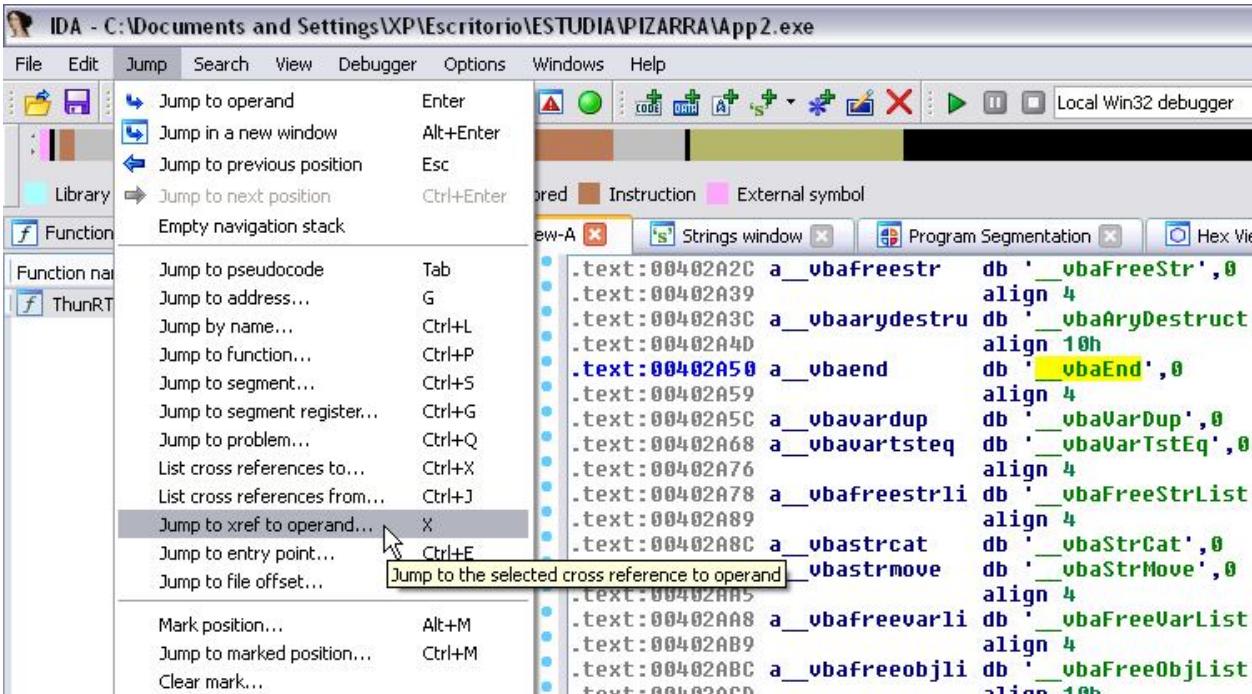
Le damos dos clicks Izquierdo de ratón y aparecemos aquí:



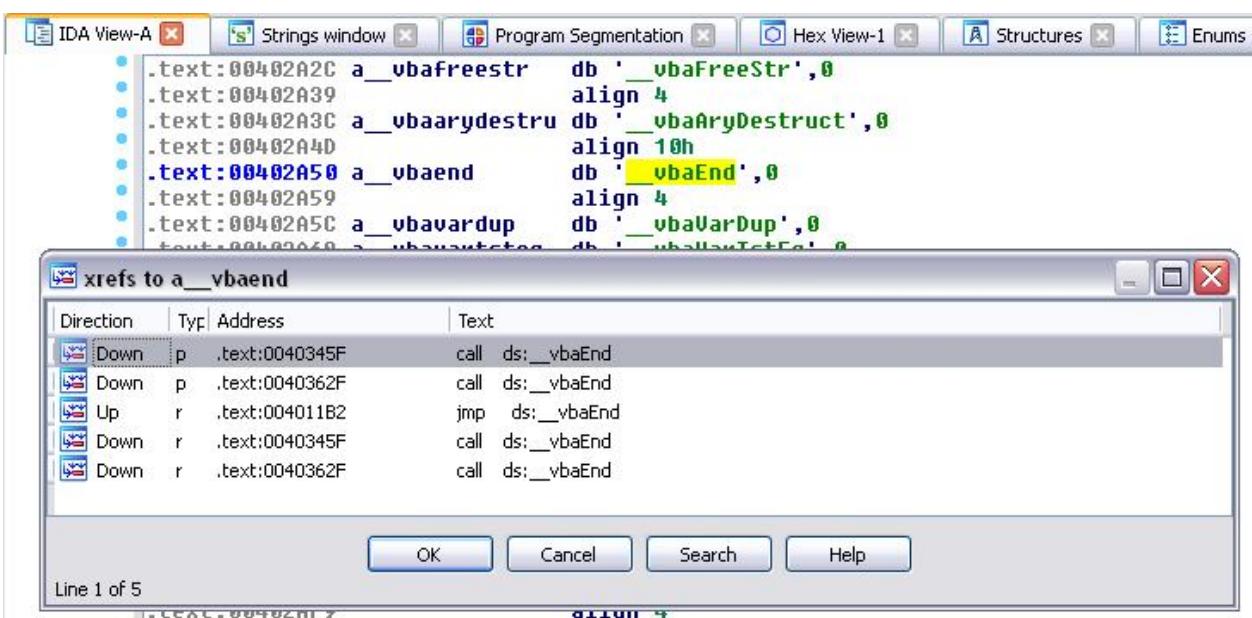
Ahora nos colocamos sobre la misma “_vbaEnd” y se nos enmarcará en amarillo

'S'	.text:00402A3C a__vbaarydestru db '_vbaAryDestruct',0
'S'	.text:00402A4D align 10h
'S'	<u>.text:00402A50 a__vbaend db '_vbaEnd',0</u>
'S'	.text:00402A59 align 4
'S'	.text:00402A5C a__vbavardup db '_vbaVarDup',0
'S'	.text:00402A68 a__vbavartsteq db '_vbaVarTstEq',0

Le damos directamente a la letra “X” de nuestro teclado para que nos muestre las referencias a esta APIs, (o lo que es lo mismo: ruta del menú superior de IDA “Jump-Jump to xref to operand...”)

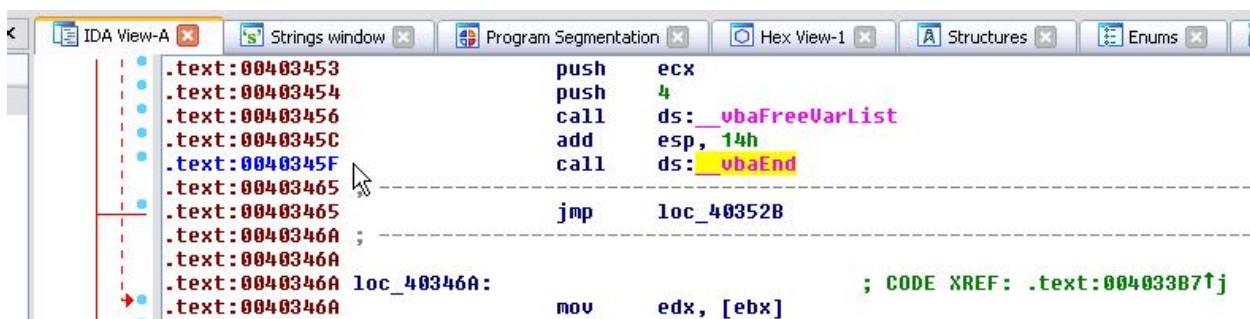


Y nos dice que las referencias son las siguientes:



Cuatro llamadas "call" dos de las cuales nos llevan a la address ".text:0040345F" , las otras dos a la address "0040362F" , y un "jmp"

Con el cursor me posiciono sobre la primera "call " ".text:0040345F", le damos dos clicks Izquierdo de ratón y aparecemos aquí:



Ahora hacemos "scroll" hacia abajo y no veo nada interesante, pero si lo hago hacia arriba observo lo siguiente

```
.text:004033E7      mov    [ebp-0A0h], eax
.text:004033ED      mov    [ebp-90h], eax
.text:004033F3      mov    dword ptr [ebp-0C8h], offset aAccessDenied "Access Denied"
.text:004033FD      mov    [ebp-0D0h], ebx
.text:00403403      call   esi ; __vbaVarDup
.text:00403405      lea    edx, [ebp-0C0h]
.text:0040340B      lea    ecx, [ebp-70h]
.text:0040340E      mov    dword ptr [ebp-0B8h], offset aTooManyAttempt ; "Too many attempts"
.text:00403418      mov    [ebp-0C0h], ebx
.text:0040341E      call   esi ; __vbaVarDup
.text:00403420      lea    eax, [ebp-0A0h]
.text:00403426      lea    ecx, [ebp-90h]
.text:0040342C      push   edx
.text:0040342D      lea    edx, [ebp-80h]
.text:00403430      push   ecx
.text:00403431      push   edx
.text:00403432      lea    eax, [ebp-70h]
.text:00403435      push   10h
.text:00403437      push   eax
.text:00403438      call   ds:rtcMsgBox
.text:0040343E      lea    ecx, [ebp-0A0h]
.text:00403444      lea    edx, [ebp-90h]
.text:0040344A      push   ecx
.text:0040344B      lea    eax, [ebp-80h]
.text:0040344E      push   edx
.text:0040344F      lea    ecx, [ebp-70h]
.text:00403452      push   eax
.text:00403453      push   ecx
.text:00403454      push   4
.text:00403456      call   ds:_vbaFreeVarList
.text:0040345C      add    esp, 14h
.text:0040345F      call   ds:_vbaEnd
```

"Access Denied", por lo que deduzco que es un lugar equivocado, ya que para llegar a este sitio, creo que previamente debería de tener el button "Submit" activado para que el crackme pueda hacer comprobaciones y decidir algo.

Me vuelvo a posicionar sobre la APIs "**_vbaEnd**", y sobre esta le vuelvo a dar a la tecla "X" para que me muestre de nuevo las referencias, y esta vez me coloco sobre la segunda "call"

```
.text:00403454      push   4
.text:00403456      call   ds:_vbaFreeVarList
.text:0040345C      add    esp, 14h
.text:0040345F      call   ds:_vbaEnd
.text:00403465      ---

.text:00403470      xrefs to .idata:0040100C
.text:00403470      Direction Typ Address          Text
.text:00403470      p     .text:0040345F      call  ds:_vbaEnd
.text:00403470      Down   p     .text:0040362F      call  ds:_vbaEnd
.text:00403470      Up    r     .text:004011B2      jmp   ds:_vbaEnd
.text:00403470      r     .text:0040345F      call  ds:_vbaEnd
.text:00403470      Down   r     .text:0040362F      call  ds:_vbaEnd
```

Le damos doble click Izquierdo de ratón y aparecemos en esta zona de código

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library function Data Regular function Unexplored Instruction External symbol

Functions window IDA View-A Strings window Program Segmentation Hex View-1 Structures

Function name ThunRTMain

```
.text:00403626 push eax
.text:00403627 mov [ebp+8], eax
.text:00403628 mov edx, [eax]
.text:0040362C call dword ptr [edx+4]
.text:0040362F call ds:_vbaEnd
.text:00403635 mov dword ptr [ebp-4], 0
.text:0040363C mov eax, [ebp+8]
.text:0040363F push eax
.text:00403640 mov ecx, [eax]
.text:00403642 call dword ptr [ecx+8]
.text:00403642
```

Hacemos de nuevo "scroll" hacia arriba y tampoco encontramos nada interesante. Ahora lo que se me ocurrió es probar a poner un punto de parada "BP" en la address "00403661" ya que veo que por la flecha roja que nos muestra IDA, una llamada que viene de más arriba, salta hasta sobreponer la "_vbaEnd" para continuar con el código.

Pues me posiciono sobre la address "00403661" le doy a "F2" para ponerle el "Break point" y queda así

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

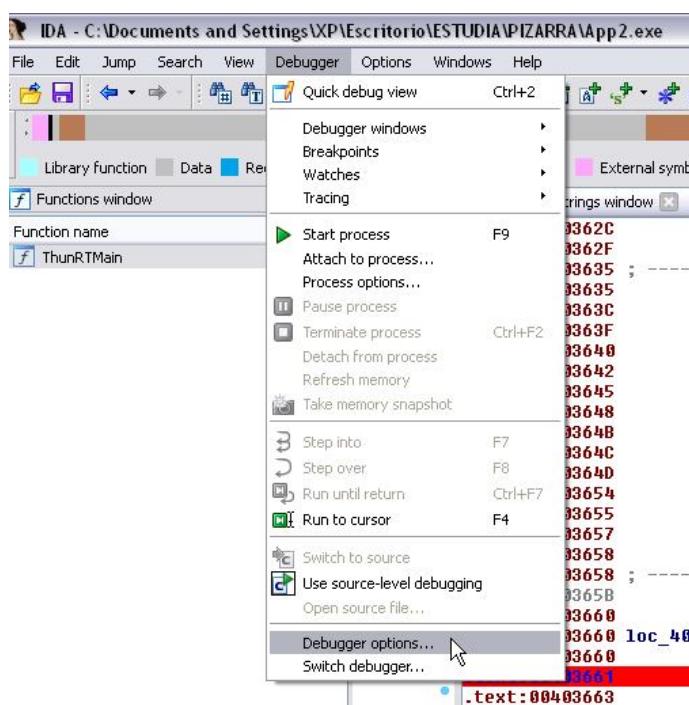
Library function Data Regular function Unexplored Instruction External symbol

Functions window IDA View-A Strings window Program Segmentation Hex View-1 Structures Enums

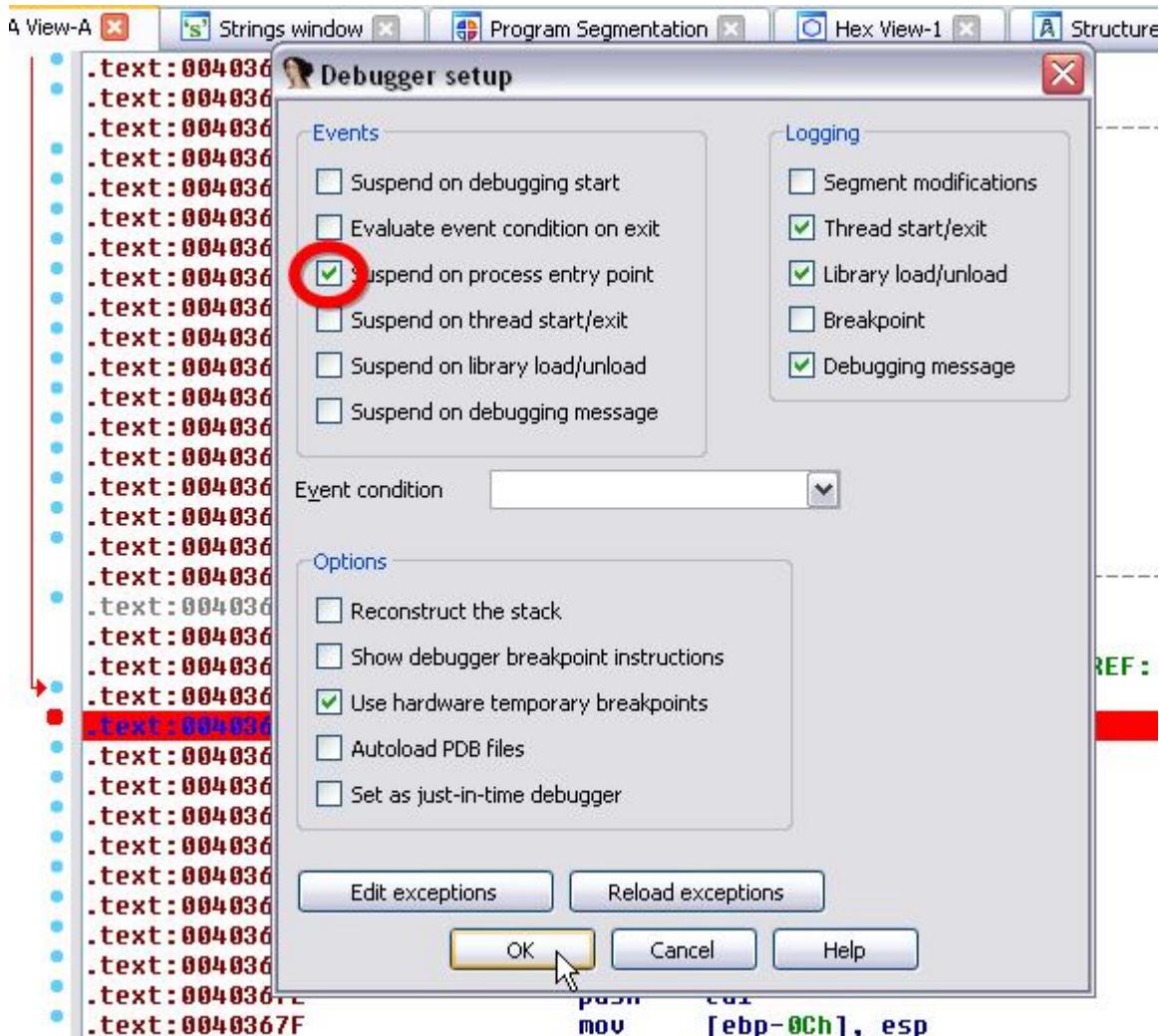
Function name ThunRTMain

```
.text:0040362C call dword ptr [edx+4]
.text:0040362F call ds:_vbaEnd
.text:00403635 ; 
.text:00403635 mov dword ptr [ebp-4], 0
.text:0040363C mov eax, [ebp+8]
.text:0040363F push eax
.text:00403640 mov ecx, [eax]
.text:00403642 call dword ptr [ecx+8]
.text:00403645 mov eax, [ebp-4]
.text:00403648 mov ecx, [ebp-14h]
.text:0040364B pop edi
.text:0040364C pop esi
.text:0040364D mov large fs:0, ecx
.text:00403654 pop ebx
.text:00403655 mov esp, ebp
.text:00403657 pop ebp
.text:00403658 retn 4
.text:00403658 ; 
.text:0040365B align 10h
.text:00403660 loc_403660: ; CODE XREF: .text:004024FB1j
.text:00403661 push ebp
.text:00403661 mov ebp, esp
.text:00403663 sub esp, 0Ch
.text:00403666 push offset loc_401116
.text:0040366B mov eax, large fs:0
```

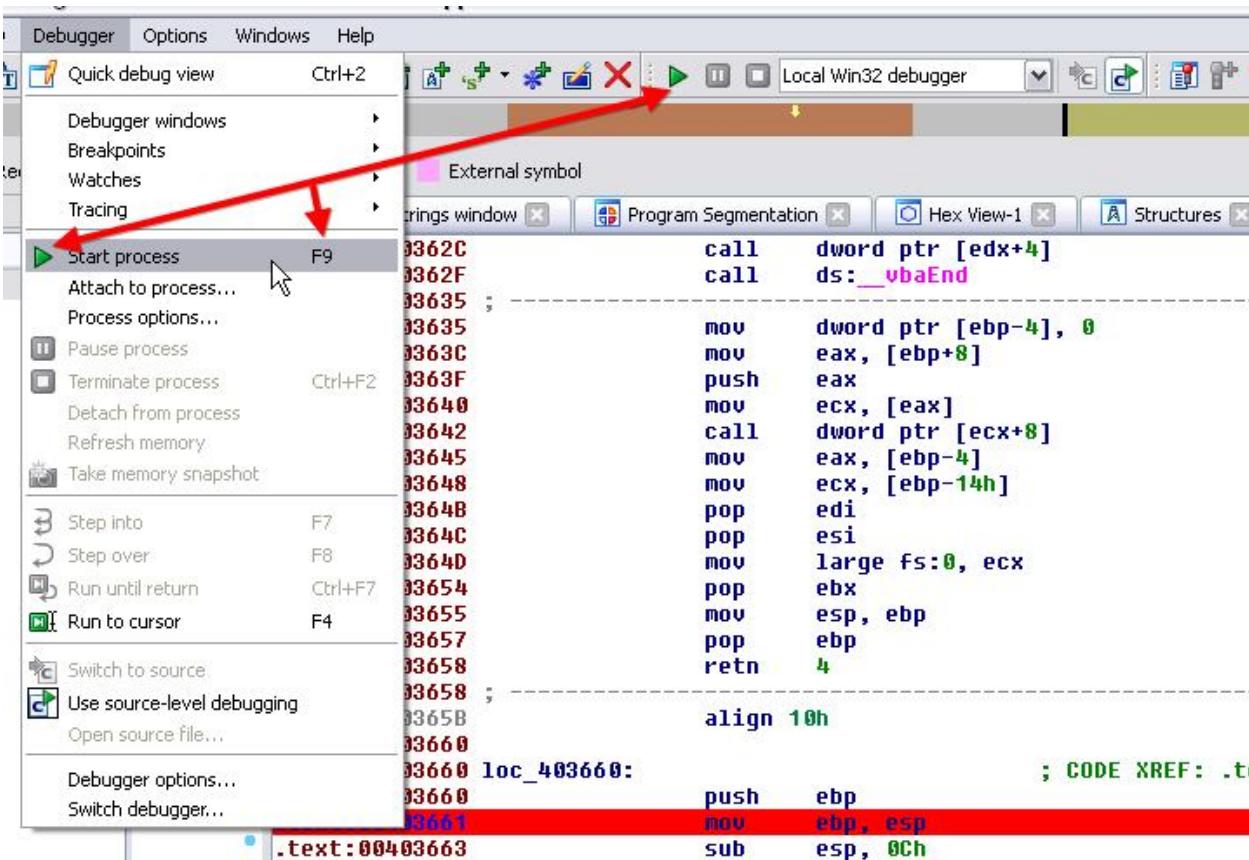
Con nuestro punto de parada colocado, vamos a dejar el modo estático para que el crackme corra en modo debugger, para ello nos vamos a "Debugger y Debugger Options..."



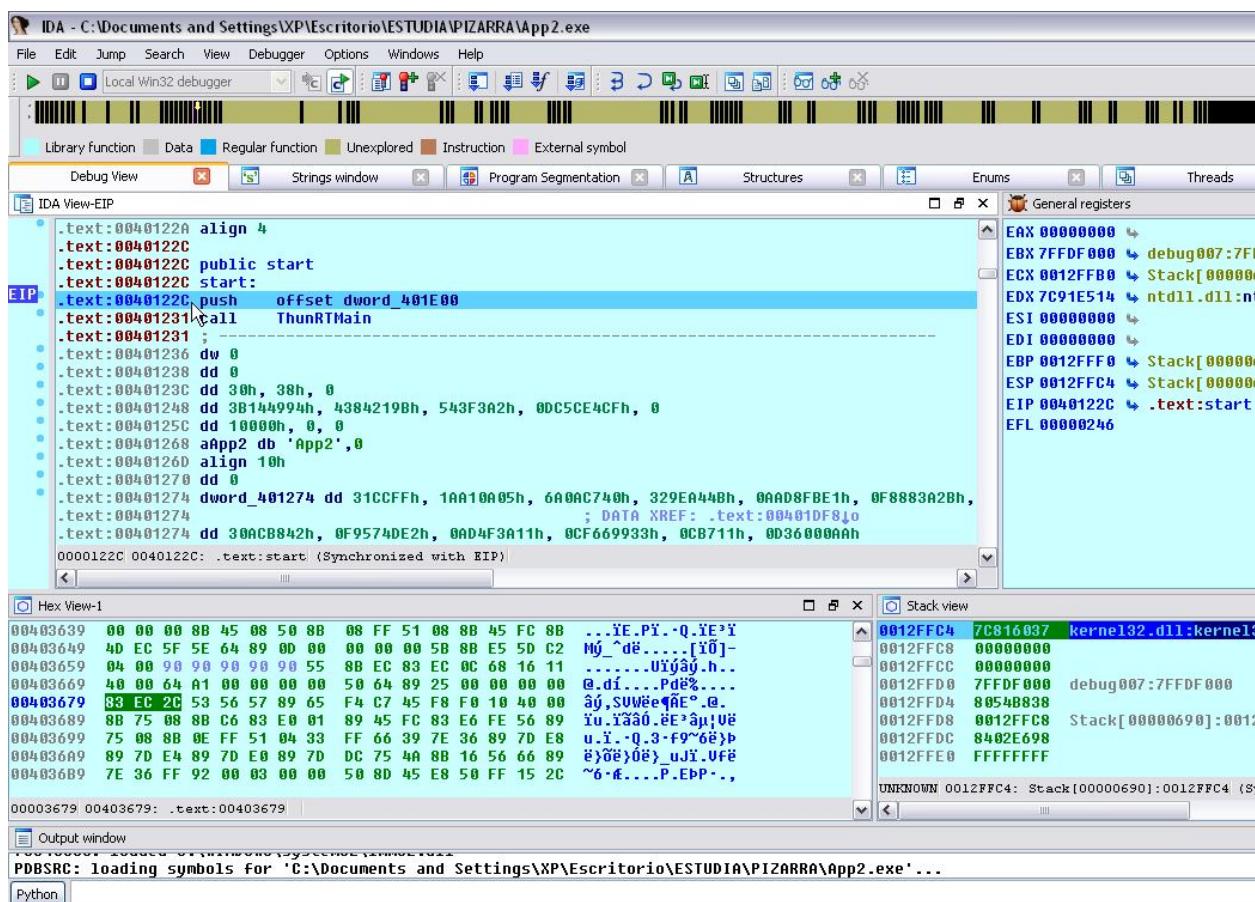
y aquí a mi me gusta tildar siempre la pestaña “Suspend on process entry point” para que cuando arranque el crackme con el debugger, pare en el punto de entrada “EP” o Entry Point, que en este caso es el “OEP” o “Original Entry Point, ya que como nos reveló el detector RDG Packer Detector, no está empacado



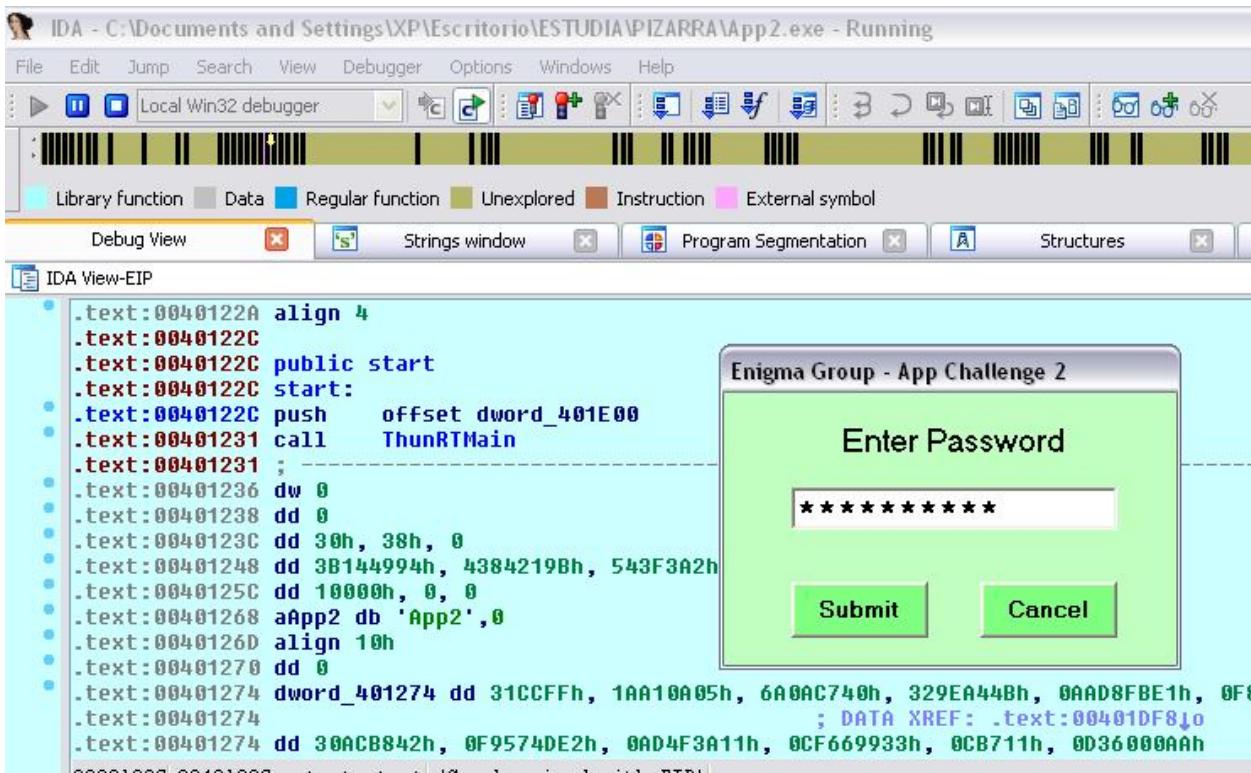
Le damos a “OK”, después a “Start process” o “F9” para que corra el crackme



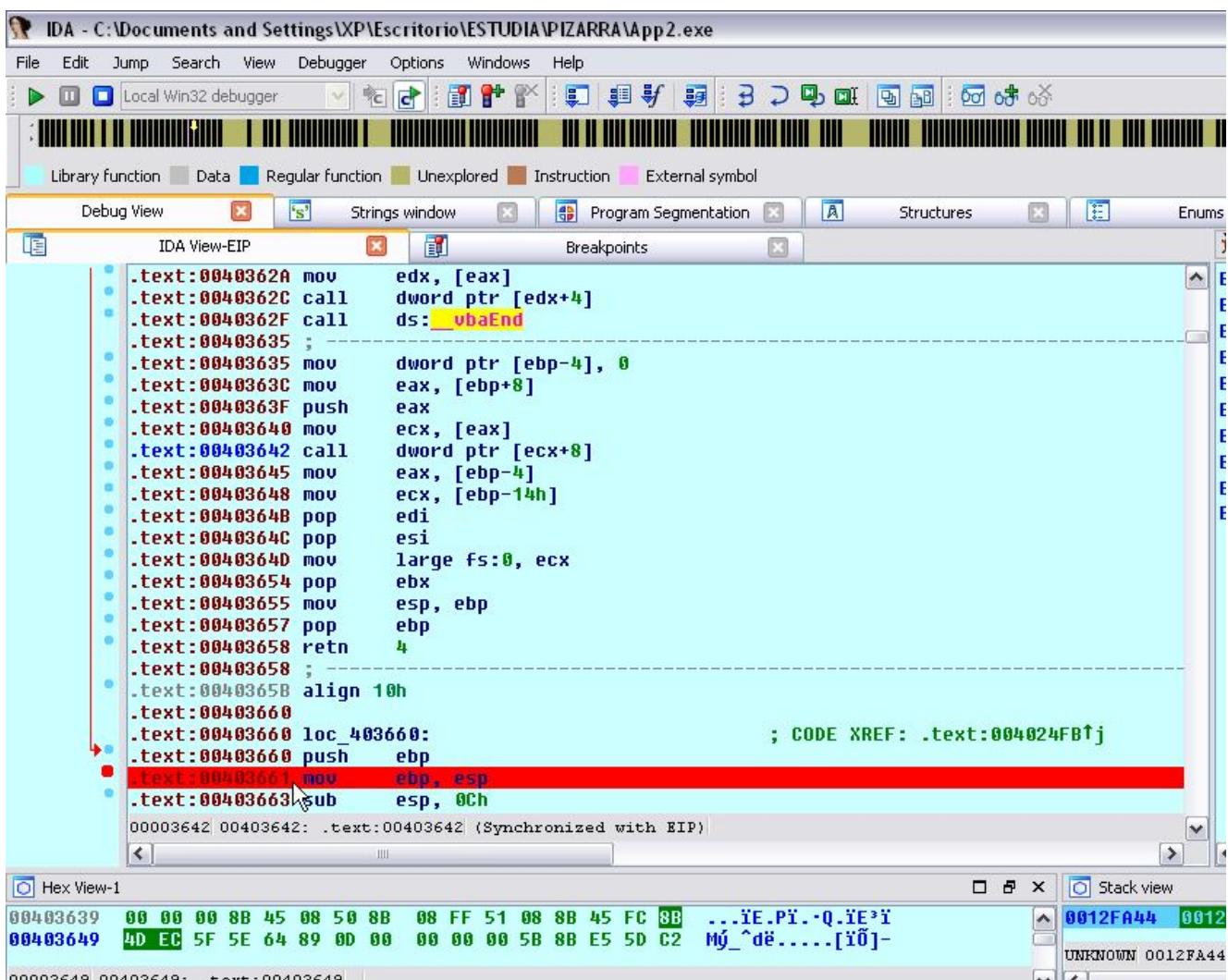
y aparecemos aquí, en el punto de entrada. La misma address donde paró IDA al iniciar el crackme en modo estático. Address “0040122C”



Damos “F9” para que siga corriendo el crackme, tipeamos nuestro Password falso “QwErTy CLS”



y al posicionarnos con el ratón sobre el button "Submit" y una vez más sin posibilidad de clicarlo, IDA nos lleva directos a nuestro "BP" que teníamos puesto



Je,je,je... esto empieza a tener buena pinta, parece que estamos en la zona caliente que estamos buscando.....

Ahora parados en el "BP" vamos traceando con "F8" hasta que en la address ".tex.004036A2" vemos una comparación sospechosa, precedida de un salto condicional ".tex.004036B2"

The screenshot shows the IDA Pro interface with the assembly view open. The EIP register is highlighted with a blue arrow and points to the instruction at address 004036B2, which is a `jnz` (jump if not zero) instruction. The instruction before it is a `cmp` (compare) instruction. Both `cmp` and `jnz` are circled in red.

```
.text:0040368C mov     eax, esi
.text:0040368E and     eax, 1
.text:00403691 mov     [ebp-4], eax
.text:00403694 and     esi, 0FFFFFFFEh
.text:00403697 push    esi
.text:00403698 mov     [ebp+8], esi
.text:0040369B mov     ecx, [esi]
.text:0040369D call    dword ptr [ecx+4]
.text:004036A0 xor     edi, edi
.text:004036A2 cmp     [esi+36h], di
.text:004036A6 mov     [ebp-18h], edi
.text:004036A9 mov     [ebp-1Ch], edi
.text:004036AC mov     [ebp-20h], edi
.text:004036AF mov     [ebp-24h], edi
.text:004036B2 jnz    short loc_4036FE
.text:004036B4 mov     edx, [esi]
.text:004036B6 push   esi
.text:004036B7 mov     [esi+36h], di
.text:004036BB call    dword ptr [edx+300h]
.text:004036C1 push   eax
.text:004036C2 lea    eax, [ebp-18h]
.text:004036C5 push   eax
.text:004036C6 call    ds:_vba0bjSet
.text:004036CC mov     esi, eax
.text:004036CE push   edi
```

000036B2 004036B2: .text:004036B2 (Synchronized with EIP)

Vamos a probar a cambiar el "`jnz`" (Salta si no es 0) por un "`jz`" (Salta si es 0).

Para ello nos posicionamos sobre el "`jnz`"

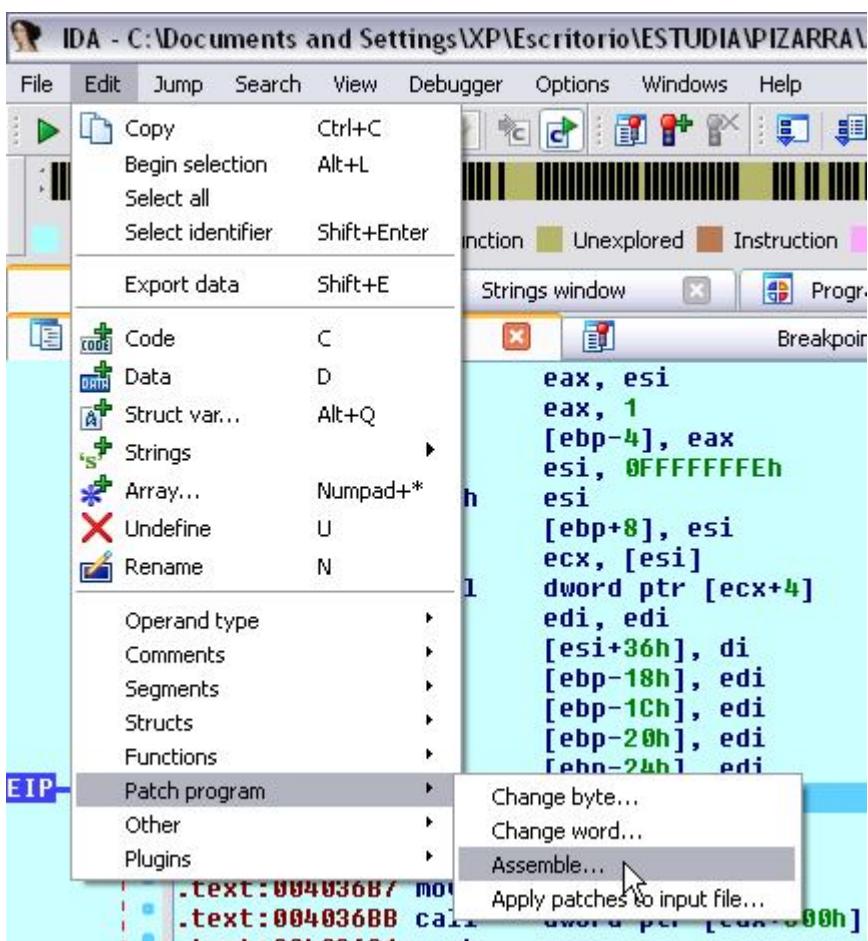
```

.text:00403690 call    dword ptr [ecx+4]
.text:004036A0 xor     edi, edi
.text:004036A2 cmp     [esi+36h], di
.text:004036A6 mov     [ebp-18h], edi
.text:004036A9 mov     [ebp-1Ch], edi
.text:004036AC mov     [ebp-20h], edi
.text:004036AF mov     [ebp-24h], edi
.text:004036B2 jnz    short loc_4036FE
.text:004036B4 mov     edx, [esi]
.text:004036B6 push    esi
.text:004036B7 mov     [esi+36h], di
.text:004036BB call    dword ptr [edx+300h]
.text:004036C1 push    eax
.text:004036C2 lea     eax, [ebp-18h]
.text:004036C5 push    eax
.text:004036C6 call    ds:_vbaObjSet
.text:004036CC mov     esi, eax
.text:004036CE push    edi

```

000036B2 004036B2: .text:004036B2 (Synchronized with EIP)

Ahora "Edit - Patch program - Assemble..."



Y lo cambiamos así,

```

.text:004036A9 mov     [ebp-1Ch], edi
.text:004036AC mov     [ebp-20h], edi
.text:004036AF mov     [ebp-24h], edi
.text:004036B2 jz     short loc_4036FE
.text:004036B4 mov     edx, [esi]
.text:004036B6 push    esi
.text:004036B7 mov     [esi+36h], di
.text:004036BB call    dword ptr [edx+300h]
.text:004036C1 push    eax
.text:004036C2 lea     eax, [ebp-18h]
.text:004036C5 push    eax
.text:004036C6 call    ds:_vbaObjSet
.text:004036CC mov     esi, eax
.text:004036CE push    edi

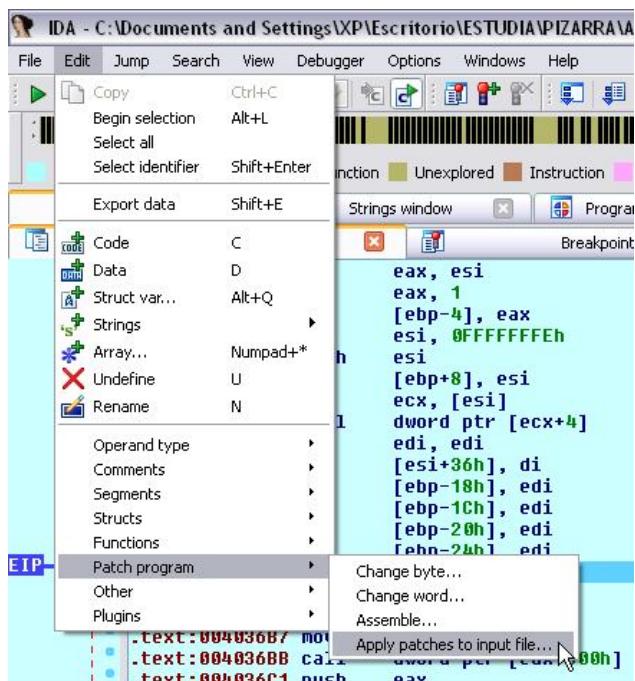
```

000036B2 004036B2: .text:004036B2 (Synchronized with EIP)

Le damos a "OK"

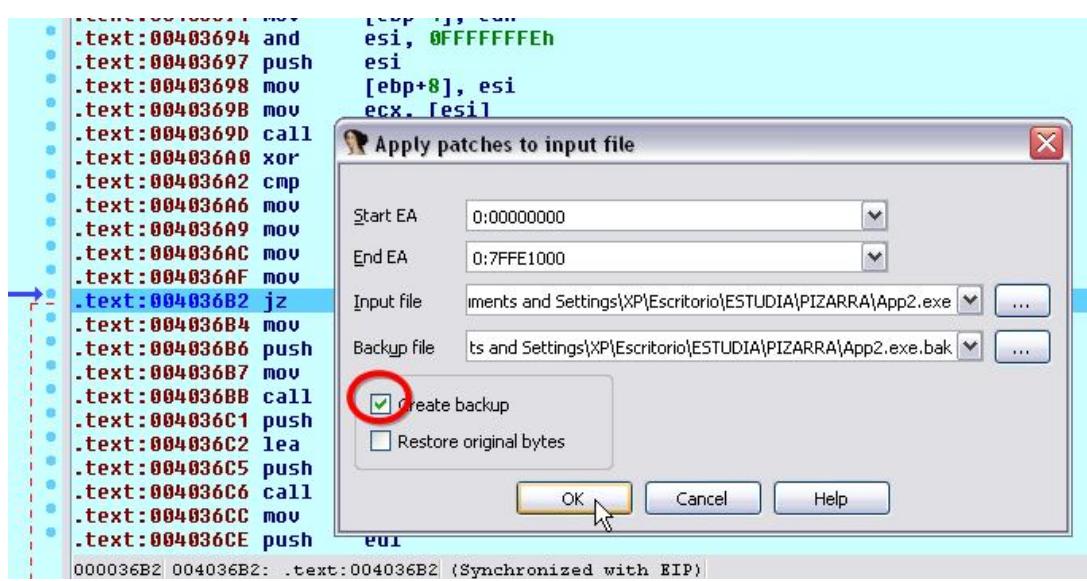
```
.text:004036AC mov     [ebp-20h], edi
.text:004036AF mov     [ebp-24h], edi
IP .text:004036B2 jz      short loc_4036FE
.text:004036B4 mov     edx, [esi]
.text:004036B6 push    esi
.text:004036B7 mov     [esi+36h], di
.text:004036BB call    dword ptr [edx+300h]
.text:004036C1 push    eax
.text:004036C2 lea     eax, [ebp-18h]
```

Guardamos cambios: "Edit - Patch program - Apply patches to input file..."

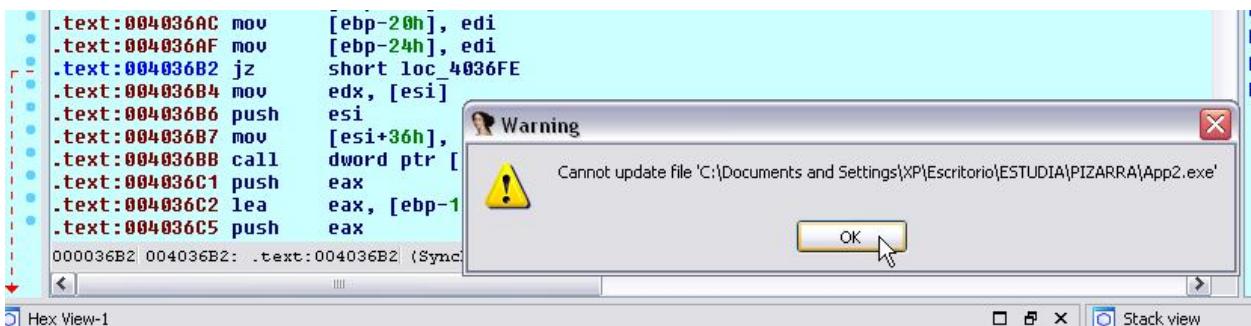


Antes de darle al "OK" , tildamos "Create backup" para que IDA nos cree una copia del crackme.

Lógicamente hacemos una copia por si la modificación que vamos a hacer no es exitosa. De esta forma siempre podremos volver a trabajar con el crackme original.

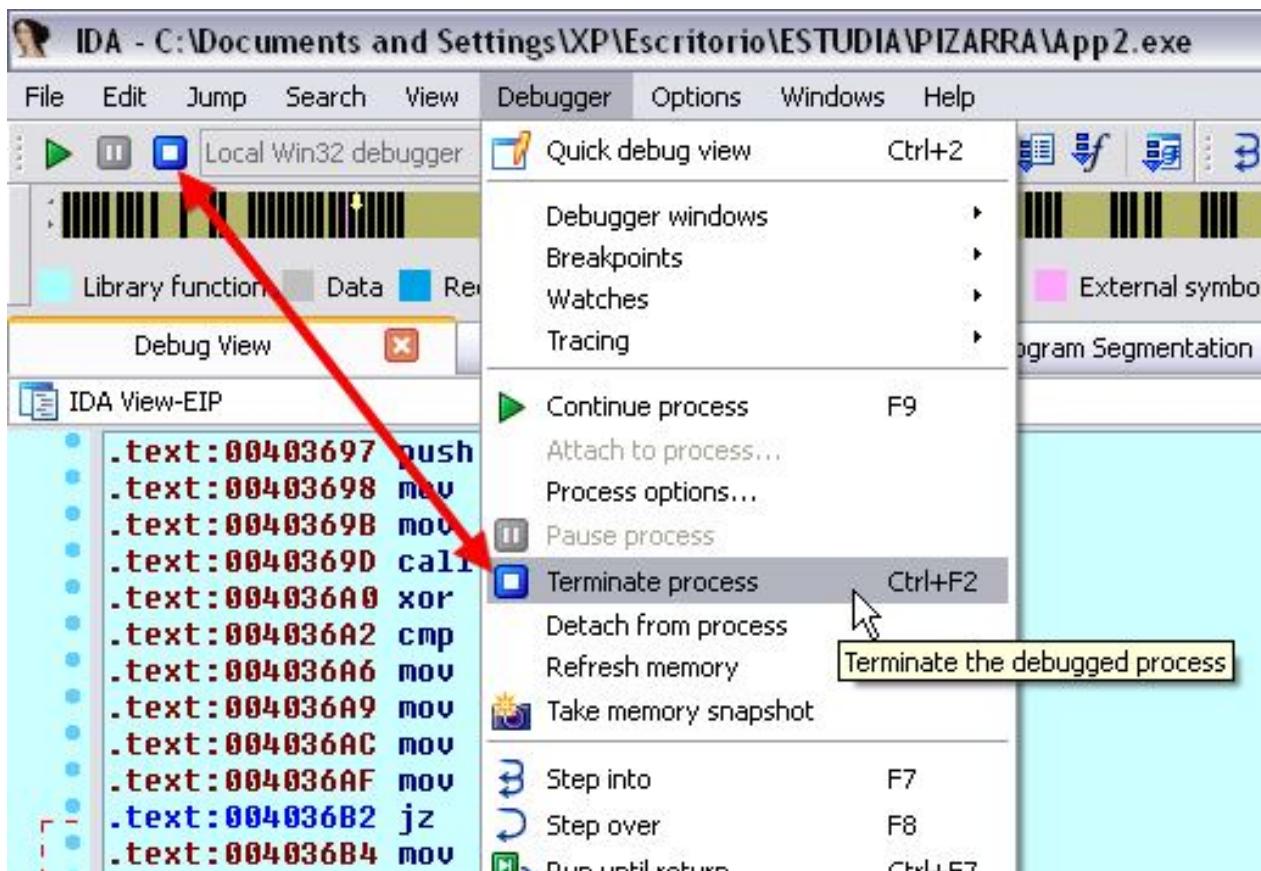


Ahora le damos a "OK", yvaya hombre..... Me sale una ventana de error.... Parece que No me ha dejado guardar el cambio....



Buenoooooo....., que pasa aquí, que cosa más extraña, si solo he cambiado una simple instrucción, ¿Qué habré hecho mal?..... posiblemente éste crackme esté protegido o no deje hacer cambios.

Pues aceptamos el aviso de error, le doy a "OK" salgo del modo debugger dándole a "Terminate process"



y IDA vuelve al modo estático. Una vez aquí me posiciono de nuevo sobre el cambio que hice anteriormente "Jz", y que recordemos que era en la address ".text:004036B2"

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library function Data Regular function Unexplored Instruction External symbol

Functions window IDA View-A Strings window Program Segmentation Hex View-1

Function name ThunRTMain

```

.text:0040368E    and    eax, 1
.text:00403691    mov    [ebp-4], eax
.text:00403694    and    esi, 0FFFFFFFEh
.text:00403697    push   esi
.text:00403698    mov    [ebp+8], esi
.text:0040369B    mov    ecx, [esi]
.text:0040369D    call   dword ptr [ecx+4]
.text:004036A0    xor    edi, edi
.text:004036A2    cmp    [esi+36h], di
.text:004036A6    mov    [ebp-18h], edi
.text:004036A9    mov    [ebp-1Ch], edi
.text:004036AC    mov    [ebp-20h], edi
.text:004036AF    mov    [ebp-24h], edi
.text:004036B2    jz     short loc_4036FE
.text:004036B4    mov    edx, [esi]
.text:004036B6    push   esi

```

Vuelvo a intentar guardar los cambios; “Edit – Patch program – Apply patches to input file...”

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Copy Ctrl+C
Begin selection Alt+L
Select all
Select identifier Shift+Enter
Export data Shift+E

Code C
Data D
Struct var... Alt+Q
Strings
Array... NumPad+*
Undefine U
Rename N

Operand type
Comments
Segments
Structs
Functions
Patch program
Change byte...
Change word...
Assemble...
Apply patches to input file...
Other
Plugins

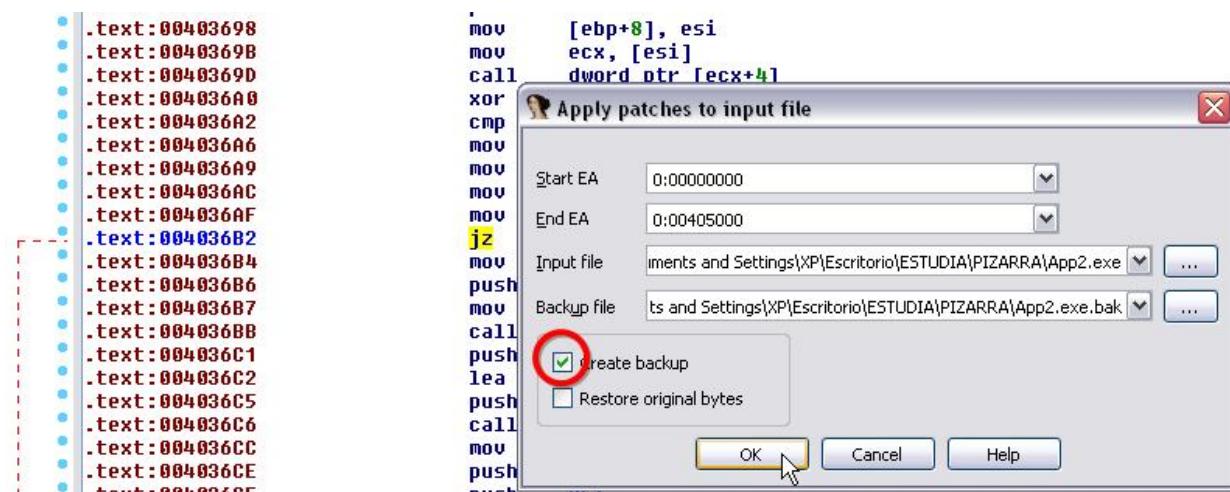
Instructions Unexplored Instruction External symbol

IDA View-A Strings window Program Segmentation Hex View-1

```

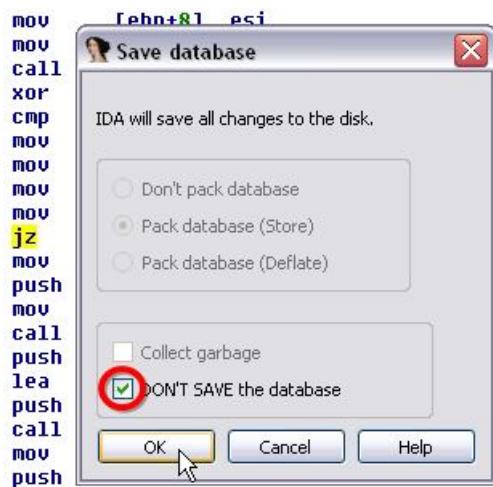
.text:0040368E    and    eax, 1
.text:00403691    mov    [ebp-4], eax
.text:00403694    and    esi, 0FFFFFFFEh
.text:00403697    push   esi
.text:00403698    mov    [ebp+8], esi
.text:0040369B    mov    ecx, [esi]
.text:0040369D    call   dword ptr [ecx+4]
.text:004036A0    xor    edi, edi
.text:004036A2    cmp    [esi+36h], di
.text:004036A6    mov    [ebp-18h], edi
.text:004036A9    mov    [ebp-1Ch], edi
.text:004036AC    mov    [ebp-20h], edi
.text:004036AF    mov    [ebp-24h], edi
.text:004036B2    jz     short loc_4036FE
.text:004036B4    mov    edx, [esi]
.text:004036B6    push   esi
.text:004036B7    mov    [esi+36h], di
.text:004036B8    call   dword ptr [edx+300h]
.text:004036C1    push   eax
.text:004036C2    lea    eax, [ebp-18h]
.text:004036C5    push   eax

```

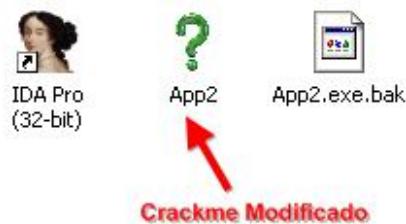


Le doy a "OK" y ahora en estático IDA los ha guardado correctamente, ya que no me ha mostrado ninguna ventana de aviso.

Salgo de IDA para comprobar el resultado, tildo "DON'T SAVE the database" ya que no tengo nada interesante que guardar a parte de la modificación que ya he realizado, y le doy a "OK"



Busco el crackme ahora modificado



Lo ejecuto directamente, tipo un Password cualquiera, le doy al button "Submit"



y..... parece que hemos acertado, ahora está perfectamente activado, y me dice que el Password que he introducido es incorrecto (como era de esperar).

Con el “button” activado damos por finalizada nuestra primera misión, y ya estamos en condiciones de afrontar la segunda.

SEGUNDA MISIÓN - Encontrar el Password

Como detalle, debo decir que con que más adelante tengo la intención de crear un "patch", lo primero que voy a hacer es cambiar el nombre del ejecutable modificado, al que ahora voy a renombrar como "App2a.exe" ya que si deseo recuperar la copia guardada con IDA del crackme original, original y parcheado tengan nombres diferentes evitando así que se sobrescriban.

Y nos queda así:



Continuemos....., vamos a la caza del Password correcto, cargamos con IDA el crackme ahora modificado y renombrado al que hemos llamado "App2a" y aparecemos aquí

nos vamos a la pestaña “Strings window” y ahora nos quedamos con la función “`__vbaVarTstEq`”, que compara o chequea strings o variables.

Nos posicionamos sobre ella

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2a.exe

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

Functions window IDA View-A Strings window Program Segments

Address	Length	Type	String	
'S'	.text:00401...	00000005	C	App2
'S'	.text:00401...	00000005	C	App2
'S'	.text:00401...	00000005	C	App1
'S'	.text:00402...	0000000C	C	frmPassword
'S'	.text:00402...	00000005	C	App2
'S'	.text:00402...	0000000C	C	lblPassword
'S'	.text:00402...	0000000A	C	cmdCancel
'S'	.text:00402...	00000009	C	VBA6.DLL
'S'	.text:00402...	00000013	C	_vbaErrorOverflow
'S'	.text:00402...	0000000D	C	_vbaFreeStr
'S'	.text:00402...	00000011	C	_vbaAryDestruct
'S'	.text:00402...	00000009	C	_vbaEnd
'S'	.text:00402...	0000000C	C	_vbaVarDup
'S'	.text:00402...	0000000E	C	<u>_vbaVarTstEq</u>
'S'	.text:00402...	00000011	C	_vbaFreeStrList
'S'	.text:00402...	0000000C	C	_vbaStrCat
'S'	.text:00402...	0000000D	C	_vbaStrMove

Le damos dos clics Izquierdo de ratón y aparecemos aquí:

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\App2a.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library function Data Regular function Unexplored Instruction External symbol

Functions window IDA View-A Strings window Program Segmentation Hex View-1

.text:00402A4D	a__vbaend	align 10h
.text:00402A50	db '_vbaEnd',0	
.text:00402A59	align 4	
.text:00402A5C	a__vbavardup	db '_vbaVarDup',0
.text:00402A68	a__vbagartsteq	db '_vbaVarTstEq',0
.text:00402A76	align 4	
.text:00402A78	a__vbafreestrli	db '_vbaFreeStrList',0
.text:00402A89	align 4	
.text:00402A8C	a__vbastrcat	db '_vbaStrCat',0
.text:00402A98	a__vbastrmove	db '_vbaStrMove',0
.text:00402AA5	align 4	
.text:00402A88	a__vbafreevarli	db '_vbaFreeVarList',0
.text:00402AB9	align 4	
.text:00402ABC	a__vbafreeobjli	db '_vbaFreeObjList',0

nos colocamos sobre la misma “_vbaVarTstEq”

```

• .text:00402A50 a__vbaend      db '_vbaEnd',0
• .text:00402A59      align 4
• .text:00402A5C a__vbavardup   db '_vbaVarDup',0
• .text:00402A68 a__vbagartsteq db '_vbaVarTstEq',0
• .text:00402A76      align 4
• .text:00402A78 a__vbafreestrli db '_vbaFreeStrList',0
• .text:00402A89      align 4

```

Miramos las referencias a esta APIs, y nos muestra dos llamadas “Call” las cuales van a la misma address “.text:00403209”, y un “jmp”

The screenshot shows the IDA Pro interface with assembly code in the background. A cross-reference dialog is open, titled "xrefs to a__vbavartsteq". It lists three entries:

Direction	Type	Address	Text
Down	p	.text:00403209	call ds:_vbaVarTstEq
Up	r	.text:004011E8	jmp ds:_vbaVarTstEq
Down	r	.text:00403209	call ds:_vbaVarTstEq

Buttons at the bottom of the dialog include OK, Cancel, Search, and Help. Below the dialog, the assembly code continues with ".text:00402B9F align 10h".

Nos posicionamos sobre la "call" que queramos (ya sabemos que cualquiera de las dos nos llevarán al mismo sitio), doble click izquierdo de ratón o le damos directamente al "OK" , y apareceremos en esta parte del código

The screenshot shows the IDA Pro interface with the "Functions window" active, displaying the function "ThunRTMain". The assembly code shown is:

```

.text:004031FE      lea    ecx, [ebp-80h]
.text:00403201      lea    edx, [ebp-8A0h]
.text:00403207      push   ecx
.text:00403208      push   edx
.text:00403209      call   ds:_vbaVarTstEq
.text:0040320F      mov    si, ax
.text:00403212      lea    eax, [ebp-60h]
.text:00403215      lea    ecx, [ebp-5Ch]
.text:00403218      push   eax
.text:00403219      push   ecx
.text:0040321A      push   2

```

Como los crackers somos curiosos por naturaleza.....(si no, no estaríamos leyendo todo este rollo...) hacemos "scroll" hacia arriba y en la address "004030F7" vemos un posible mensaje de chico malo

The screenshot shows the assembly code for the "ThunRTMain" function. A red oval highlights the string "Incorrect Password" located at address 004030F7.

```

.text:004030E5      call   esi ; _vbaStrCopy
.text:004030E7      mov    eax, [ebp-20h]
.text:004030EA      mov    edx, offset dword_4028FC
.text:004030EF      lea    ecx, [eax+0F4h]
.text:004030F5      call   esi ; _vbaStrCopy
.text:004030F7      mov    edx, offset aIncorrectPassw  "Incorrect Password"
.text:004030FC      lea    ecx, [ebp-34h]
.text:004030FF      call   esi ; _vbaStrCopy
.text:00403101      mov    ecx, [ebx]
.push   ebx

```

Y si hacemos "scroll" hacia abajo, en la address "00403274" encontramos el posible mensaje de chico bueno

```

.text:00403265    mov    ebx, ds:_vbaStrCat
.text:00403268    mov    [ebp-11Ch], eax
.text:00403271    mov    eax, [edx+28h]
.text:00403274    push   offset aCongratulation
.text:00403279    push   eax
.text:0040327A    call   ebx ; _vbaStrCat
.text:0040327C    mov    edx, eax
.text:0040327E    lea    ecx, [ebp-38h]
.text:00403281    call   ds:_vbaStrMove

```

"Congratulations! The password is "

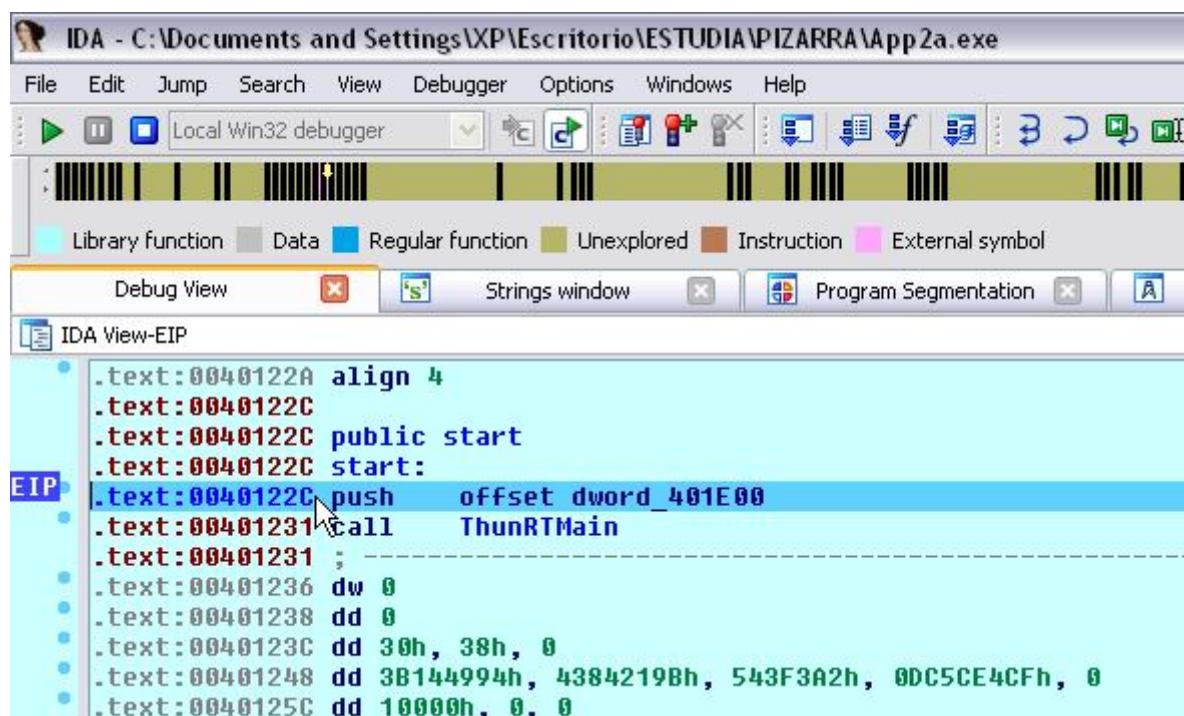
Pues, parece que estamos cerca de la zona caliente, volvemos a hacer "scroll" hacia arriba y ponemos un "BP" en la "call" address ".tex:00403209" misma APIs "_vbaVarTstEq"

```

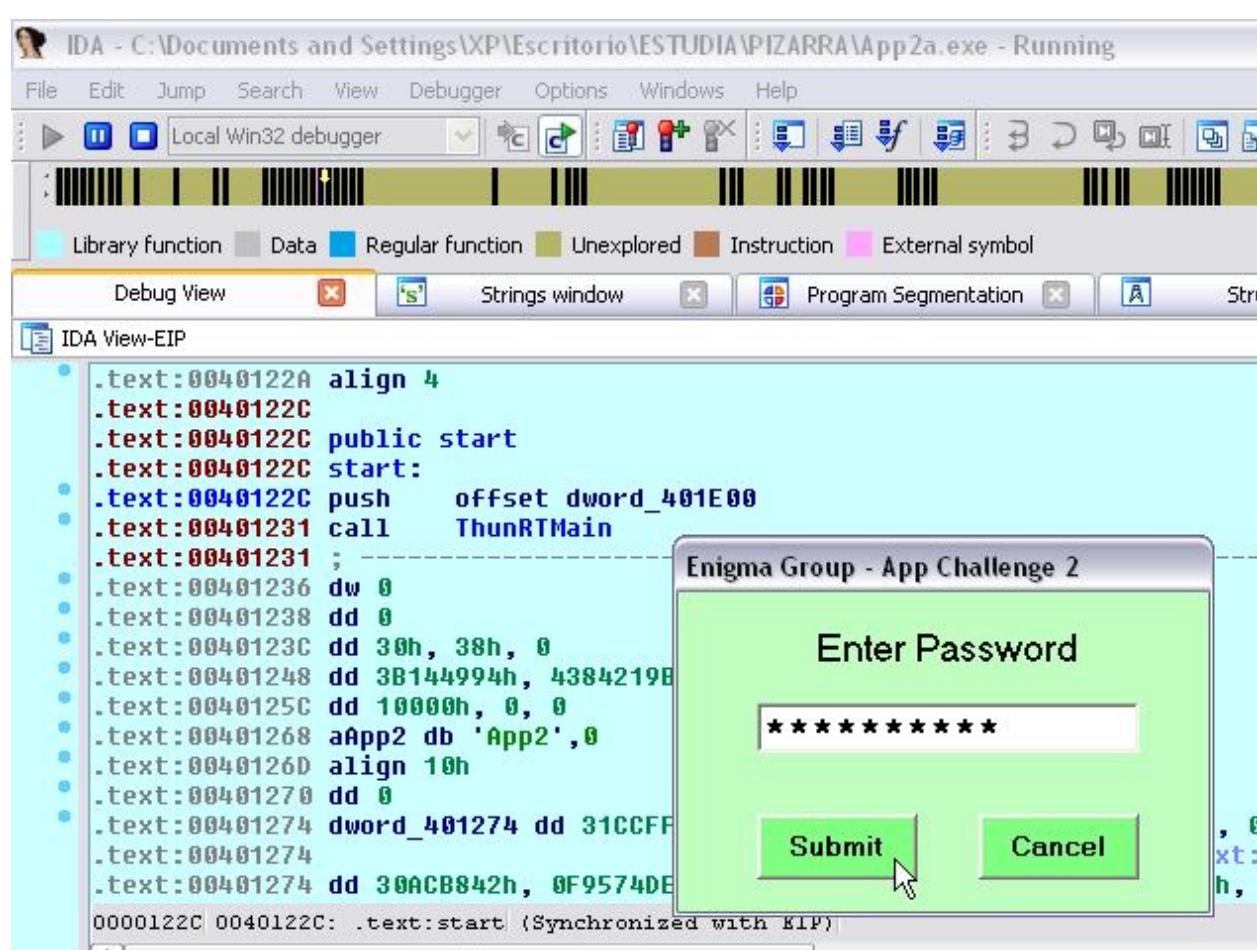
.text:004031EA    push   eax
.text:004031EB    mov    [ebp-3Ch], edi
.text:004031EE    mov    dword ptr [ebp-90h], 8
.text:004031F8    call   ds:rtcUpperCaseVar
.text:004031FE    lea    ecx, [ebp-80h]
.text:00403201    lea    edx, [ebp-0A0h]
.text:00403207    push   ecx
.text:00403208    push   edx
.text:00403209    call   ds:_vbaVarTstEq
.text:0040320F    mov    si, ax
.text:00403212    lea    eax, [ebp-60h]
.text:00403215    lea    ecx, [ebp-5Ch]
.text:00403218    push   eax
.text:00403219    push   ecx
.text:0040321A    push   2
.text:0040321C    call   ds:_vbaFreeObjList

```

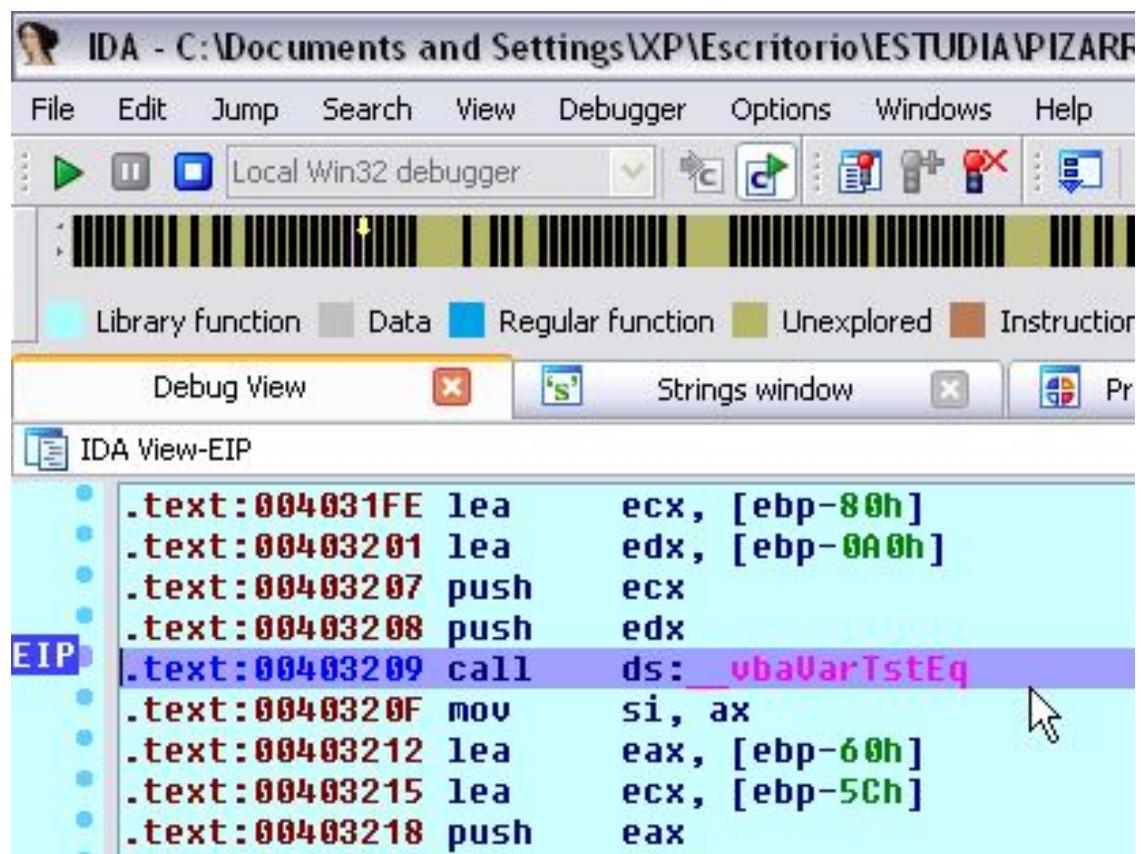
Corremos el crackme con el debugger y aparecemos en el "OEP", (ya que yo volví a tildar la pestaña "Suspend on process entry point" en "debugger - debugger options...", y aparecemos aquí)



Le damos a para que siga corriendo el crackme, este se abre, tipeamos nuestro Password falso "QwErTy CLS"



Le damos al button "**Submit**" y **IDA** para en nuestro "**Break point**"



Entramos con "F7" en la "Call" address "00403209", continuamos traceando con "F7" hasta encontrar una segunda "call" address "734A9800" a la que también entramos con "F7"

```
msvbvm60.dll:734A97F6 ; 
msvbvm60.dll:734A97F6 
msvbvm60.dll:734A97F6 msvbvm60__vbaVarTstEq:
msvbvm60.dll:734A97F6 push    dword ptr [esp+8]
msvbvm60.dll:734A97FA push    dword ptr [esp+8]
msvbvm60.dll:734A97FE push    0
EIP msvbvm60.dll:734A9800 call    near ptr unk_734A9656
msvbvm60.dll:734A9805 mov     eax, dword_733CE644[eax*4]
msvbvm60.dll:734A980C retn    8
msvbvm60.dll:734A980C ; 
msvbvm60.dll:734A980F msvbvm60__vbaVarTstNe db 0FFh
msvbvm60.dll:734A9810 db 74h ; t
```

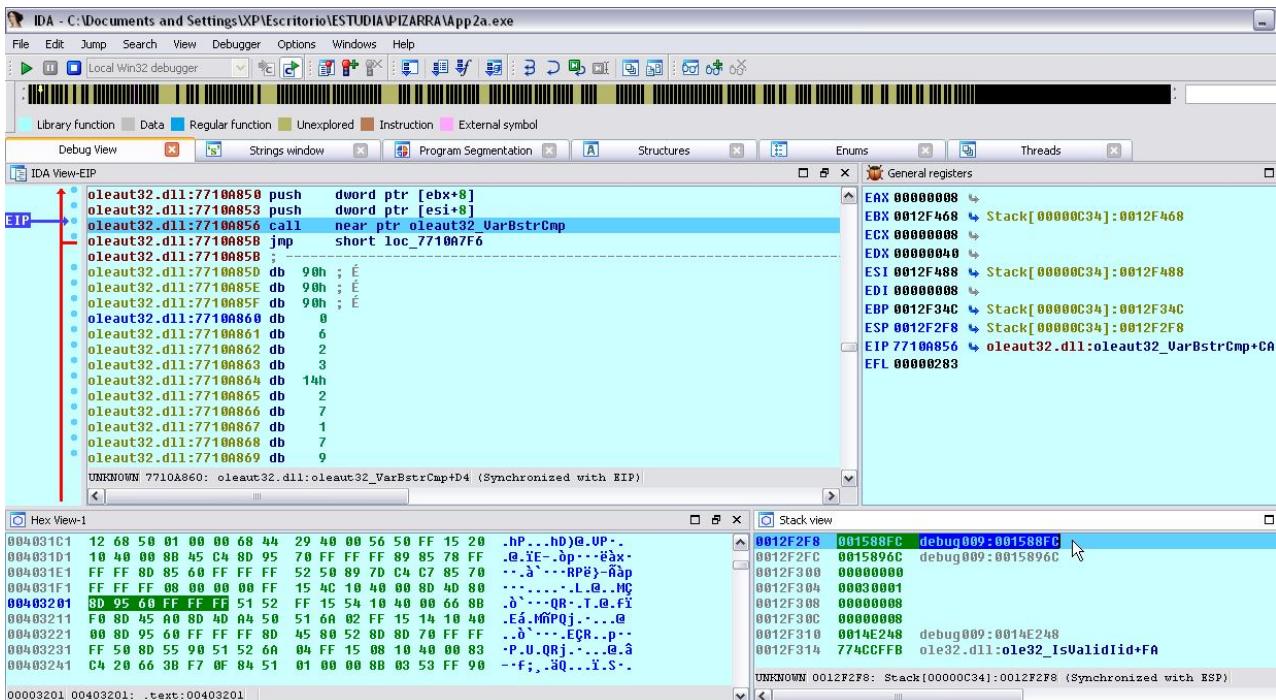
Seguimos traceando y vemos otra tercera "Call" address "734A968C" que también entraremos con "F7"

```
msvbvm60.dll:734A9678 jz     short loc_734A96AC
msvbvm60.dll:734A967A cmp    si, 9
msvbvm60.dll:734A967E jz     short loc_734A96AC
msvbvm60.dll:734A9680 movzx  eax, word ptr [ebp+8]
msvbvm60.dll:734A9684 push   30001h
msvbvm60.dll:734A9689 push   eax
msvbvm60.dll:734A968A push   ecx
msvbvm60.dll:734A968B push   edx
EIP msvbvm60.dll:734A968C call   off_734AEF24
msvbvm60.dll:734A9692 mov    [ebp+8], eax
msvbvm60.dll:734A9695 loc_734A9695:
```

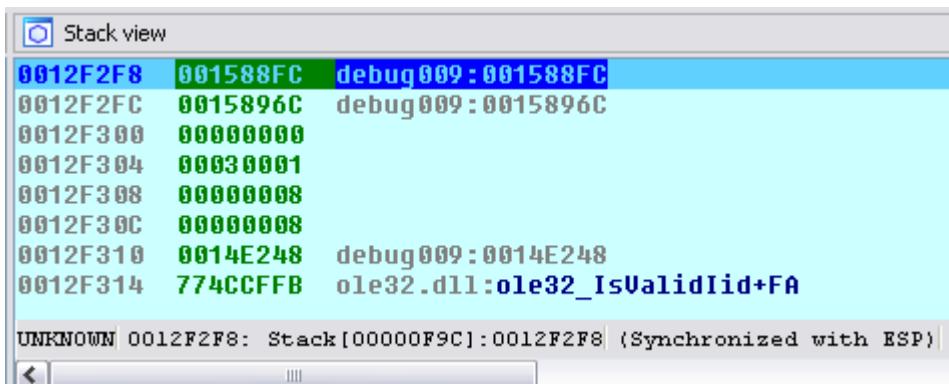
Y continuamos traceando un poco más hasta llegar a la "call" "7710A856" donde IDA nos muestra algo interesante para nuestro propósito "_VarBstrCmp"

```
oleaut32.dll:7710A84A push   dword ptr [ebp+14h]
oleaut32.dll:7710A84D push   dword ptr [ebp+10h]
oleaut32.dll:7710A850 push   dword ptr [ebx+8]
oleaut32.dll:7710A853 push   dword ptr [esi+8]
EIP oleaut32.dll:7710A856 call   near ptr oleaut32__VarBstrCmp
oleaut32.dll:7710A85B jmp    short loc_7710A7F0
oleaut32.dll:7710A85B ;
oleaut32.dll:7710A85D db 90h ; É
```

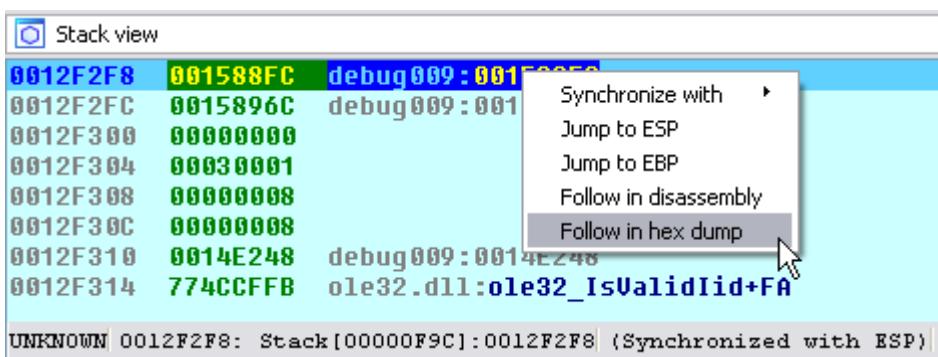
Aquí parados, nos vamos a la ventana "Stack"



Y observamos dos "debug00.:.....",



nos posicionamos sobre el primero, que es "001588FC debug009:001588FC", clic derecho de ratón y "Follow in hex dump" para que nos muestre en el la ventana "Dump" lo que lleva escondido dentro



Y vemos nuestro Password trucho que tipeamos

Hex View-1
001588BC BC 07 18 00 14 00 00 00 51 00 77 00 45 00 72 00 +Q.w.E.r.
001588CC 54 00 79 00 20 00 43 00 4C 00 53 00 00 00 AD BA T.y...C.L.S...;!
001588DC 0D F0 AD BA AB AB AB AB AB AB AB AB 00 00 00 00 -i!zzzzzzzzzzz-
001588EC 00 00 00 00 07 00 07 00 B5 07 18 00 14 00 00 00
001588FC 51 00 57 00 45 00 52 00 54 00 59 00 20 00 43 00 Q.W.E.R.T.Y...C.
0015890C 4C 00 53 00 00 00 AD BA 0D F0 AD BA AB AB AB AB L.S...;!i!zzzz
0015891C AB AB AB AB 00 00 00 00 00 00 00 00 00 07 00 07 00 zzzzzz
0015892C 8E 07 18 00 0C 00 00 00 54 00 6F 00 70 00 47 00 Ä.....T.o.p.G.
0015893C 75 00 6E 00 00 00 AD BA 0D F0 AD BA 0D F0 AD BA u.n...;!i!i!i

Ahora hacemos la misma operación sobre el segundo “debug00....”

0015896C debug009:0015896C ", click derecho de ratón y "**Follow in hex dump**" y vemos esto

	AB	AB	AB	AB	00	00	00	00	00	00	00	00	00	00	00	00	00	07	00	07	00	14444
0015891C	AB	AB	AB	AB	00	00	00	00	00	00	00	00	00	00	00	00	00	07	00	07	00	14444
0015892C	8E	07	18	00	0C	00	00	00	54	00	6F	00	70	00	47	00	AAT.o.p.G.				
0015893C	75	00	6E	00	00	00	AD	BA	0D	F0	AD	BA	0D	F0	AD	BA	u.n.... . . .					
0015894C	0D	F0	AD	BA	AB	- . 4444444444444444																
0015895C	0C	00	00	00	07	00	07	00	87	07	18	00	0C	00	00	00	00	00	00	00	00	
0015896C	54	00	4F	00	50	00	47	00	55	00	4E	00	00	00	00	00	00	00	00	00	00	
0015897C	0D	F0	AD	BA	0D	F0	AD	BA	0D	F0	AD	BA	AB	AB	AB	AB	T.O.P.G.U.N..					
0015898C	AB	AB	AB	AB	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	07	00	2222222222222222
0015899C	EE	04	EE	00	90	DB	15	00	78	01	14	00	EE	FE	EE	FE	EE	00	00	00	00	

Osea que "TopGun" o "TOPGUN" podría ser el Password que estamos buscando.

Vamos a comprobarlo, pero primero vamos a poner otro "Breakpoint" a esta "call" address "**7710A856**" donde actualmente estamos parados posicionándonos sobre ella

oleaut32.dll:7710A84A ; --
oleaut32.dll:7710A84A
oleaut32.dll:7710A84A loc_7710A84A:
oleaut32.dll:7710A84A
oleaut32.dll:7710A84A push dword ptr [ebp+14h]
oleaut32.dll:7710A84D push dword ptr [ebp+10h]
oleaut32.dll:7710A850 push dword ptr [ebx+8]
oleaut32.dll:7710A853 push dword ptr [esi+8]
oleaut32.dll:7710A856 call near ptr oleaut32_VarBstrCmp
oleaut32.dll:7710A85B jmp short loc_7710A7F6
oleaut32.dll:7710A85B ; --
oleaut32.dll:7710A85D db 90h ; É

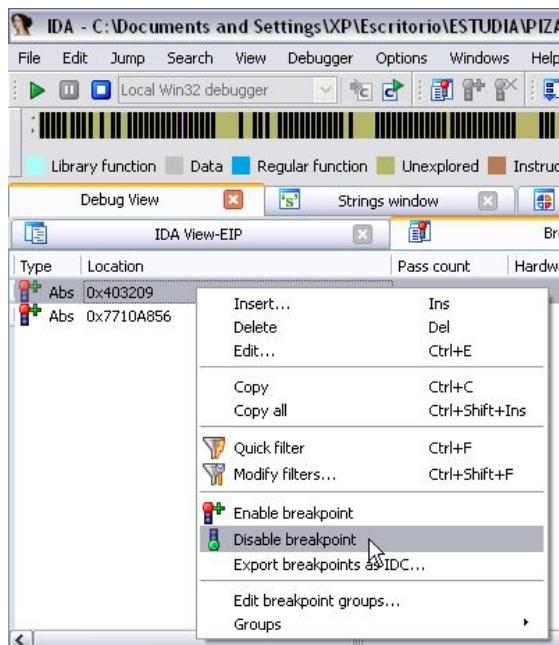
Y dándole a



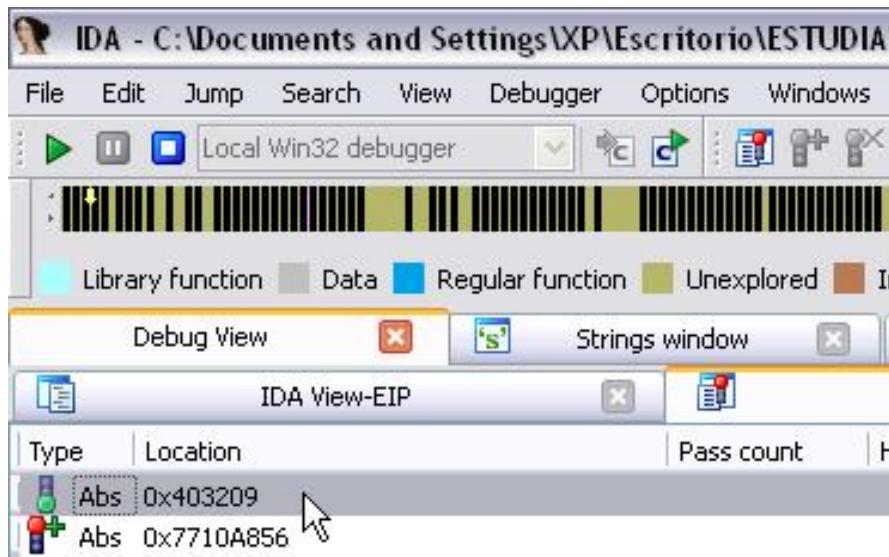
Ahora entramos en la lista de "Breakpoints" que tenemos colocados



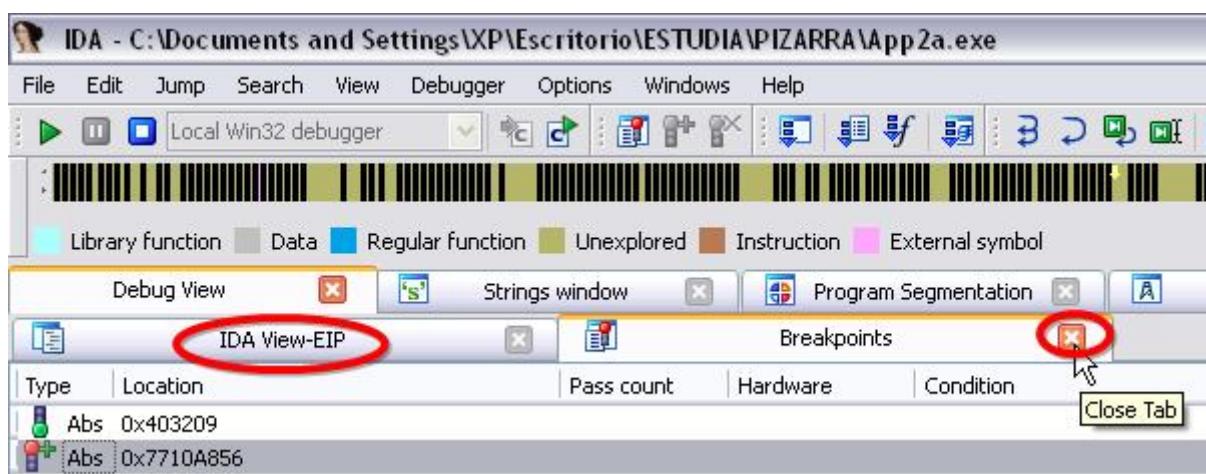
Y vamos a deshabilitar el anterior que teníamos puesto posicionándonos sobre él, click derecho de ratón y “Disable breakpoint”



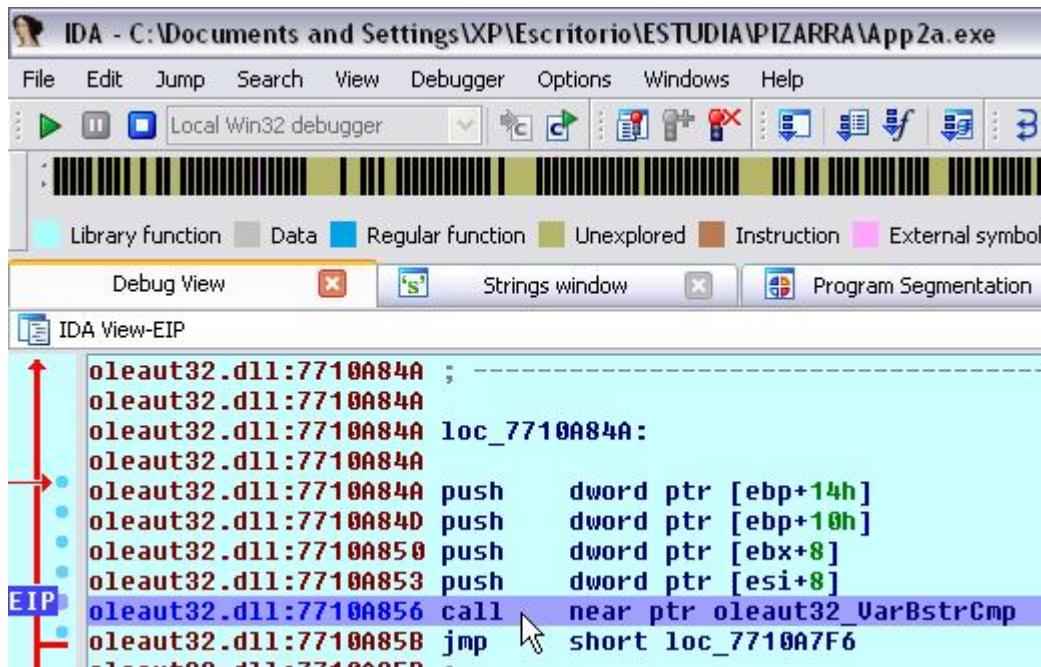
Y nos queda así



Volvemos a la vista normal dándole a “IDA View-EIP”, o bien cerrando la ventana “Breakpoints”



Y estamos de nuevo en la ventana principal



Damos ▶ para que finalice el crackme, y una vez más, como era de esperar, pero que ya empieza a mosquearnos, nos salta el mensaje de chico malo.



En esta misma pantalla, vamos a entrar nuestro posible Password "TopGun" que obtuvimos anteriormente y le damos a "Submit"



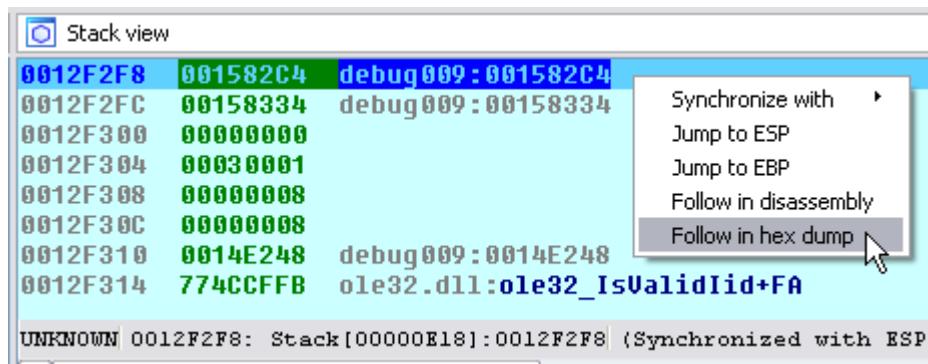
IDA para en nuestro único “breakpoint” que hemos dejado habilitado

The screenshot shows the assembly view in IDA Pro. A red arrow points from the left margin to the instruction at address `oleaut32.dll:7710A85B`, which is highlighted in purple. The instruction is `jmp short loc_7710A7F6`. The stack frame is visible on the left, and the assembly code is listed below it.

```
0012F2F8 001582C4 debug009:001582C4
0012F2FC 00158334 debug009:00158334
0012F300 00000000
0012F304 00030001
0012F308 00000008
0012F30C 00000008
0012F310 0014E248 debug009:0014E248
0012F314 774CCFFB ole32.dll:ole32_IsValidIid+FA

UNKNOWN 0012F2F8: Stack[00000E18]:0012F2F8 (Synchronized with ESP)
```

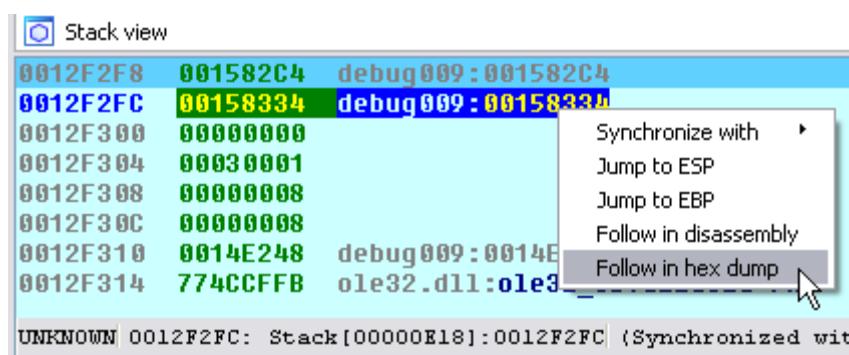
Nos dirigimos a la ventana “Stack”, nos posicionamos sobre el primer “debug009:001582C4” y “Follow in hex dump”



Y en la ventana “Dump” nos muestra el Password introducido



Volvemos a dirigirnos a la ventana “Stack” y repetimos la misma operación con el segundo “debug009:00158334” y “Follow in hex dump”



Y en la ventana “Dump” nos muestra el Password que comparará con el introducido. .je,je,je

```

00 <.....T.o.p.G.
BA u.n...i|i|i|
00 .i|zzzzzzzz...zzz
00 -----
BA T.O.P.G.U.N...i|
BA .i|i|i|i|zzzz
00 zzz...>
FE -.-h--h-->
FE i|i|i|i|i|i|i

```

Ya solo nos queda seguir traceando para ir viendo como va avanzando el código hasta que llegue al final (AVISO: si elegimos esta opción del traceo probablemente nos quedaremos dormidos y nuestra cabeza terminará apoyada en el teclado, ya que el proceso es larguísimo), si queremos ir más rápidos le damos a "start" ➤, y.....



Segunda misión solucionada.

Pero la curiosidad me invade, y sin salir de la pantalla, con el cursor del ratón borro los asteriscos, introduzco el Password "TOPGUN" (todo con mayúsculas) y que IDA también me mostró, y a ver que pasa. Le damos a "Submit", IDA para de nuevo en el "BP", le doy directamente a "start" ➤ y también lo acepta.

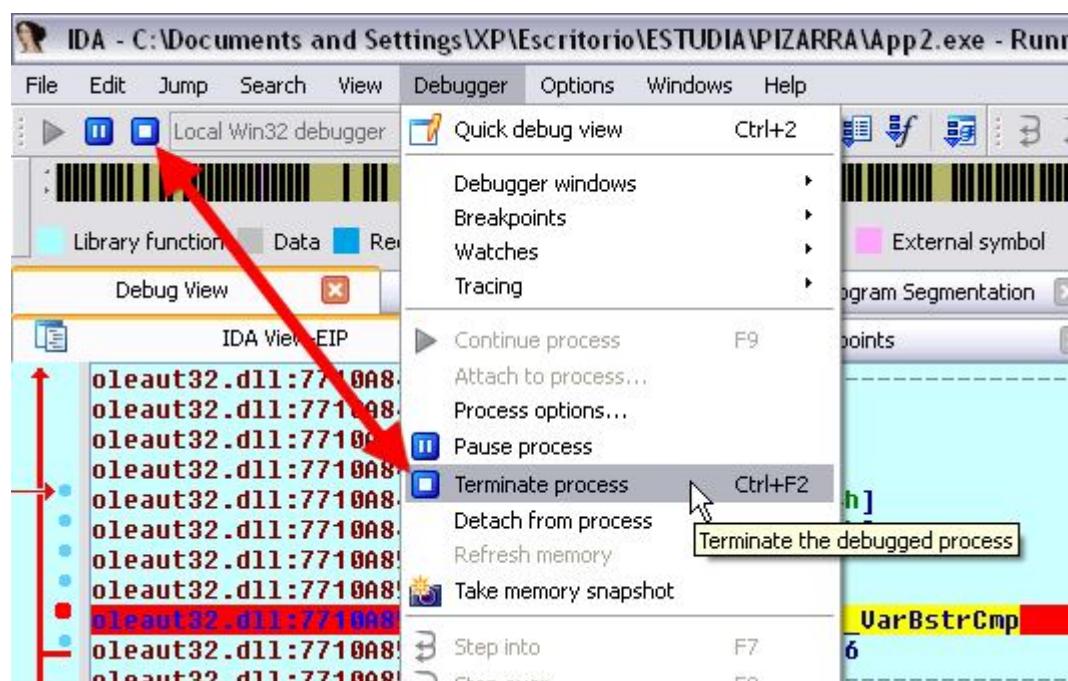


Y ahora lo intento poniendo el Password con mayúsculas y minúsculas aleatoriamente como nos venga en gana "tOpgUn" y.....parece que también va perfecto

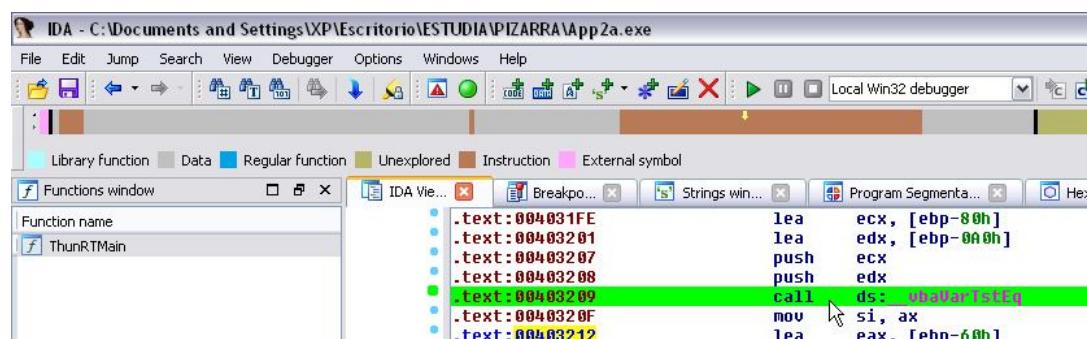


Osea que la Password correcta es la palabra "topgun" indiferentemente de que la tipeemos toda con mayúsculas, minúsculas o alternando ambas, la única condición es que debe respetarse su orden.

Ahora salimos del modo debugger dándole a "Terminate process"



Y aparecemos en el modo estático justo donde tenemos el "BP" deshabilitado (por eso está resaltado en verde)



Si hacemos "scroll" hacia abajo vemos una comparación "cmp" precedida de un salto condicional "jz" je,je,je que tiene toda la pinta de ser el decisivo.

```

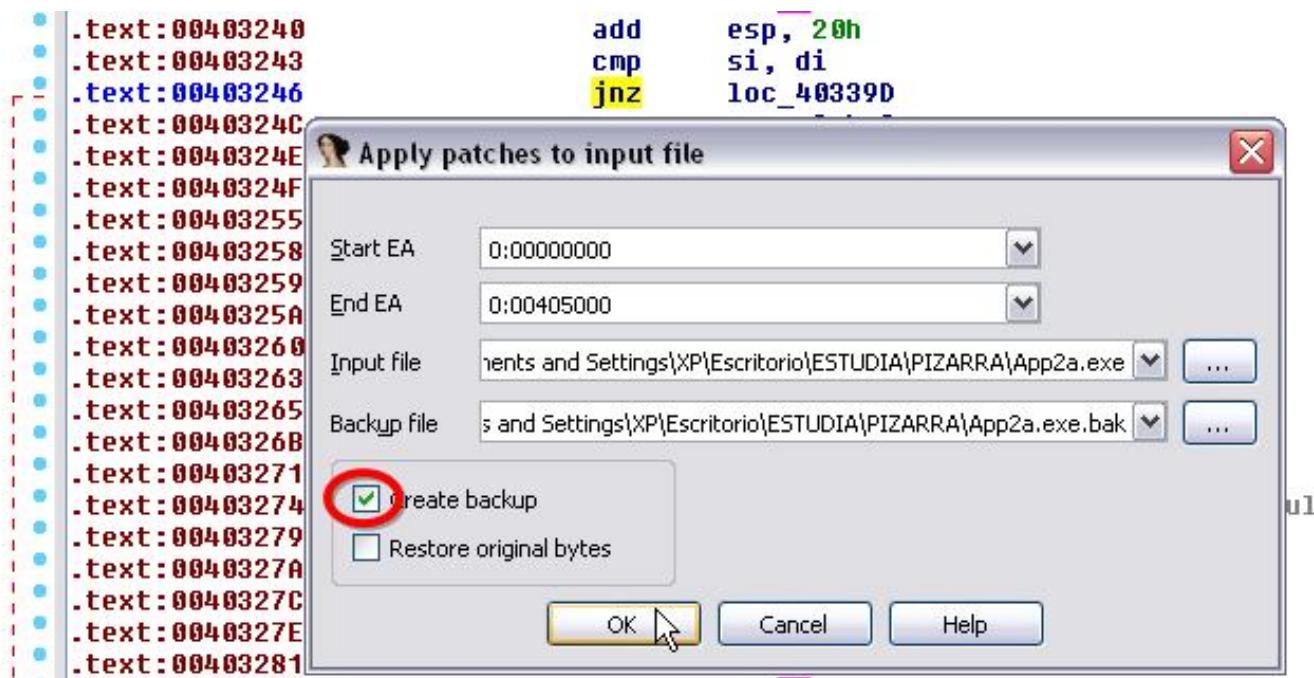
.text:00403243
.text:00403246
.text:0040324C
.text:0040324E
.text:0040324F
.text:00403255
.text:00403259
.text:0040325A
.text:00403260
.text:00403263
.text:00403265
.text:0040326B
.text:00403271
.text:00403274
.text:00403279
.text:0040327A
.text:0040327C
.text:0040327E
.text:00403281
.text:00403287
.text:0040328A

    cmp    si, di
    jz    loc_40339D
    mov    eax, [ebx]
    push   ebx
    call   dword ptr [eax+308h]
    lea    ecx, [ebp 5Ch]
    push   eax
    push   ecx
    call   ds:_vbaObjSet
    mov    edx, [ebp-20h]
    mov    esi, [eax]
    mov    ebx, ds:_vbaStrCat
    mov    [ebp-11Ch], eax
    mov    eax, [edx+24h]
    push   offset aCongratulation ; "Congratulations! The password is "
    push   eax
    call   ebx ; _vbaStrCat
    mov    edx, eax
    lea    ecx, [ebp-38h]
    call   ds:_vbaStrMove
    mov    ecx, [ebp-20h]
    push   eax

```

00003255 00403255: .text:00403255 (Synchronized with Hex View-1)

Vamos a invertir el salto por un “jnz” (a estas alturas del tute ya sabemos como hacerlo). Y también guardamos un “backup” por si algo va mal.



Le damos a “OK”, salimos totalmente de IDA,

Reconocemos que estamos nerviosos pero a la vez tremadamente decididos y convencidos de lo que hemos hecho, nos secamos el sudor de la frente, nos dirigimos a la ruta donde tenemos nuestros archivos y deberíamos tener esto:

Crackme con el button activado y parcheado para que acepte cualquier password menos el correcto



Acto seguido me posiciono sobre el crackme doblemente modificado/parcheado "App2a" y lo renombro como "App2ab"



Lo ejecuto, le pongo un "Password" cualquiera (menos el correcto) , le damos a "Submit"



Y..... nos salta el mensaje de chico bueno felicitandonos, concluyendo que hemos vuelto a vencer al Crackme.

Si queremos recuperar las dos copias que hemos realizado anteriormente, (Crackme original sin ninguna modificación y crackme modificado solamente con el button activado), solo tenemos que posiconarnos sobre ellas y cambiarles el nombre simplemente quitándoles el ".bat"

Y nos quedarán así de bonitas y las tres completamente funcionales.



IDA Pro
(32-bit)



App2ab



App2



App2a

eso es todo.

Ahora si que voy a por una merecida cerveza Woll Damm..... bien, bien fresquitaaaa.

Curiosidades de este crackme con IDA

- No permite ser mostrado en modo gráfico.
- No permite guardar cambios mientras está siendo debuggeado.
- Dándole a la tecla tabulador "TAB" (⇥) y sin mover el mouse, el posicionamiento cambia entre los objetos y una vez parado en el de "Submit", le damos a <Enter> y se activa la función de comprobación.

→ Gracias a **tincopasan**, quien me hizo ver esta tercera curiosidad. ←



.....MISIÓN CUMPLIDA.....

Mis agradecimientos infinitos

Al grandísimo **RICARDO NARVAJA**, a todo el grupo **CracksLatinoS** y a todos los crackers del mundo.



Salu2

<< QwErTy >>