



REVERSE ENGINEER

by QwErTy

CracksLatinoS

Crackmes	Crackme_AbelJM_Level_0 Crackme_AbelJM_Level_1 Crackme_AbelJM_Level_2 Crackme_AbelJM_Level_3 Crackme_AbelJM_Level_4 Crackme_AbelJM_Level_5
Misión	Solucionarlos
Compilado	Los 6 con Borland Delphi 6.0 - 7.0
Protección	Ninguna
Herramientas	x32dbg - PEiD v0.95 - E2A V 2.020 - IDA 6.8 - Resource Hacker v4.2.5
Sistema Operativo	Windows Xp SP3
Reverser	QwErTy CLS
Dedicado a	AbelJM - CracksLatinoS - y a todos los crackers del mundo
Descargar Crackmes	https://mega.nz/#!SsUAlTAY!1jhq1L8b-S0VogD3JG1R_dnVuwPa7fgJFB_JpxUSZv0

ESTUDIANDO LA PRIMERA VÍCTIMA

Crackme_AbelJM_Level-0



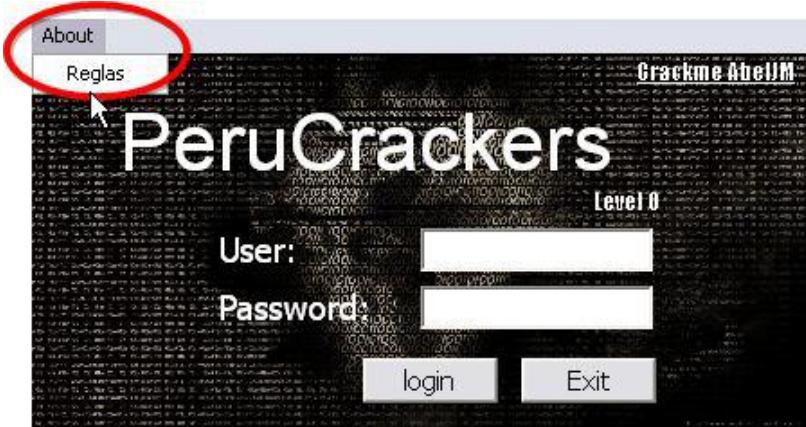
Crackme_Abel
JM_Level_0

Lo ejecutamos y nos pide un "User" y "Password", ingresamos unos datos cualesquiera para probar suerte



y como era de esperar nos salta el mensaje de "bad boy". (la verdad es que nunca, nunca, nunca... pero es que nunca.... he acertado a la primera con ningún crackme)

Miramos las Reglas



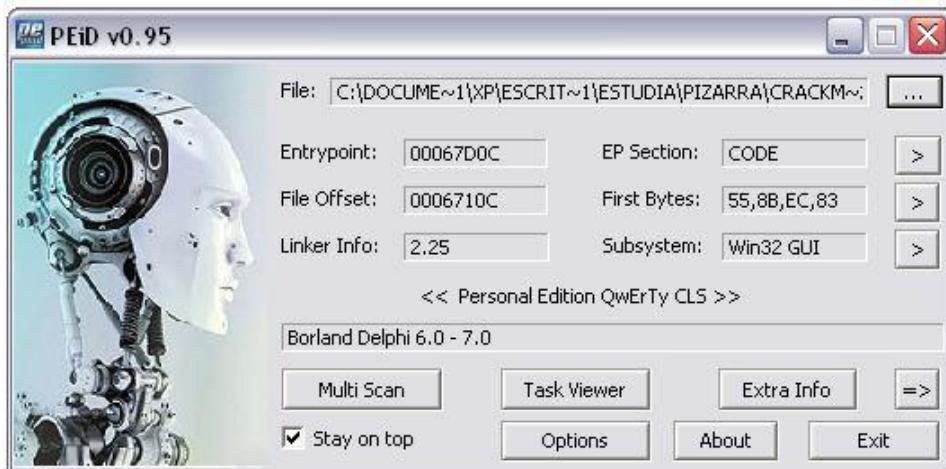
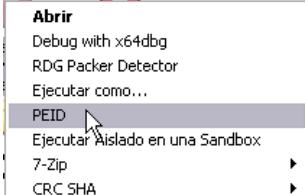
Y leemos las indicaciones del autor.



Bien....le haremos caso.

CONTINUEMOS ESTUDIANDO LA VÍCTIMA

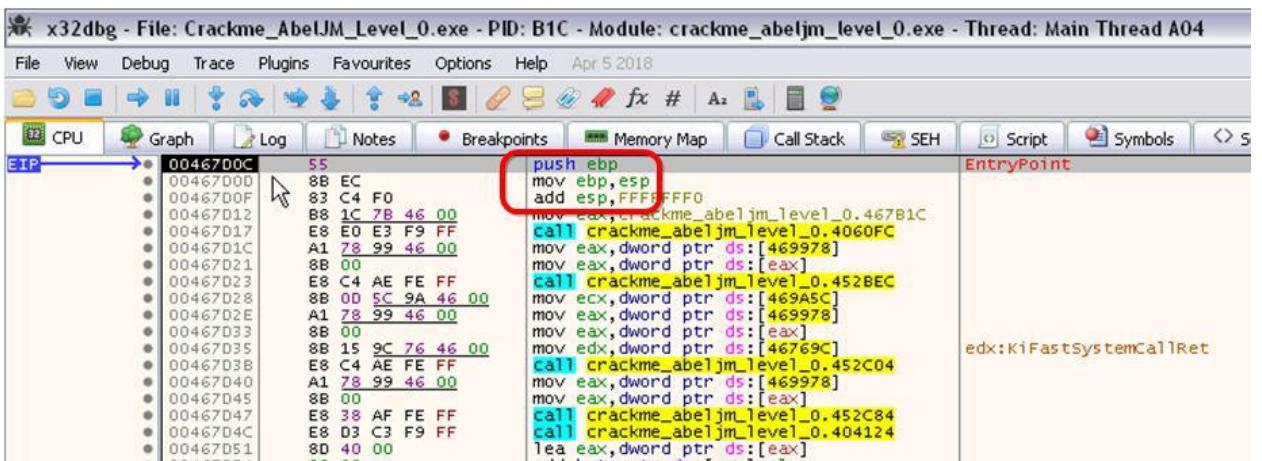
Le pasamos el detector "PEiD"



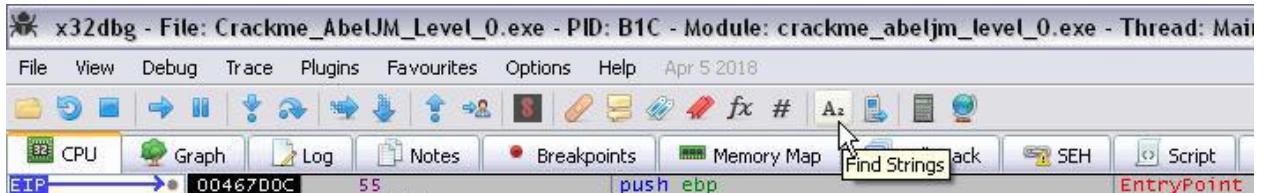
Pero bueno....., no nos quedemos embobados mirando el espectacular trabajo de maqueado que le hice a la tool "PEiD" y centrémonos en lo que estamos haciendo....., nos revela que nos enfrentamos a un compilado en "Borland Delphi"

VAMOS A POR ELLA

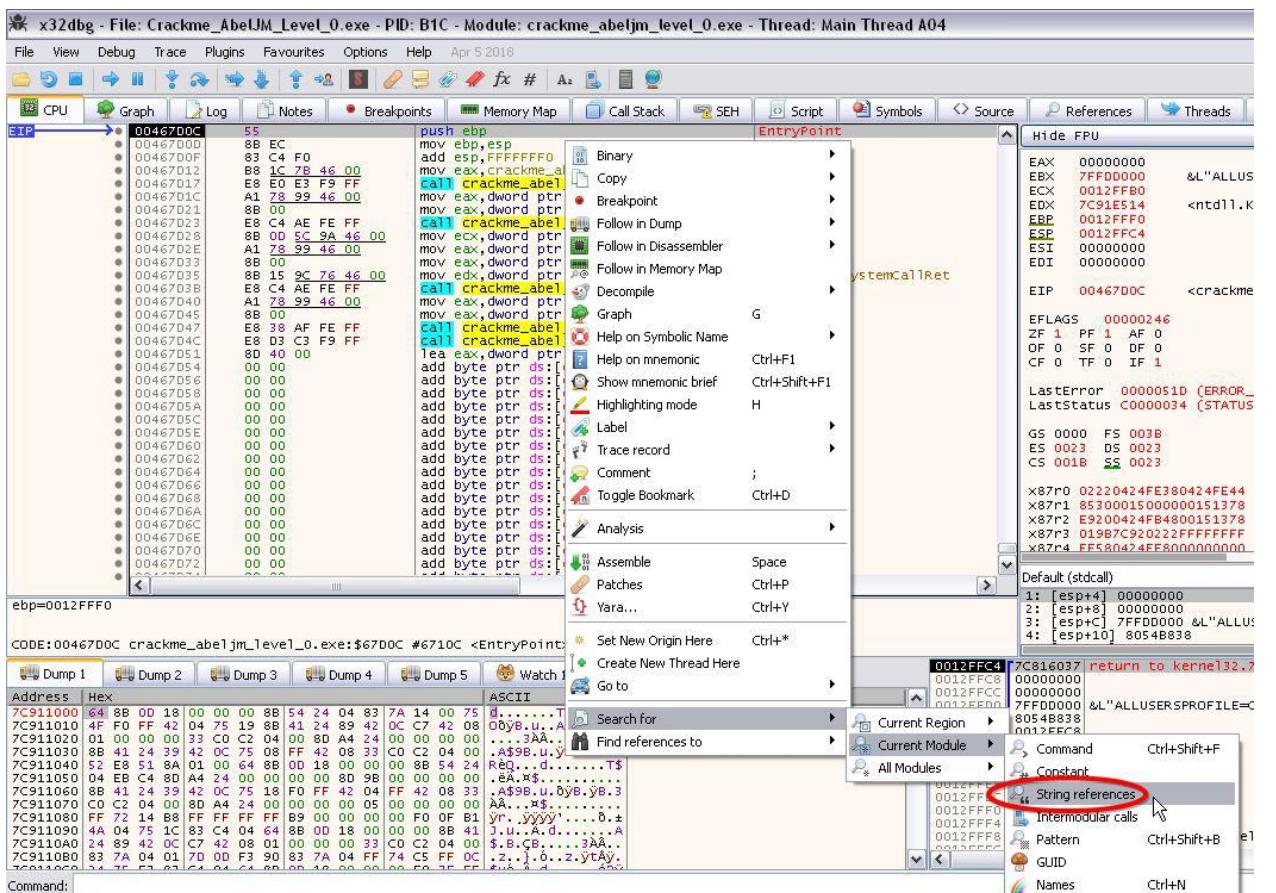
Cargamos nuestra víctima con "x32dbg", y nos encontramos parados en el "Entry Point" address "00467D0C" que al no estar empacado ya sabemos que se trata del "OEP" (Punto de Entrada Original), además también tenemos las tres instrucciones de inicio clásicas de los Borland Delphi - "push-mov-add"



Miramos las "String references" dándole directamente a "Find Strings"



O bien,



Y observamos los mensajes de "chico malo" y "chico bueno"

```

"XCE"
"pBE"
"jpg"
"\r"
"admin"
"grancracker"
"Genial lo lograste a seguir por el otro reto" ← Good boy
"Sigue intentando, ya lo lograras" ← Bad boy
"la regla es simple no usar por el momento Dede, IDR o cualquier descompilador de Delphi"
L"ALLUSERSPROFILE=C:\\Documents and Settings\\All Users"
"\r"
L"Crackme_AbelJM_Level_0.exe - PID: B1C - Module: crackme_abeljm_level_0.exe - Thread: Main"
L"Crackme_AbelJM_Level_0.exe - PID: B1C - Module: crackme_abeljm_level_0.exe - Thread: Main Tl"
L"fault IME"
"q$"
"etFilePointer"

```

Nos posicionamos sobre el "Bad boy"

```

"jpg"
"\r"
"admin"
"grancracker"
"Genial lo lograste a seguir por el otro reto"
"Sigue intentando, ya lo lograras" → I
"la regla es simple no usar por el momento Dede, IDR o cualquier descompilador de Delphi"
L"ALLUSERSPROFILE=C:\\Documents and Settings\\All Users"
"\r"

```

Dos clicks Izquierdo de ratón y apareceremos en la zona caliente, donde de entrada ya vemos la posible solución... y dos saltos condicionales "jne" decisivos.

```

x32dbg - File: Crackme_AbelJM_Level_0.exe - PID: B1C - Module: crackme_abeljm_level_0.exe - Thread: Main Thread A04
File View Debug Trace Plugins Favourites Options Help Apr 5 2018
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads
0046793B 8B 83 08 03 00 00 mov eax,dword ptr ds:[ebx+308]
00467941 E8 76 BB FC FF call crackme_abeljm_level_0_4334BC
00467946 8D 55 F8 lea edx,dword ptr ss:[ebp-8]
00467949 8B 83 04 03 00 00 mov eax,dword ptr ds:[ebx+304]
0046794F E8 68 BB FC FF call crackme_abeljm_level_0_4334BC
00467954 8B 45 FC mov eax,dword ptr ss:[ebp-4]
00467957 BA 00 79 46 00 mov edx,crackme_abeljm_level_0_467900
0046795C E8 C7 CC F9 FF call crackme_abeljm_level_0_467900
00467961 75 27 jne crackme_abeljm_level_0_46798A
00467963 8B 45 F8 mov eax,dword ptr ss:[ebp-4]
00467966 BA E0 79 46 00 mov edx,crackme_abeljm_level_0_4679E0
0046796B E8 B8 CC F9 FF call crackme_abeljm_level_0_4679E0
00467970 75 18 jne crackme_abeljm_level_0_46798A
00467972 8B F4 79 46 00 mov eax,crackme_abeljm_level_0_4679F4
00467977 E8 60 04 FC FF call crackme_abeljm_level_0_4270DC
00467979 A1 78 99 46 00 mov eax,dword ptr ds:[eax]
00467981 8B 00 call crackme_abeljm_level_0_404628
00467983 E8 D0 B3 FE FF jmp crackme_abeljm_level_0_452070
00467988 33 C0 mov eax,crackme_abeljm_level_0_4679A0
0046798A E8 2C 7A 46 00 call crackme_abeljm_level_0_4679F4
0046798F E8 48 04 FC FF call crackme_abeljm_level_0_4270DC
00467994 A1 78 99 46 00 mov eax,dword ptr ds:[eax]
00467999 8B 00 call crackme_abeljm_level_0_404628
0046799B E8 D0 B3 FE FF jmp crackme_abeljm_level_0_452070
004679A0 33 C0 xor eax,eax
004679A2 5A pop edx
004679A3 59 pop ecx
004679A4 59 pop ecx
004679A5 64 89 10 mov dword ptr [eax],edx
004679A8 68 C2 79 46 00 push crackme_abeljm_level_0_4679C2
004679AD 8D 45 F8 lea eax,dword ptr ss:[ebp-8]

```

Annotations in the screenshot:

- Two jne instructions at addresses 00467961 and 00467970 are highlighted with red arrows pointing to them.
- The text "4679F4: 'Genial lo lograste a seguir por el otro'" is circled in green.
- The text "467A2C: 'sigue intentando, ya lo lograras'" is circled in red.

Vamos a poner un "BP" en el primer salto condicional, address "00467961" a ver que pasa

```

x32dbg - File: Crackme_AbelJM_Level_0.exe - PID: C54 - Module: crackme_abeljm_level_0.exe - Thread: Main Thread F2C
File View Debug Trace Plugins Favourites Options Help Apr 5 2018
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads
0046793B 8B 83 08 03 00 00 mov eax,dword ptr ds:[ebx+308]
00467941 E8 76 BB FC FF call crackme_abeljm_level_0_4334BC
00467946 8D 55 F8 lea edx,dword ptr ss:[ebp-8]
00467949 8B 83 04 03 00 00 mov eax,dword ptr ds:[ebx+304]
0046794F E8 68 BB FC FF call crackme_abeljm_level_0_4334BC
00467954 8B 45 FC mov eax,dword ptr ss:[ebp-4]
00467957 BA 00 79 46 00 mov edx,crackme_abeljm_level_0_467900
0046795C E8 C7 CC F9 FF call crackme_abeljm_level_0_467900
00467961 75 27 jne crackme_abeljm_level_0_46798A
00467963 8B 45 F8 mov eax,dword ptr ss:[ebp-4]
00467966 BA E0 79 46 00 mov edx,crackme_abeljm_level_0_4679E0
0046796B E8 B8 CC F9 FF call crackme_abeljm_level_0_4679E0
00467970 75 18 jne crackme_abeljm_level_0_46798A
00467972 8B F4 79 46 00 mov eax,crackme_abeljm_level_0_4679F4
00467977 E8 60 04 FC FF call crackme_abeljm_level_0_4270DC
00467979 A1 78 99 46 00 mov eax,dword ptr ds:[eax]
00467981 8B 00 call crackme_abeljm_level_0_404628
00467983 E8 D0 B3 FE FF jmp crackme_abeljm_level_0_452070
00467988 33 C0 mov eax,crackme_abeljm_level_0_4679A0
0046798A E8 2C 7A 46 00 call crackme_abeljm_level_0_4679F4
0046798F E8 48 04 FC FF call crackme_abeljm_level_0_4270DC
00467994 A1 78 99 46 00 mov eax,dword ptr ds:[eax]
00467999 8B 00 call crackme_abeljm_level_0_404628
0046799B E8 D0 B3 FE FF jmp crackme_abeljm_level_0_452070
004679A0 33 C0 xor eax,eax
004679A2 5A pop edx
004679A3 59 pop ecx
004679A4 59 pop ecx
004679A5 64 89 10 mov dword ptr [eax],edx
004679A8 68 C2 79 46 00 push crackme_abeljm_level_0_4679C2
004679AD 8D 45 F8 lea eax,dword ptr ss:[ebp-8]

```

Damos "run" para que corra el crackme, rellenamos las cajas de texto "User" y "Password" con nuestros datos truchos

The screenshot shows the assembly code with a red box highlighting the first conditional jump at address 00467961. The crackme application window is overlaid on the debugger, displaying the text "PeruCrackers" and "Level 0". The "User:" field contains "QwErTy CLS" and the "Password:" field contains "0123456789".

Le damos a "Login" y

vemos con sumo agrado lo que ya intuimos.... seguimos traceando y nos vamos directos a "Bad boy" ya que las comparaciones de los "User" y "Password" que hemos tipeado difieren de los mostrados.

Damos "run".

Probamos ahora con los datos mostrados y que creemos correctos

Traceamos y ahora al ser iguales los valores, los saltos condicionales nos llevan directos a la zona correcta "Good boy"



ESTUDIANDO LA SEGUNDA VÍCTIMA

Crackme_AbelJM_Level-1



Lo ejecutamos y al igual que el anterior, nos pide un "User" y "Password", ingresamos unos datos cualesquiera para probar suerte, y como de costumbre nos salta el mensaje de "bad boy", (gggggrrrrrrrrrr)



Miramos las Reglas indicadas por el autor

Archivo

Reglas

PeruCrackers

Level 1

User:

Password:

Crackme AbelJM - Level 01

Login

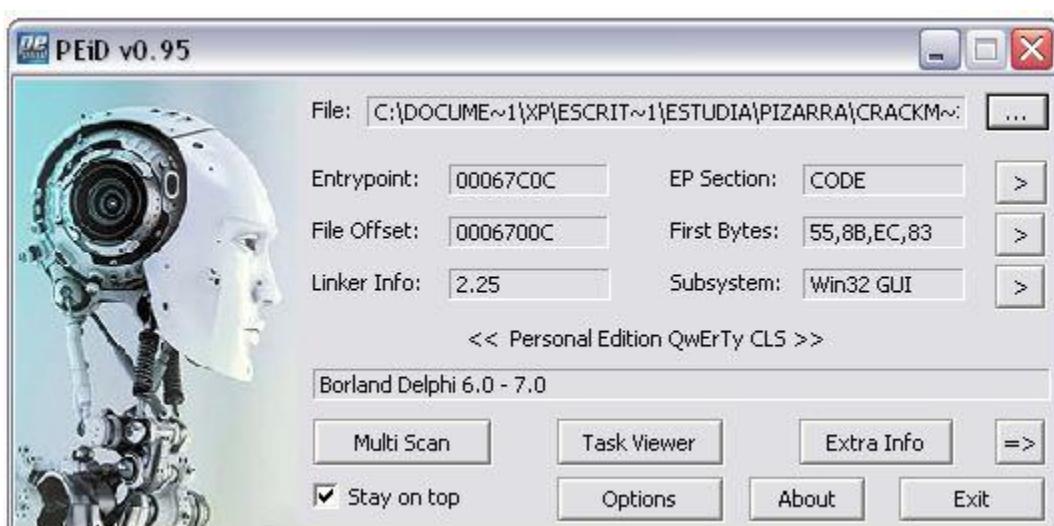
Y la leemos.



Bien....pues también le haremos caso.

CONTINUEMOS ESTUDIANDO LA VÍCTIMA

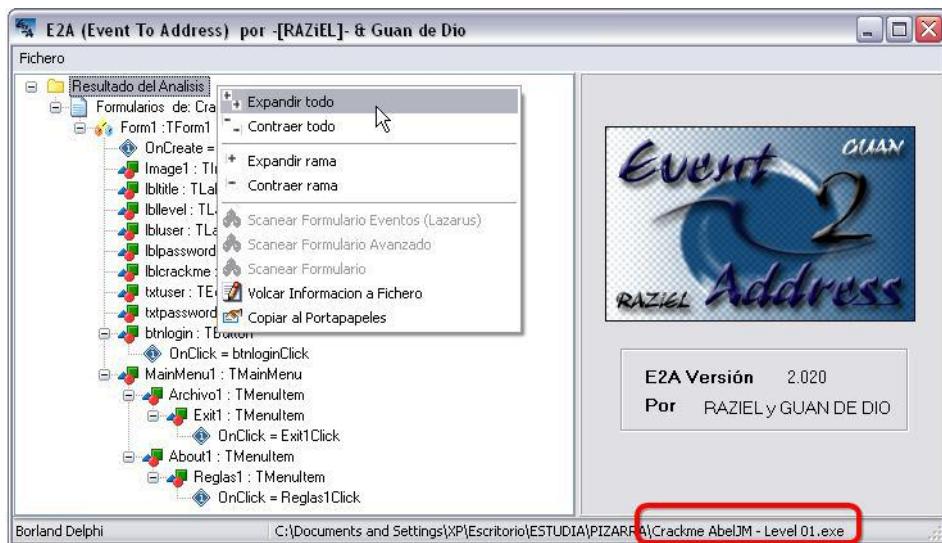
Le pasamos el detector "PEiD"



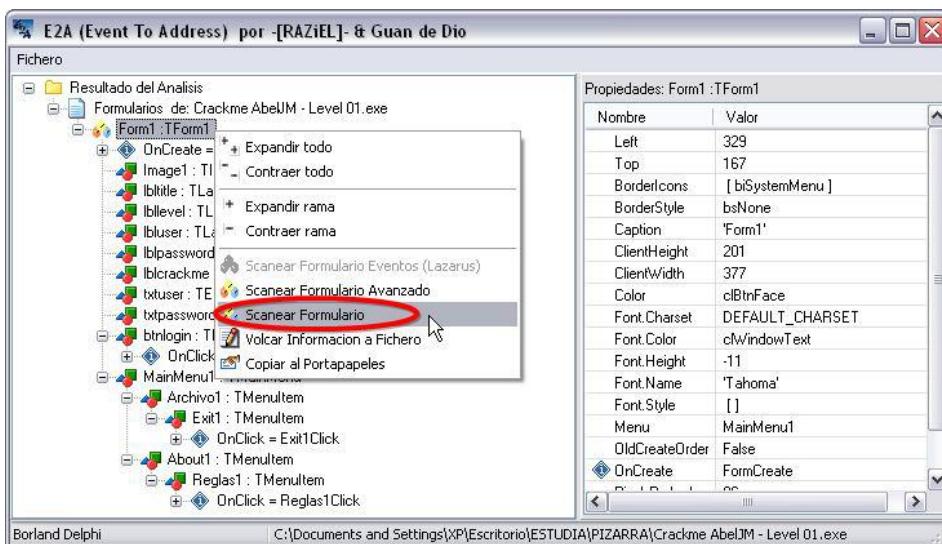
Nuevamente nos enfrentamos a otro "Borland Delphi" .

VAMOS A POR ELLA

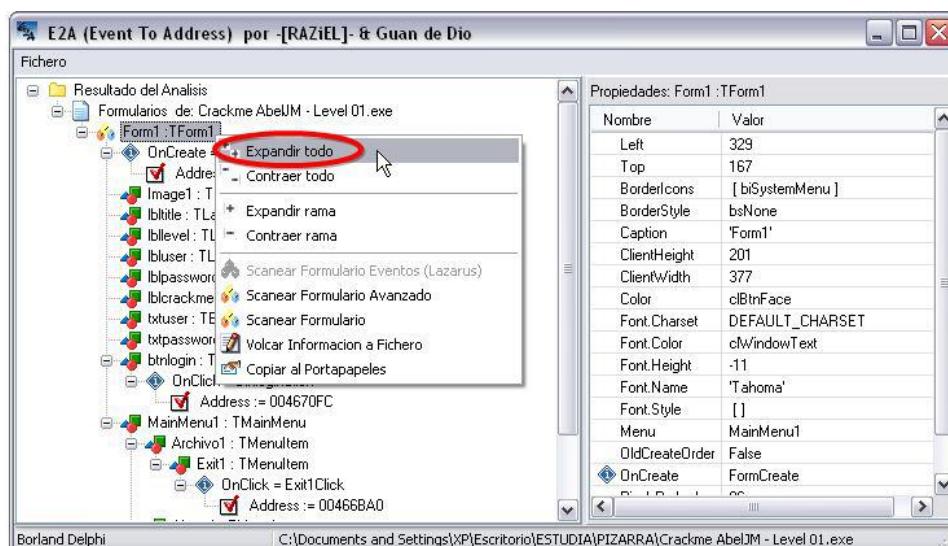
Cargamos el crackme con la fantástica tool "E2A", le damos a "Expandir todo"



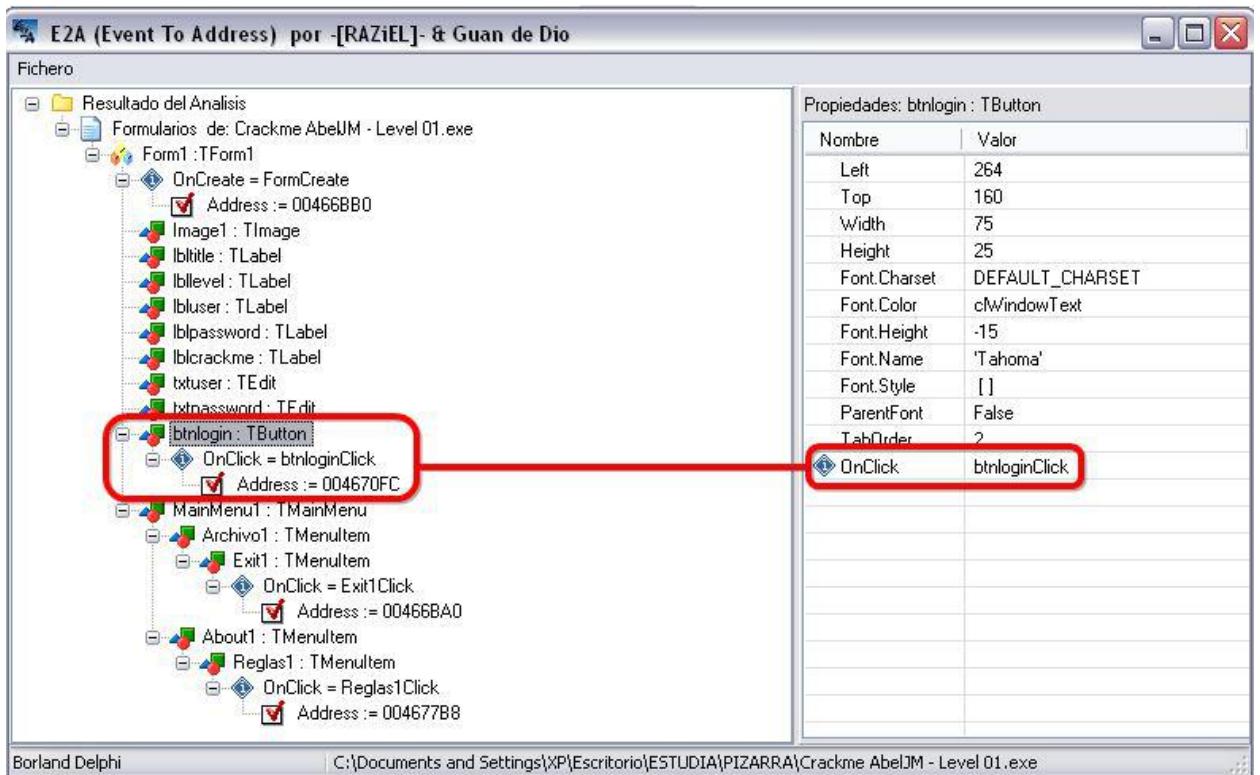
Nos posicionamos sobre "Form1 :Form1" y le damos a "Scanear Formulario"



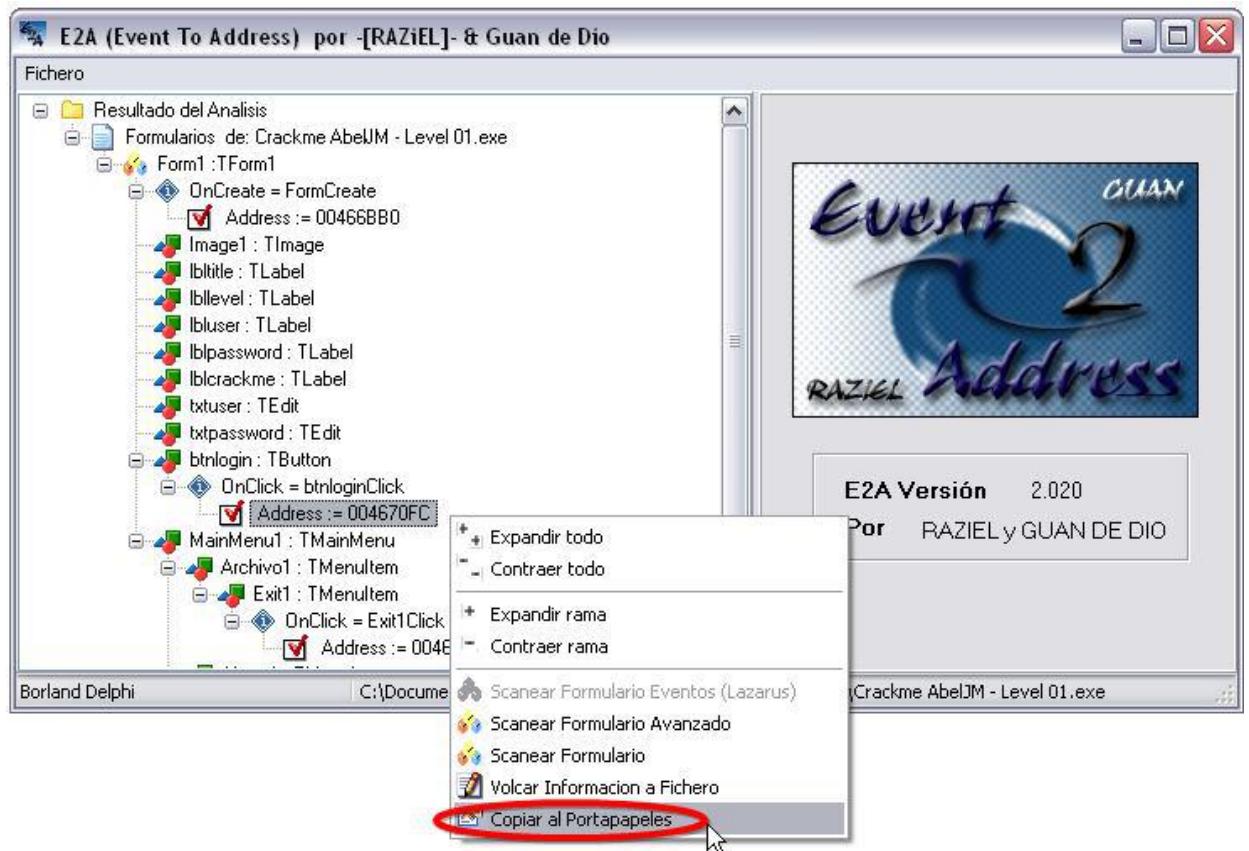
Una vez escaneado, le volvemos a dar a "Expandir todo"



y buscamos la address donde se ejecutará la primera instrucción de inicio del button "Login" de nuestra víctima (cuando lo clickeemos en el momento de validar, claro).

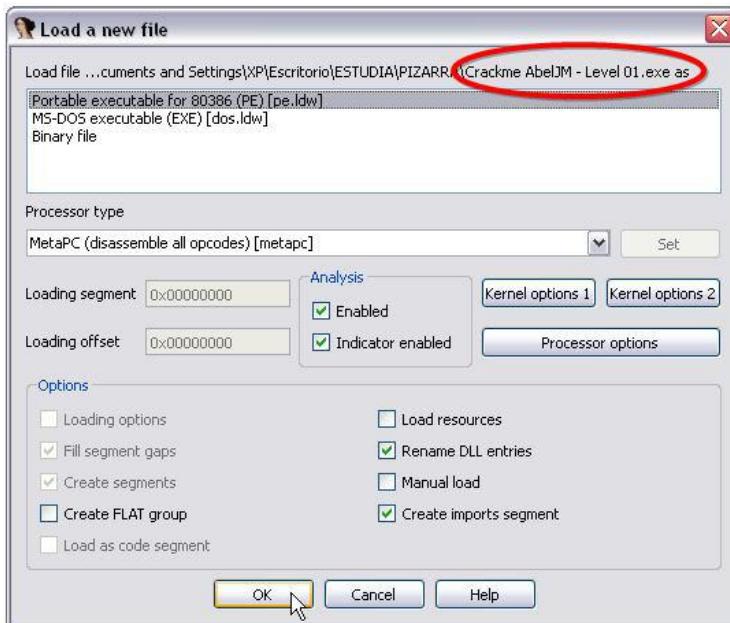


Y obtenemos que lo hará justamente en la address "004670FC", copiamos esa valiosísima información directamente al portapapeles.

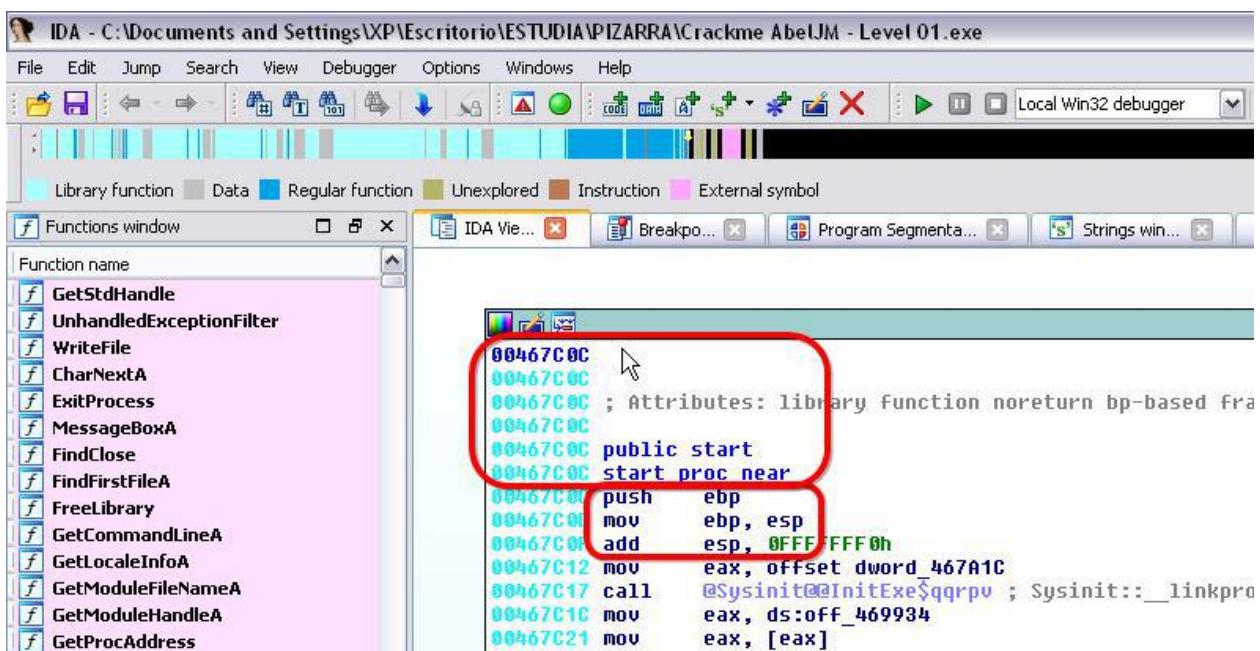


Y ya podemos salir de esta tool.

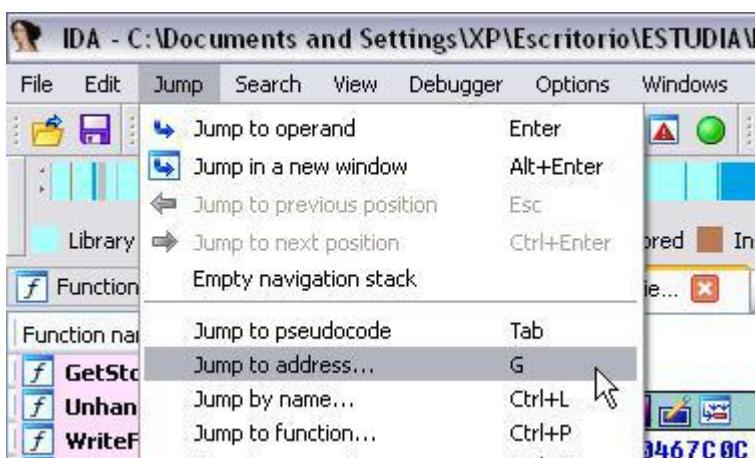
Cargamos nuestro "Crackme_AbelJM_Level-1" con "IDA",



y nos encontramos parados en el Entry Point address "00467C0C", (OEP)



Ahora nos vamos directos a la address que obtuvimos anteriormente



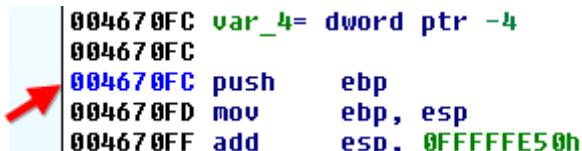
La pegamos

```
; Attributes: library function noreturn bp-based frame

public start
start proc near
push    ebp
mov     ebp, esp
add    esp, 0FFFFE50h
mov     eax, offset @Sysinit@_initExe$qqrpv ; sysinit::__linkproc__ _ini
call    eax. ds:off 469934
```



Le damos a "OK", y aparecemos aquí:



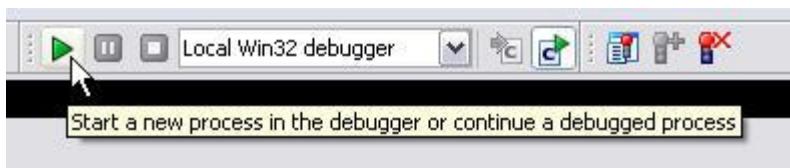
```
004670FC var_4= dword ptr -4
004670FC
004670FC push    ebp
004670FD mov     ebp, esp
004670FF add    esp, 0xFFFFFE50h
```

Le ponemos un "BP"

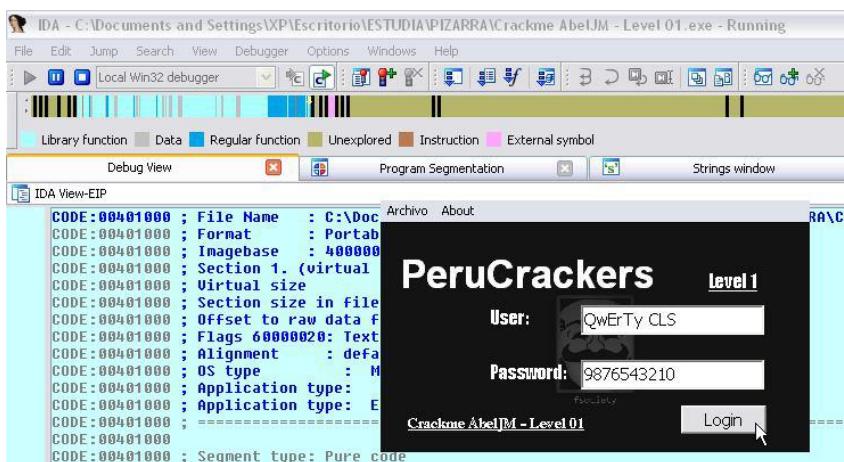


```
004670FC var_4= dword ptr -4
004670FC
004670FC push    ebp
004670FD mov     ebp, esp
004670FF add    esp, 0xFFFFFE50h
00467105 push    ebx
```

Con nuestro punto de parada colocado, le damos a "Start" para que corra el debugger



Rellenamos datos:



Le damos al button "Login" y nos encontramos parados en nuestro Breakpoint, justo donde le indicamos a IDA.

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 01.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library function Data Regular Function Unexplored Instruction External symbol

Debug View Program Segmentation Strings

IDA View-EIP

```
CODE:004670FC var_C= dword ptr -0Ch
CODE:004670FC var_8= dword ptr -8
CODE:004670FC var_4= dword ptr -4
CODE:004670FC
EIP CODE:004670FC push ebp
CODE:004670FD mov esp, ebp
CODE:004670FF add esp, 0FFFFFFE5h
CODE:00467105 push ebx
CODE:0046710A xor ecx, ecx
```

Ahora vamos traceando y observando como curiosamente va leyendo y guardando texto:

Debug View Program Segmentation

IDA View-EIP

```
CODE:00467133 mov fs:[eax], esp
CODE:00467136 lea eax, [ebp+var_8]
CODE:00467139 push eax
CODE:0046713A mov eax, offset _str_r_0.Text
CODE:0046713F mov [ebp+var_30], eax
CODE:00467142 mov eax, offset _str_o_0.Text
CODE:00467147 mov [ebp+var_2C], eax
CODE:0046714A mov eax, offset _str_m_0.Text
CODE:0046714F mov [ebp+var_28], eax
CODE:00467152 mov eax, offset _str_p.Text
CODE:00467157 mov [ebp+var_24], eax
CODE:0046715A mov eax, offset _str_e_2.Text
CODE:0046715F mov [ebp+var_20], eax
CODE:00467162 mov eax, offset _str_o_0.Text
CODE:00467167 mov [ebp+var_1C], eax
CODE:0046716A lea eax, [ebp+var_30]
CODE:0046716D xor ecx, ecx
CODE:0046716F mov edx, 5
0006653F 0046713F: sub_4670FC+43 (Synchronized with EIP)
```

Seguimos traceando

Debug View Program Segmentation

IDA View-EIP

```
CODE:00467179 lea eax, [ebp+var_4]
CODE:0046717C push eax
CODE:0046717D mov eax, offset _str_c_0.Text
CODE:00467182 mov [ebp+var_54], eax
CODE:00467185 mov eax, offset _str_r_0.Text
CODE:0046718A mov [ebp+var_50], eax
CODE:0046718D mov eax, offset _str_a_0.Text
CODE:00467192 mov [ebp+var_4C], eax
CODE:00467195 mov eax, offset _str_c_0.Text
CODE:0046719A mov [ebp+var_48], eax
CODE:0046719D mov eax, offset _str_k_0.Text
CODE:004671A2 mov [ebp+var_44], eax
CODE:004671A5 mov eax, offset _str_m_0.Text
CODE:004671AA mov [ebp+var_40], eax
CODE:004671AD mov eax, offset _str_e_2.Text
CODE:004671B2 mov [ebp+var_3C], eax
CODE:004671B5 mov eax, offset _str_s_0.Text
CODE:004671BA mov [ebp+var_38], eax
CODE:004671BD mov eax, offset _str_o_0.Text
EIP CODE:004671C2 mov [ebp+var_34], eax
CODE:004671C5 lea eax, [ebp+var_54]
```

Y traceando....y observando....

```
CODE:004671D7 push    eax
CODE:004671D8 mov     eax, offset _str_F_0.Text
CODE:004671DD mov     [ebp+var_E4], eax
CODE:004671E3 mov     eax, offset _str_e_2.Text
CODE:004671E8 mov     [ebp+var_E8], eax
CODE:004671EE mov     eax, offset _str_l_0.Text
CODE:004671F3 mov     [ebp+var_DC], eax
CODE:004671F9 mov     eax, offset _str_i_0.Text
CODE:004671FE mov     [ebp+var_D8], eax
CODE:00467204 mov     eax, offset _str_c_0.Text
CODE:00467209 mov     [ebp+var_D4], eax
CODE:0046720F mov     eax, offset _str_i_0.Text
CODE:00467214 mov     [ebp+var_D0], eax
CODE:0046721A mov     eax, offset _str_d_0.Text
CODE:0046721F mov     [ebp+var_CC], eax
CODE:00467225 mov     eax, offset _str_a_0.Text
CODE:0046722A mov     [ebp+var_C8], eax
CODE:00467230 mov     eax, offset _str_d_0.Text
CODE:00467235 mov     [ebp+var_C4], eax
CODE:0046723B mov     eax, offset _str_e_2.Text
CODE:00467240 mov     [ebp+var_C0], eax
CODE:00467246 mov     eax, offset _str_s_0.Text
EIP CODE:0046724B mov     [ebp+var_B0], eax
CODE:00467251 mov     eax, offset _str__15.Text
```

0006661A 0046721A: sub_4670FC+11E (Synchronized with EIP)

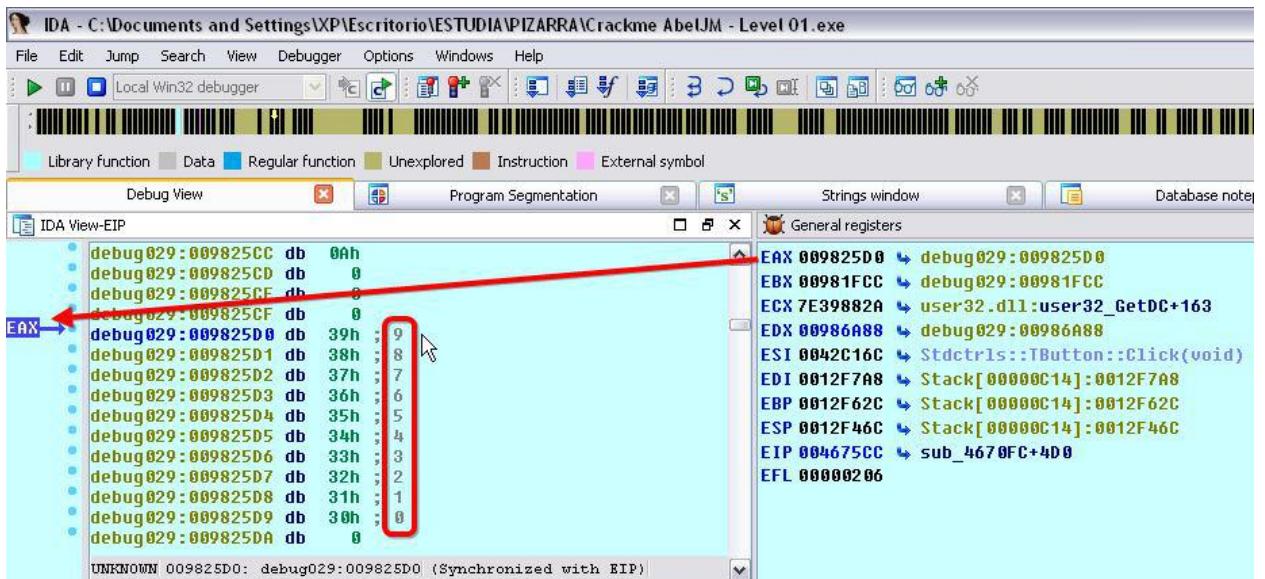
hasta llegar a la address "004675CC" donde vemos dos llamadas "CALL" precedidas de dos instrucciones condicionales "jnz" que tienen toda la pinta de que estamos en la zona caliente

```
File Edit Jump Search View Debugger Options Windows Help
Local Win32 debugger
Library Function Data Regular function Unexplored Instruction External symbol
Debug View Program Segmentation Strings window
IDA View-EIP
EIP → CODE:004675B8 mov     eax, [ebx+318h] ; this
CODE:004675BE call    @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675C3 mov     eax, [ebp+var_1AC]
CODE:004675C9 mov     edx, [ebp+var_4]
CODE:004675CC call    @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void) ←
CODE:004675D1 jnz    short loc_467618
CODE:004675D3 lea     edx, [ebp+var_1B0]
CODE:004675D9 mov     eax, [ebx+32Ch] ; this
CODE:004675DF call    @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675E4 mov     eax, [ebp+var_1B0]
CODE:004675EA mov     edx, [ebp+var_8]
CODE:004675ED call    @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void) ←
CODE:004675F2 jnz    short loc_467618
CODE:004675F4 push    0 ; uType
CODE:004675F6 mov     eax, [ebp+var_14]
```

Aquí parados, y posicionados sobre la address "004675CC" nos vamos a la ventana "General registers" colocamos el cursor sobre la flechita azul del Registro "EAX 009825D0"

EAX	009825D0	↳ debug029:009825D0	OF 0
EBX	00981FCC	↳ debug029:00981FCC	DF 0
ECX	7E39882A	↳ user32.dll:user32_GetDC+163	IF 1
EDX	00986A88	↳ debug029:00986A88	TF 0
ESI	0042C16C	↳ Stdctrls::TButton::Click(void)	SF 0
EDI	0012F7A8	↳ Stack[00000C14]:0012F7A8	ZF 0
EBP	0012F62C	↳ Stack[00000C14]:0012F62C	AF 0
ESP	0012F46C	↳ Stack[00000C14]:0012F46C	PF 1
EIP	004675CC	↳ sub_4670FC+4D0	CF 0
EFL	00000206		

Y sobre ella damos dos clicks Derecho de ratón, y en la ventana del desensamblado nos desvela que ahí guarda nuestro "Password" tipeado.



Si hacemos scroll hacia arriba o hacia abajo no vemos ni rastro de nuestro "User" "QwErTy CLS".

Seguimos.....ahora hacemos la misma operación, pero sobre el registro "EDX"

General registers
EAX 009825D0 ↳ debug029:009825D0
EBX 00981FCC ↳ debug029:00981FCC
ECX 7E39882A ↳ user32.dll:user32_GetDC+163
EDX 00986A88 ↳ debug029:00986A88
ESI 0042C16C ↳ Stdctrls::TButton::Click(void)
EDI 0012F7A8 ↳ Stack[00000C14]:0012F7A8
EBP 0012F62C ↳ Stack[00000C14]:0012F62C
ESP 0012F46C ↳ Stack[00000C14]:0012F46C
EIP 004675CC ↳ sub_4670FC+4D0
EFL 00000206

y en la ventana del desensamblado vemos que ahí guarda la string "crackmes"

```

IDA View-EIP
[...]
EDX → debug029:00986A88 db 63h ; c
debug029:00986A89 db 72h ; r
debug029:00986A8A db 61h ; a
debug029:00986A8B db 63h ; c
debug029:00986A8C db 66h ; k
debug029:00986A8D db 66h ; m
debug029:00986A8E db 65h ; e
debug029:00986A8F db 73h ; s
debug029:00986A90 db 0
debug029:00986A91 db 0
debug029:00986A92 db 46h ; F

```

General registers

```

EAX 009825D0 ↳ debug029:009825D0
EBX 00981FCC ↳ debug029:00981FCC
ECX 7E39882A ↳ user32.dll:user32_GetDC+163
EDX 00986A88 ↳ debug029:00986A88
ESI 0042C16C ↳ StdCtrls:TButton::Click(void)
EDI 0012F7A8 ↳ Stack[00000C14]:0012F7A8
EBP 0012F62C ↳ Stack[00000C14]:0012F62C
ESP 0012F46C ↳ Stack[00000C14]:0012F46C
EIP 004675CC ↳ sub_4670FC+4D0
EFL 00000206

```

Y si hacemos un poco de scroll hacia arriba vemos la string "rompe"

```

IDA View-EIP
[...]
debug029:00986A6F db 0
debug029:00986A70 db 5
debug029:00986A71 db 0
debug029:00986A72 db 0
debug029:00986A73 db 0
debug029:00986A74 db 72h ; r
debug029:00986A75 db 6Fh ; o
debug029:00986A76 db 6Dh ; m
debug029:00986A77 db 70h ; p
debug029:00986A78 db 65h ; e
debug029:00986A79 db 0
debug029:00986A7A db 0
debug029:00986A7B db 0
debug029:00986A7C db 1Ah
debug029:00986A7D db 0

```

UNKNOWN 00986A73: debug029:00986A73 (Synchronized with EIP)

General registers

```

EAX 009825D0 ↳ debug029:009825D0
EBX 00981FCC ↳ debug029:00981FCC
ECX 7E39882A ↳ user32.dll:user32_GetDC+163
EDX 00986A88 ↳ debug029:00986A88
ESI 0042C16C ↳ StdCtrls:TButton::Click(void)
EDI 0012F7A8 ↳ Stack[00000C14]:0012F7A8
EBP 0012F62C ↳ Stack[00000C14]:0012F62C
ESP 0012F46C ↳ Stack[00000C14]:0012F46C
EIP 004675CC ↳ sub_4670FC+4D0
EFL 00000206

```

Bien....., ya hemos visto lo que guardan los registros "EAX" y "EDX" en memoria, ahora vamos a volver a la vista normal la ventana principal, para ello colocamos el cursor del ratón sobre la flechita azul del registro "EIP 004675CC", le damos un click izquierdo y aquí estamos:

```

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 01.exe
File Edit Jump Search View Debugger Options Windows Help
[...]
Library Function Data Regular function Unexplored Instruction External symbol
Debug View Program Segmentation Strings window Database notepad
IDA View-EIP
[...]
EIP → CODE:004675B8 mov eax, [ebx+318h] ; this
CODE:004675B8 call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675C3 mov eax, [ebp+var_1AC]
CODE:004675C9 mov edx, [ebp+var_4]
CODE:004675CC call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void)
EIP → CODE:004675D1 jnz short loc_467618
CODE:004675D3 lea edx, [ebp+var_180]
CODE:004675D9 mov eax, [ebx+32Ch] ; this
CODE:004675DF call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675E4 mov eax, [ebp+var_180]
CODE:004675EA mov edx, [ebp+var_8]
CODE:004675ED call @System@LStrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004675F2 jnz short loc_467618
CODE:004675F4 push 0 ; uType
CODE:004675F6 mov eax, [ebp+var_14]

```

000669CC 004675CC: sub_4670FC+4D0 (Synchronized with EIP)

General registers

```

EAX 009825D0 ↳ debug029:009825D0
EBX 00981FCC ↳ debug029:00981FCC
ECX 7E39882A ↳ user32.dll:user32_GetDC+163
EDX 00986A88 ↳ debug029:00986A88
ESI 0042C16C ↳ StdCtrls:TButton::Click(void)
EDI 0012F7A8 ↳ Stack[00000C14]:0012F7A8
EBP 0012F62C ↳ Stack[00000C14]:0012F62C
ESP 0012F46C ↳ Stack[00000C14]:0012F46C
EIP 004675CC ↳ sub_4670FC+4D0
EFL 00000206

```

Ahora nos posicionamos sobre la ventana "Hex View" damos un click izquierdo de ratón para que IDA sepa que estamos en ese sitio,

```

Hex View-1
[...]
004670BC 6D 00 00 00 FF FF FF 01 00 00 00 41 00 00 00 m....-....A...
004670C0 FF FF FF FF 01 00 00 00 62 00 00 00 FF FF FF .....b.....
004670DC 01 00 00 00 4A 00 00 00 FF FF FF 01 00 00 00 ....J.....
004670E0 4D 00 00 00 FF FF FF 01 00 00 00 2D 00 00 00 M....-....-
004670F0 55 88 EC 81 C4 50 FE FF F5 33 C9 89 8D 50 FE UÍÜ.-P!-S3+E.P!
FF FF 89 4D FF FF FF 89 4D F8 89 4D .....-.....
0046710C FF FF 89 4D FF FF FF 89 4D F8 89 4D .....-.....
0046711C F8 89 4D EC 89 4D E8 88 D8 33 C0 55 68 qménhjémp1f3-uh
0046712C 71 76 46 09 64 FF 30 64 89 28 8D 45 F8 50 88 88 qVF.d-0dè-E.P'øé
0046713C 76 46 00 89 45 D8 B8 94 76 46 00 89 45 D4 B8 A8 VF.ÆØØØVF.ÆØØØ
0046714C 76 46 00 89 45 D8 B8 AC 76 46 00 89 45 DC B8 B8 VF.ÆØØØVF.ÆØØØ
0046715C 76 46 00 89 45 E0 B8 94 76 46 00 89 45 E4 8D 45 VF.ÆØØØVF.ÆØØØ.E
0046716C D0 33 C9 FF BA 05 00 00 00 E8 D3 F9 FF FF 8D 45 FC Ø3+!...þE'...E³

```

000664FC 004670FC: sub_4670FC |

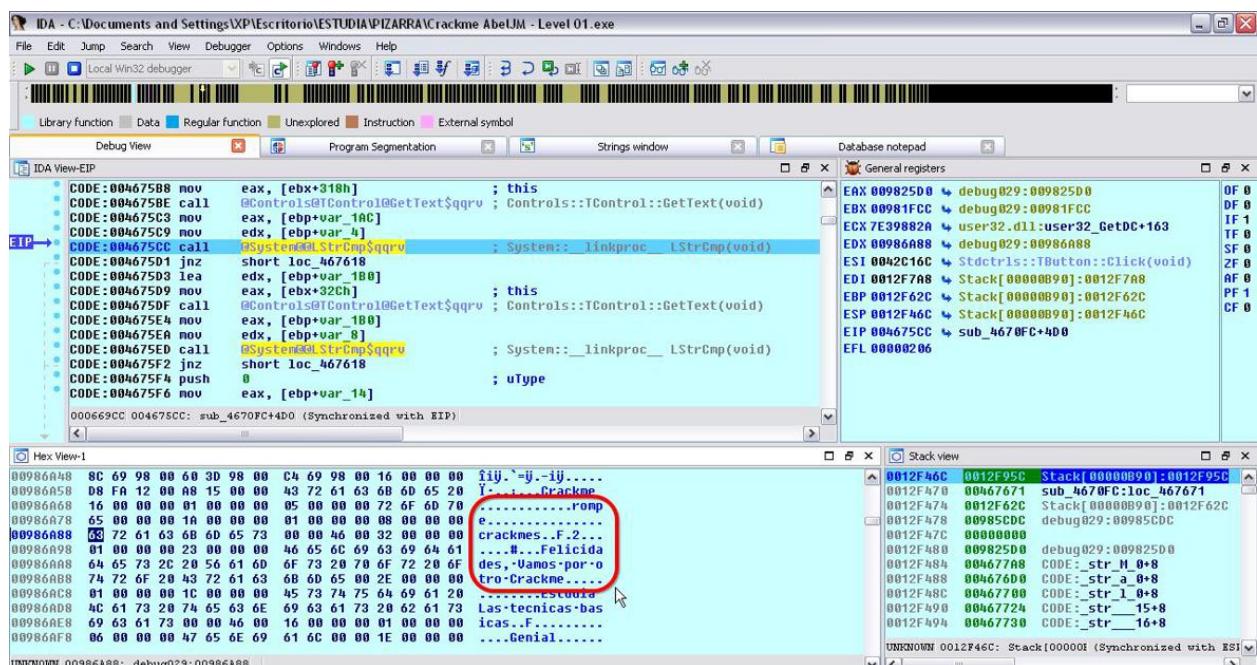
Vuelvo a la ventana "Registers" , coloco el cursor sobre la flechita azul del registro "EDX"

```

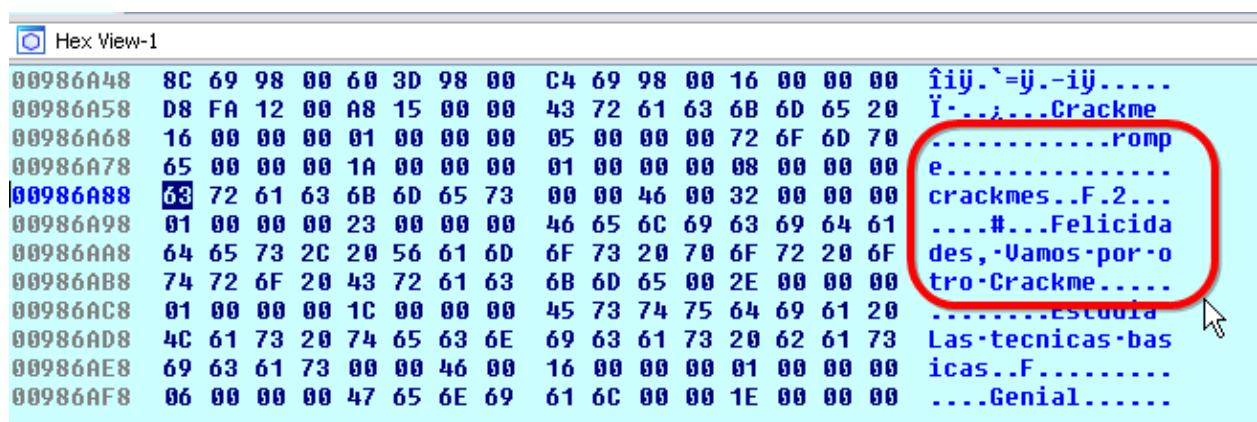
EAX 009825D0 ↳ debug029:009825D0
EBX 00981FCC ↳ debug029:00981FCC
ECX 7E39882A ↳ user32.dll:user32_GetDC+163
EDX 00986A88 ↳ debug029:00986A88
ESI 0042C16C ↳ Stdctrls::TButton::Click(void)
EDI 0012F7A8 [Jump in disassembly] :0012F7A8
EBP 0012F62C ↳ Stack[00000B90]:0012F62C
ESP 0012F46C ↳ Stack[00000B90]:0012F46C
EIP 004675CC ↳ sub_4670FC+4D0
EFL 00000206

```

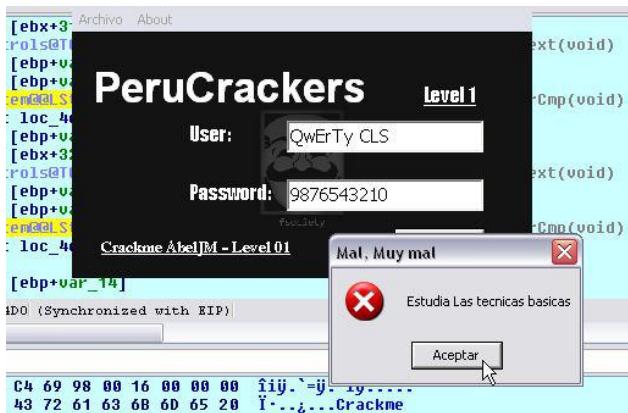
Y en la ventana "Hex View" veo esto:



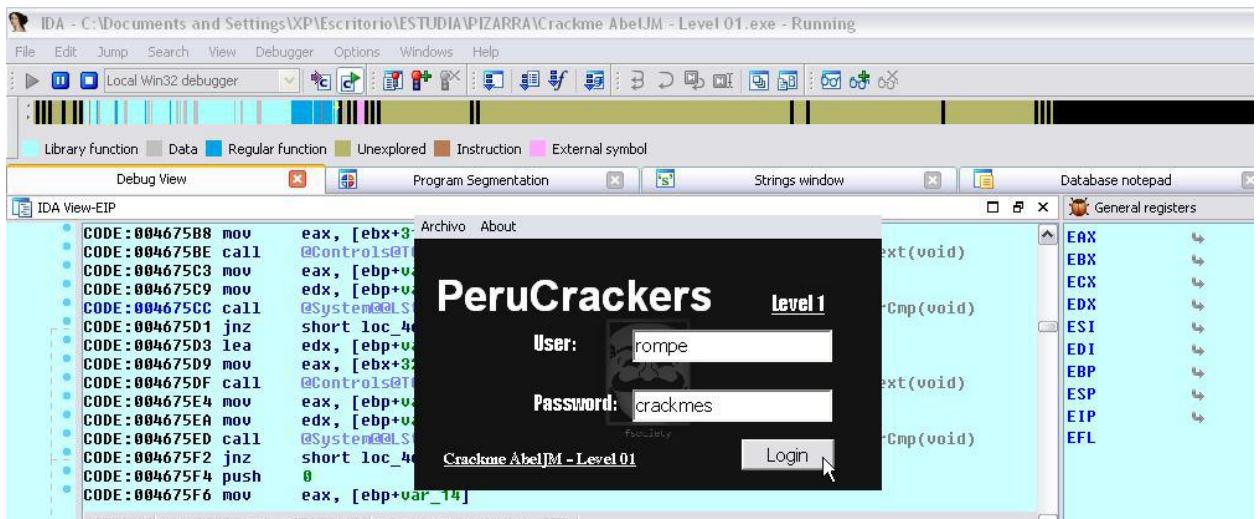
Voy a aumentarlo para que se vea más claro:



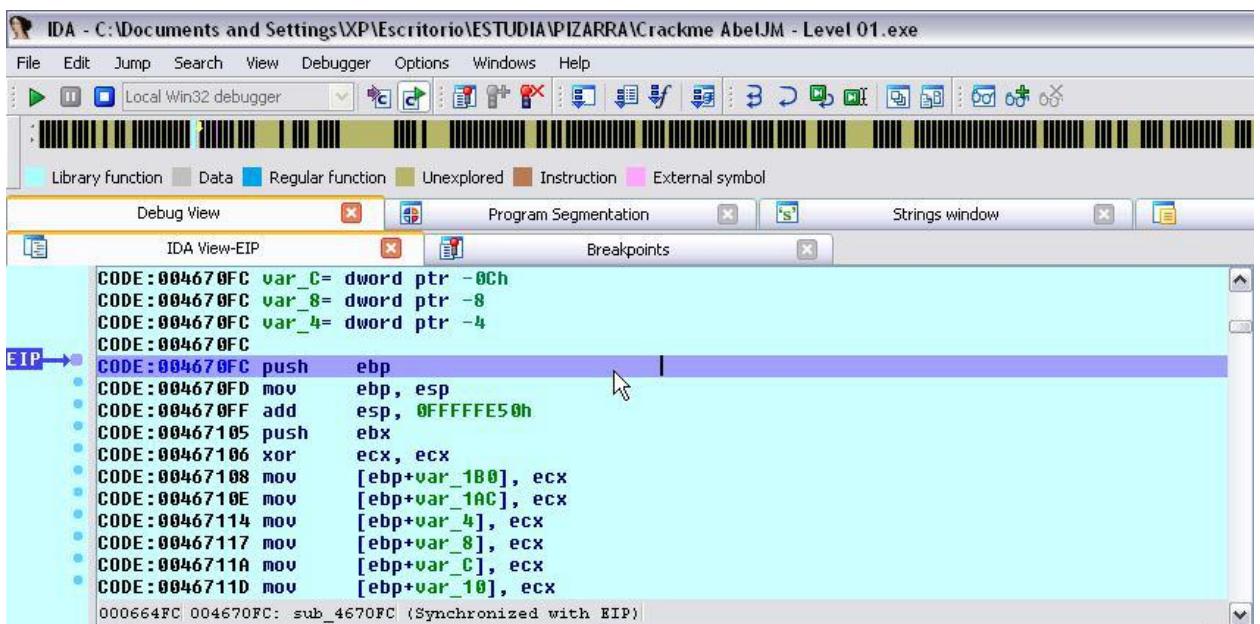
De lo que deduzco que el "User" que estamos buscando podría ser la String "rompe" y el "Password" la String "crackmes" . Acto seguido le pongo un "BP" en la address en la que todavía estamos parados "004675CC" , le damos a "Start" para que salga el dichoso mensaje del "bad boy"



Reiniciamos, volvemos al debugger y ahora tipeamos los datos que hemos obtenido:



Le damos otra vez a "Start", y IDA para en nuestro primer "BP".



otro "Start" más y nos encontramos parados en nuestro segundo "BP"

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 01.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library Function Data Regular function Unexplored Instruction External symbol

Debug View Program Segmentation Strings window

IDA View-EIP Breakpoints

EIP →

```
CODE:004675B8 mov eax, [ebx+318h] ; this
CODE:004675BE call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675C3 mov eax, [ebp+var_1AC]
CODE:004675C9 mov edx, [ebp+var_4]
CODE:004675CC call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void)
CODE:004675D1 jnz short loc_467618
CODE:004675D3 lea edx, [ebp+var_1B0]
CODE:004675D9 mov eax, [ebx+32Ch] ; this
CODE:004675DF call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675E4 mov eax, [ebp+var_1B0]
CODE:004675EA mov edx, [ebp+var_8]
CODE:004675ED call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void)
CODE:004675F2 jnz short loc_467618
CODE:004675F4 push 0 ; uType
CODE:004675F6 mov eax, [ebp+var_14]

000669CC 004675CC: sub_4670FC+4D0 (Synchronized with EIP)
```

Ahora vamos a tracear con mucho cuidado con "F8", y vemos como el salto condicional "jnz" nos deja pasar...je,je,je... vamos bien...

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 01.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library Function Data Regular function Unexplored Instruction External symbol

Debug View Program Segmentation Strings window

IDA View-EIP Breakpoints

EIP →

```
CODE:004675B8 mov eax, [ebx+318h] ; this
CODE:004675BE call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675C3 mov eax, [ebp+var_1AC]
CODE:004675C9 mov edx, [ebp+var_4]
CODE:004675CC call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void) ← Red box
CODE:004675D1 jnz short loc_467618
CODE:004675D3 lea edx, [ebp+var_1B0] ← Red box
CODE:004675D9 mov eax, [ebx+32Ch] ; this
CODE:004675DF call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675E4 mov eax, [ebp+var_1B0]
CODE:004675EA mov edx, [ebp+var_8]
CODE:004675ED call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void)
CODE:004675F2 jnz short loc_467618
CODE:004675F4 push 0 ; uType
CODE:004675F6 mov eax, [ebp+var_14]

000669D3 004675D3: sub_4670FC+4D7 (Synchronized with EIP)
```

Continuamos traceando con "F8" y....

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 01.exe

File Edit Jump Search View Debugger Options Windows Help

Local Win32 debugger

Library Function Data Regular function Unexplored Instruction External symbol

Debug View Program Segmentation Strings window

IDA View-EIP Breakpoints

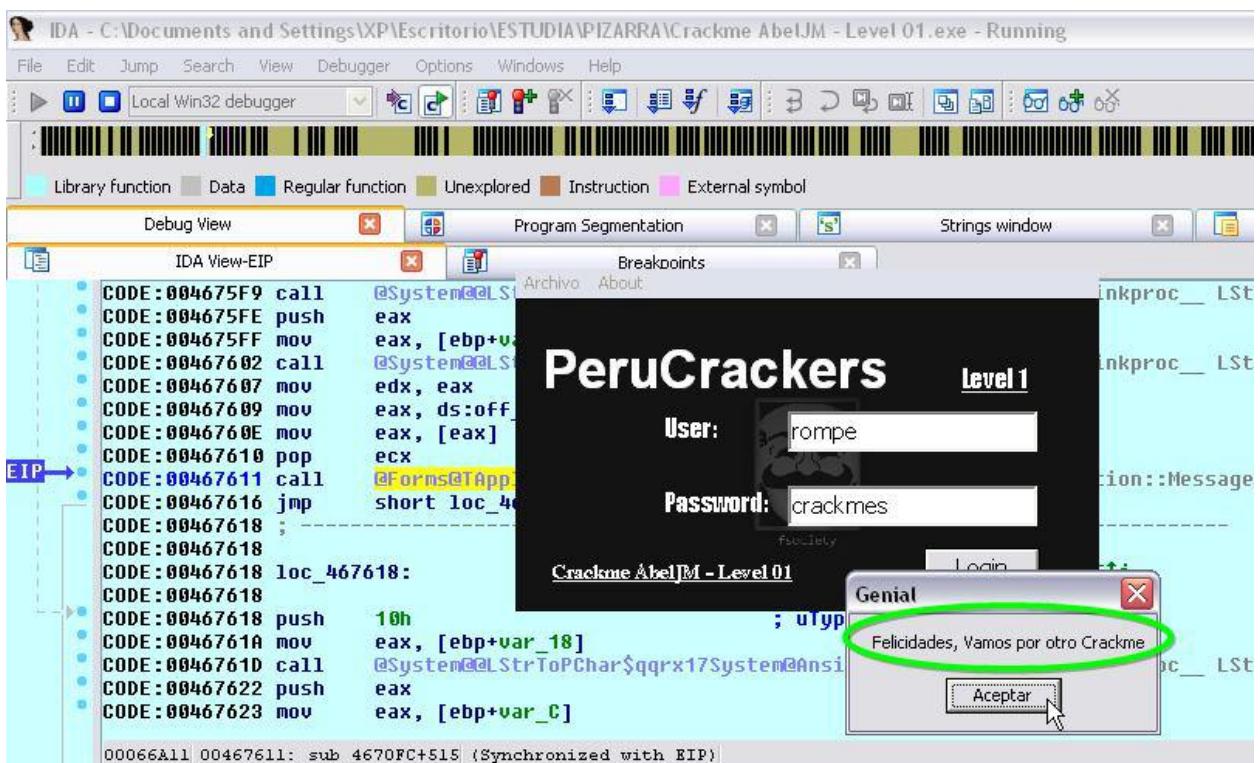
EIP →

```
CODE:004675B8 mov eax, [ebx+318h] ; this
CODE:004675BE call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675C3 mov eax, [ebp+var_1AC]
CODE:004675C9 mov edx, [ebp+var_4]
CODE:004675CC call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void) ← Red box
CODE:004675D1 jnz short loc_467618
CODE:004675D3 lea edx, [ebp+var_1B0] ← Red box
CODE:004675D9 mov eax, [ebx+32Ch] ; this
CODE:004675DF call @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004675E4 mov eax, [ebp+var_1B0]
CODE:004675EA mov edx, [ebp+var_8]
CODE:004675ED call @System@LStrCmp$qqrv ; System:: linkproc LStrCmp(void) ← Red box
CODE:004675F2 jnz short loc_467618
CODE:004675F4 push 0 ; uType
CODE:004675F6 mov eax, [ebp+var_14]
CODE:004675F9 call @System@LStrToInt$qqrx17System@Ansistring ; System:: linkproc LStrToInt
CODE:004675FE push eax
CODE:004675FF mov eax, [ebp+var_10]
CODE:00467602 call @System@LStrToPChar$qqrx17System@Ansistring ; System:: linkproc LStrToPChar

000669F4 004675F4: sub_4670FC+4F8 (Synchronized with EIP)
```

También nos ha dejado pasar el segundo saldo condicional “**jne**”, con lo cual la segunda “**StrCmp**” también se cumple.

Y seguimos traceando un poco más, y por fin al llegar a la address “00467611” Salta el tan ansioso y esperado mensaje de “**Good boy**”, felicitándonos de que hemos acertado.



ESTUDIANDO LA TERCERA VÍCTIMA

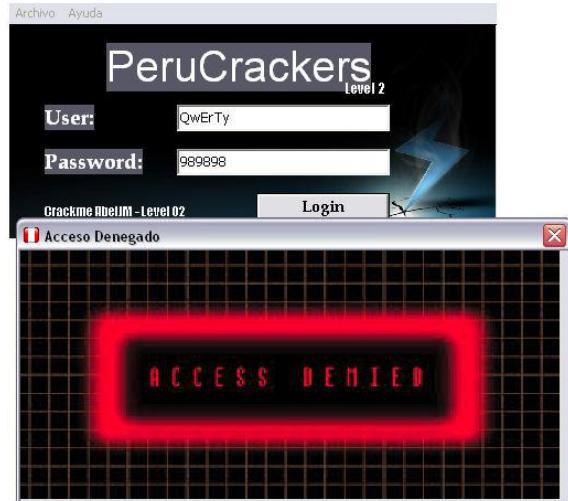
Crackme_AbelJM_Level-2



Crackme

AbelJM - Level
02

Lo ejecutamos y también nos pide un “User” y “Password”, ingresamos datos cualesquiera para probar suerte...



y nuevamente nos salta el mensaje de "bad boy". (desesperante.....)

Miramos las Reglas



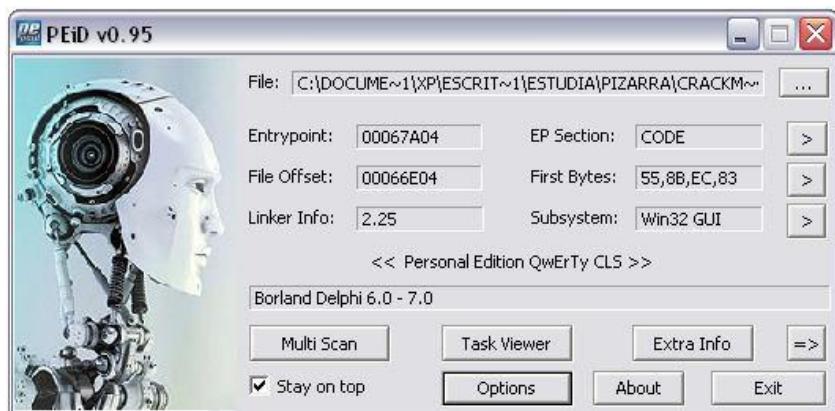
Y la indicación del autor.



Pues... una vez más también le haremos caso.

CONTINUEMOS ESTUDIANDO LA VÍCTIMA

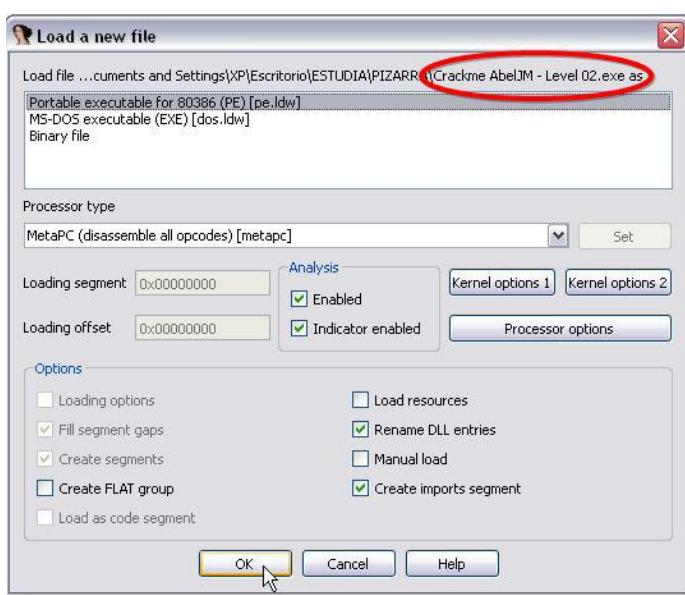
Le pasamos el detector "PEiD"



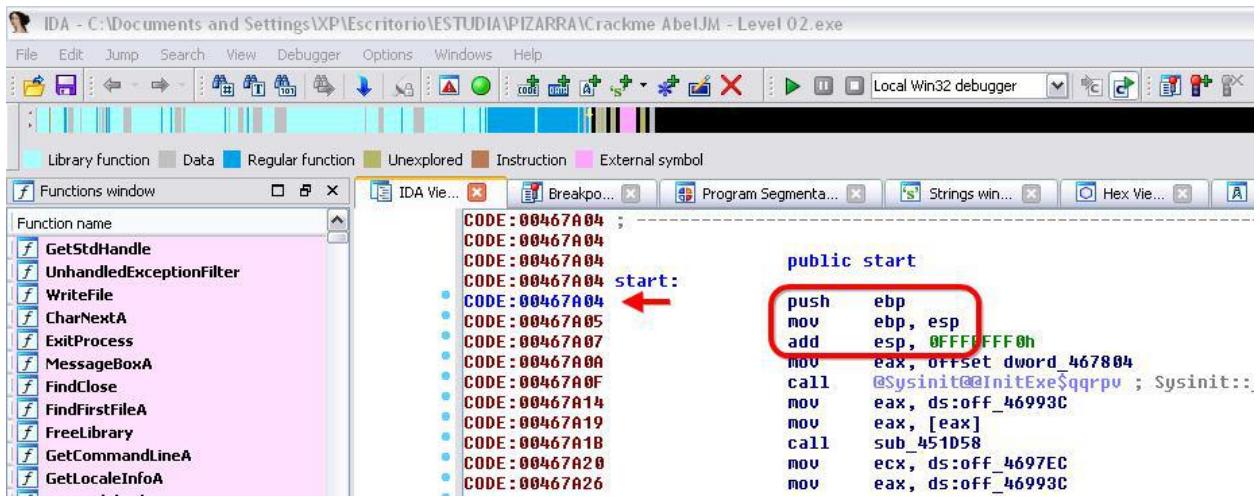
Otro compilado "Borland Delphi"

VAMOS A POR ELLA

Cargamos nuestro "Crackme_AbelJM_Level-2" con "IDA",

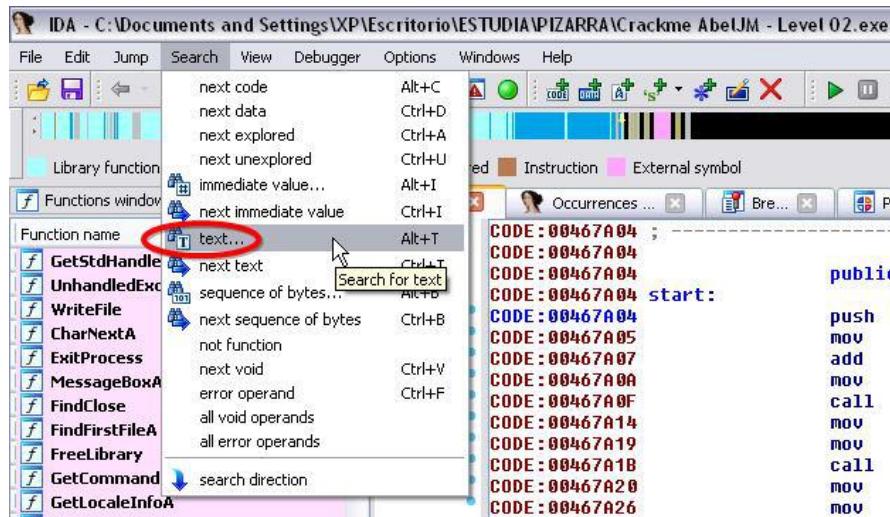


y nos encontramos parados en el Entry Point address "00467A04", (OEP)



Ahora, vamos a intentar buscar la address donde se ejecutará la primera instrucción de inicio del button "Login" de nuestra víctima, con la intención de que cuando lo clickeemos en el momento de validar, IDA pare allí.

Nos dirigimos a "Search" - "Text..."



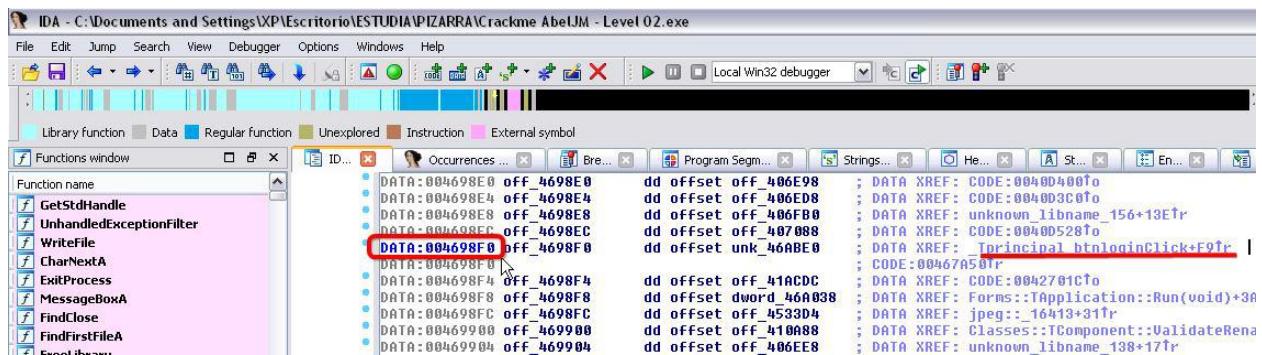
Tipeamos el caption del Button, que en nuestro caso es "Login"

```

67A04          public start
67A04 start:
67A04         push    ebp
67A05         mov     ebp, esp
67A07         add     esp, 0FFFFFFF0h
67A09         mov     eax, offset dword _h679804
67A0A
67A0A Text search (slow!)
67A1
67A1 String Login (red circle)
67A1 Parameters
67A2 Case sensitive
67A2 Regular expression
67A2 Identifier
67A4
67A5 Find all occurrences
67A5
67A5 OK (highlighted)
67A5 Cancel
67A63 call    eax, ds:off_46993C
67A68

```

Le damos a "Ok" y aparecemos aquí:



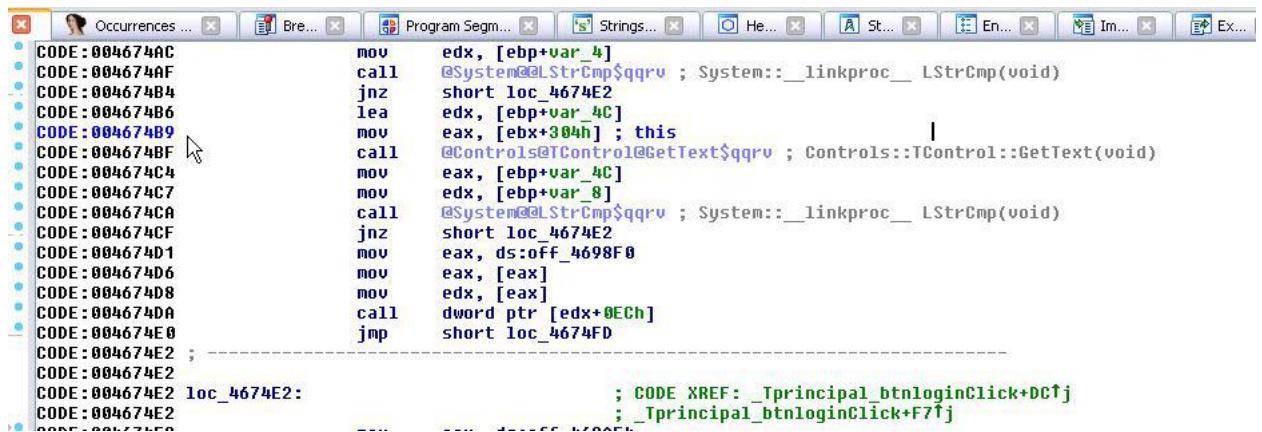
Nos colocamos sobre "_Tprincipal_btnloginClick"

```

; DATA XREF: unknown_libname_156+13E↑r
; DATA XREF: CODE:0040D528↑o
; DATA XREF: _Tprincipal_btnloginClick+F9↑r (highlighted)
; CODE:00467A50↑r
; DATA XREF: CODE:0042701C↑o

```

Le damos dos clicks izquierdo de ratón y aparecemos dentro de una subrutina



Ahora hacemos scroll hacia arriba hasta llegar al inicio de la misma y vemos que empieza en la address "004673D8"

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 02.exe

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

Functions window ID... Occurrences Bre... Program Segm... Strings... He... St... En... Im...

Function name

CODE:004673D6 align 4

===== S U B R O U T I N E =====

Attributes: bp-based frame

Tprincipal_btnloginClick proc near ; DATA XREF: CODE:00466DB7?o

CODE:004673D8 var_4C = dword ptr -4Ch

CODE:004673D8 var_4B = dword ptr -4Bh

CODE:004673D8 var_44 = dword ptr -44h

CODE:004673D8 var_40 = dword ptr -40h

CODE:004673D8 var_3C = dword ptr -3Ch

CODE:004673D8 var_38 = dword ptr -38h

CODE:004673D8 var_34 = dword ptr -34h

CODE:004673D8 var_30 = dword ptr -30h

CODE:004673D8 var_2C = dword ptr -2Ch

CODE:004673D8 var_28 = dword ptr -28h

CODE:004673D8 var_24 = dword ptr -24h

CODE:004673D8 var_20 = dword ptr -20h

CODE:004673D8 var_1C = dword ptr -1Ch

CODE:004673D8 var_18 = dword ptr -18h

CODE:004673D8 var_14 = dword ptr -14h

CODE:004673D8 var_10 = dword ptr -10h

CODE:004673D8 var_C = dword ptr -0Ch

CODE:004673D8 var_8 = dword ptr -8

CODE:004673D8 var_4 = dword ptr -4

push ebp

mov ebp, esp

add esp, 0FFFFFB4h

push ebx

Pues vamos a colocarle un "BP" por si suena la flauta, ya con un poco de suerte este sería el punto que buscamos.

Nos colocamos sobre el puntito azul de la última address "004673D8"

CODE:004673D8 var_10 = dword ptr -10h

CODE:004673D8 var_C = dword ptr -0Ch

CODE:004673D8 var_8 = dword ptr -8

CODE:004673D8 var_4 = dword ptr -4

CODE:004673D8

CODE:004673D9

CODE:004673DB

CODE:004673DE

push ebp

mov ebp, esp

add esp, 0xFFFFFB4h

push ebx

Y sobre él, le damos un solo click Izquierdo de ratón y el "BP" nos queda colocado

CODE:004673D8 var_8 = dword ptr -8

CODE:004673D8 var_4 = dword ptr -4

CODE:004673D8

CODE:004673D8

CODE:004673D9

CODE:004673DB

CODE:004673DE

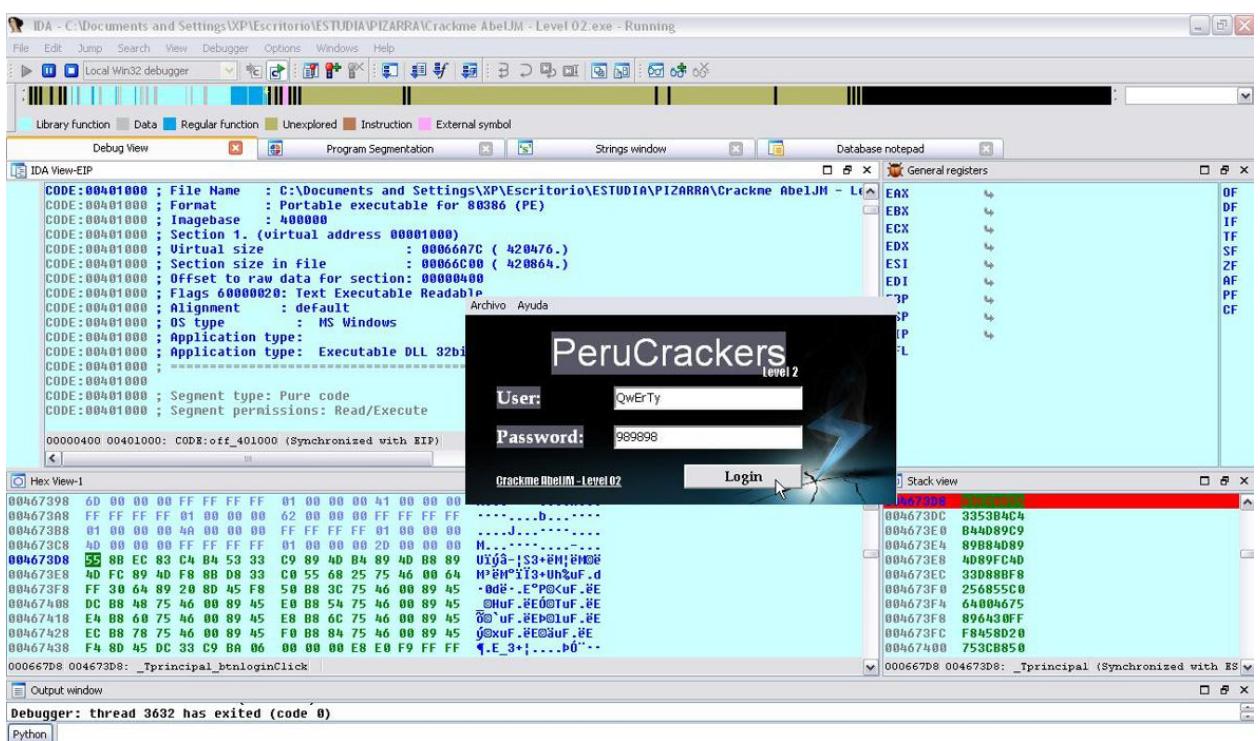
push ebp

mov ebp, esp

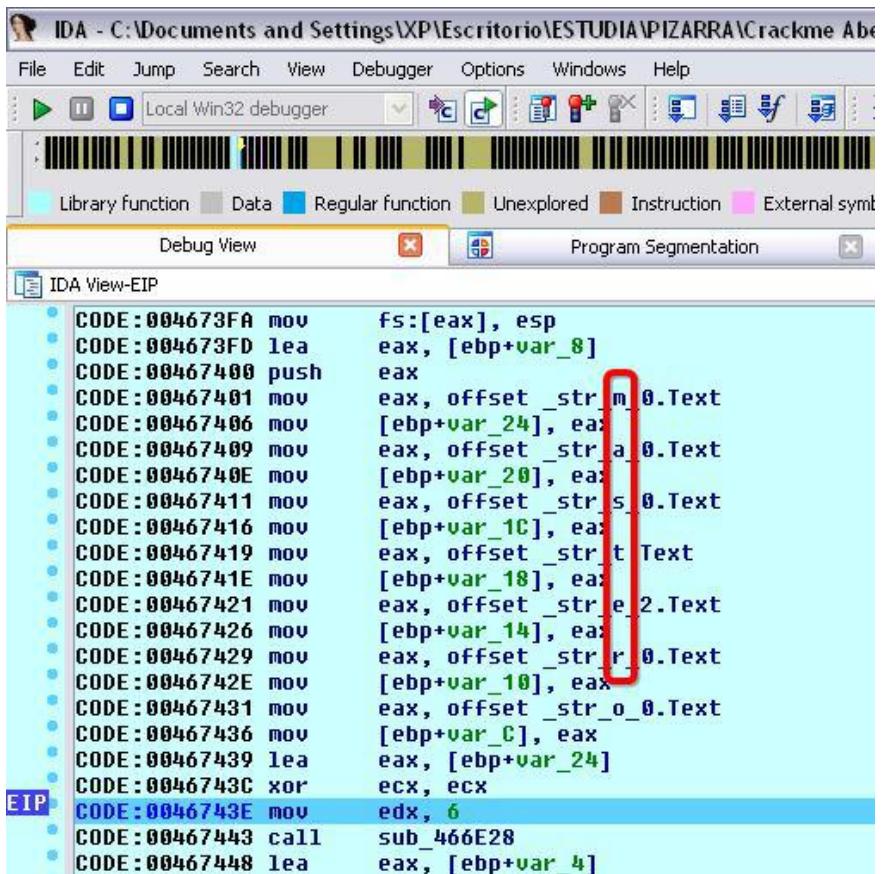
add esp, 0xFFFFFB4h

push ebx

Ahora le damos a "Start" para que arranque el debugger, rellenamos datos



Le damos a "Login" y.....biennnn, IDA ha parado en nuestro "BP" automáticamente al accionar el button "Login" , ahora vamos traceando con "F8" y con mucha atención y ojos de camaleón observamos esto



Continuamos traceando y también nos quedamos con esto

IDA View-EIP

```

CODE:00467443 call    sub_466E28
CODE:00467448 lea     eax, [ebp+var_4]
CODE:0046744B push   eax
CODE:0046744C mov    eax, offset _str_c_0.Text
CODE:00467451 mov    eax, [ebp+var_44], eax
CODE:00467454 mov    eax, offset _str_r_0.Text
CODE:00467459 mov    eax, [ebp+var_40], eax
CODE:0046745C mov    eax, offset _str_a_0.Text
CODE:00467461 mov    eax, [ebp+var_3C], eax
CODE:00467464 mov    eax, offset _str_c_0.Text
CODE:00467469 mov    eax, [ebp+var_38], eax
CODE:0046746C mov    eax, offset _str_k_0.Text
CODE:00467471 mov    eax, [ebp+var_34], eax
CODE:00467474 mov    eax, offset _str_e_2.Text
CODE:00467479 mov    eax, [ebp+var_30], eax
CODE:0046747C mov    eax, offset _str_r_0.Text
CODE:00467481 mov    eax, [ebp+var_2C], eax
CODE:00467484 mov    eax, offset _str_o_0.Text
CODE:00467489 mov    eax, [ebp+var_28], eax
EIP: CODE:0046748C lea     eax, [ebp+var_44]
CODE:0046748F xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:0046749B lea     edx, [ebp+var_48]
0006688C 0046748C: _Tprincipal_btnloginClick+84 (Synchronized with EIP)

```

Y continuamos con nuestro traceo hasta llegar a la address "0046749E" donde vemos lo que sin duda es la zona caliente...je,je,je.... Ahora me van a disculpar unos segundos, me dirijo al frigorífico abro un quinto Voll-Damm bien fresquito....., le pego un buen trago..... y aquí estoy de nuevo.....

Estamos parados en "0046749E"

IDA - C:\Documents and Settings\XP\Escritorio\ESTUDIA\PIZARRA\Crackme AbelJM - Level 02.exe

```

File Edit Jump Search View Debugger Options Windows Help
File Local Win32 debugger Program Segmentation Strings window Database notepad
Debug View X Program Segmentation X Strings window X Database notepad X
Library function Data Regular function Unexplored Instruction External symbol
EIP: CODE:0046748C xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:0046749B lea     edx, [ebp+var_48]
CODE:0046749E mov    eax, [ebx+308h] ; this
CODE:004674A4 call   @Controls@Control@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674A9 mov    eax, [ebp+var_48]
CODE:004674AC mov    edx, [ebp+var_4]
CODE:004674AF call   @System@StrCmp$qqrv ; System::__linkproc__LStrCmp(void)
CODE:004674B4 jnz   short loc_4674E2
CODE:004674B6 lea     edx, [ebp+var_4C]
CODE:004674B9 mov    eax, [ebx+30Ah] ; this
CODE:004674B8 call   @Controls@Control@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674C4 mov    eax, [ebp+var_4C]
CODE:004674C7 mov    edx, [ebp+var_8]
CODE:004674CA call   @System@StrCmp$qqrv ; System::__linkproc__LStrCmp(void)
CODE:004674CF jnz   short loc_4674E2
CODE:004674D1 mov    eax, ds:off 4698F0

```

General registers
EAX 009986DC debug030:009986DC
EBX 00991FCC debug030:00991FCC
ECX 00000000
EDX 0012F5E4 Stack[0000030]db 72h
ESI 0042C16C StdCtrls::TBuf db 61h
EDI 0012F7A8 Stack[0000030]db 63h
EBP 0012F62C Stack[0000030]db 65h
ESP 0012F5D0 Stack[0000030]db 72h
EIP 0046749E _Tprincipal_bt
EFL 00000202 db 8

Y en la imagen superior vemos muchas cosas interesantes: dos llamadas "Call" donde a su vez se encuentran dos "StrCmp" precedidas de dos saltos condicionales "jnz".

Si en la ventana "General registers" nos posicionamos simplemente con el cursor del ratón sobre "debug030:0099860C" del registro "EAX" vemos que guarda la string "cracker"

Sigamos traceando hasta la "Call" address "004674AF" y aquí parados vemos como la instrucción "mov eax, [ebp+var_48]" lo que hace es mover al registro "EAX" nuestro Password trucho "989898" en memoria.

4 View-EIP

```

CODE:0046748F xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:00467498 lea    edx, [ebp+var_48]
CODE:0046749E mov    eax, [ebx+308h]
CODE:004674A4 call   @Controls@IControl@GetText$qqrw ; this
CODE:004674A9 mov    eax, [ebp+var_8]
CODE:004674AC mov    edx, [ebp+var_4]
CODE:004674AF call   @System@LStrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004674C1 jnz   short loc_4674E2

```

General registers	
EAX 009986F0	debug030:009986F0
EBX 00991FCC	debug030:00991FCC
ECX 7E39882A	user32.dll:user32:_GetDC+16
EDX 009986DC	debug030:009986DC
ESI 0042C16C	Stdctrls::IBu
EDI 0012F748	Stack[000003D]db 39h ; 9
EIP 0012F62C	Stack[000003D]db 38h ; 8
ESP 0012F5D0	Stack[000003D]db 39h ; 9
EIP 004674AF	_Tprincipal_b@0
EFL 00000202	db 8
	db 0A6h ; a

y la siguiente instrucción "mov edx, [ebp+var_4]" lo que hace es guardar en memoria dentro del registro "EDX" el Password correcto "cracker"

```

:0046748F xor    ecx, ecx
:00467491 mov    edx, 7
:00467496 call   sub_466E28
:00467498 lea    edx, [ebp+var_48]
:0046749E mov    eax, [ebx+308h]
:004674A4 call   @Controls@IControl@GetText$qqrw ; this
:004674A9 mov    eax, [ebp+var_8]
:004674AC mov    edx, [ebp+var_4]
:004674AF call   @System@LStrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
:004674B4 jnz   short loc_4674E2
:004674B6 lea    edx, [ebp+var_4C]
:004674B9 mov    eax, [ebx+304h]
:004674BF call   @Controls@IControl@GetText$qqrw ; Controls::IControl::GetText(void)
:004674C4 mov    eax, [ebp+var_4C]
:004674C7 mov    edx, [ebp+var_8]
:004674CA call   @System@LStrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
:004674CF jnz   short loc_4674E2

```

General registers	
EAX 009986F0	debug030:009986F0
EBX 00991FCC	debug030:00991FCC
ECX 7E39882A	user32.dll:user32:_GetDC+16
EDX 009986DC	debug030:009986DC
ESI 0042C16C	Stdctrls::IBu
EDI 0012F748	Stack[000003D]db 63h ; c
EIP 0012F62C	Stack[000003D]db 72h ; r
ESP 0012F5D0	Stack[000003D]db 63h ; c
EIP 004674AF	_Tprincipal_b@0
EFL 00000202	db 6Bh ; k
	db 72h ; r
	db 8
	db 16h
	db 0

En la "Call" se compararán ("StrCmp"), y finalmente el salto condicional "jnz" decidirá que si el resultado de la comparación de los valores de los registros "EAX" y "EBX" son iguales nos dejará continuar, si por el contrario son distintos nos llevará directos a la address "004674E2" donde seguro que se encuentra el mensaje "bad boy" a evitar.

Conclusión: primero valida el Password, y el Password correcto es "cracker".

Damos un solo trazoe más y ahora nos encontramos parados en el salto condicional "jnz" address "004674B4", y vemos que con que la condición no se cumple, la flecha verde de la Izquierda nos indica que nos llevará directos a "bad boy".

4 View-EIP

```

EIP → CODE:0046748F xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:00467498 lea    edx, [ebp+var_48]
CODE:0046749E mov    eax, [ebx+308h]
CODE:004674A4 call   @Controls@IControl@GetText$qqrw
CODE:004674A9 mov    eax, [ebp+var_48]
CODE:004674AC mov    edx, [ebp+var_4]
CODE:004674AF call   @System@LStrCmp$qqrv
CODE:004674B4 jnz   short loc_4674E2
CODE:004674B6 lea    edx, [ebp+var_4C]
CODE:004674B9 mov    eax, [ebx+304h]
CODE:004674BF call   @Controls@IControl@GetText$qqrw
CODE:004674C4 mov    eax, [ebp+var_4C]
CODE:004674C7 mov    edx, [ebp+var_8]
CODE:004674CA call   @System@LStrCmp$qqrv
CODE:004674D1 jnz   short loc_4698F0
CODE:004674D6 mov    eax, [eax]
CODE:004674D8 mov    edx, [eax]
CODE:004674DA call   dword ptr [edx+0ECH]
CODE:004674E2 jmp   short loc_4674FD
CODE:004674E2 ;
CODE:004674E2

```

000668B4 004674B4: _Tprincipal_btnloginClick+DC (Synchronize)

Pues vamos a engañar al código subiendo la bandera del flag "ZF 0" posicionándonos con el cursor del ratón sobre el mismo flag

```

General registers

EAX FFFFFFFF ↵
EBX 00991FCC ↵ debug@0:00991FCC
ECX 38393839 ↵
EDX 00000006 ↵
ESI 0042C16C ↵ Stdctrls::TButton::Click(void)
EDI 0012F7A8 ↵ Stack[ 000003D8]:0012F7A8
EBP 0012F62C ↵ Stack[ 000003D8]:0012F62C
ESP 0012F5D0 ↵ Stack[ 000003D8]:0012F5D0
EIP 004674B4 ↵ _Tprincipal_btnloginClick+DC
EFL 00000283

```

OF 0
DF 0
IF 1
TF 0
SE 1
ZF 0
AF 1
PF 0
CF 1

y con un golpe de barra espaciadora de teclado, subimos bandera y ahora queda con valor 1 “ZF 1”

```

General registers

EAX FFFFFFFF ↵
EBX 00991FCC ↵ debug@0:00991FCC
ECX 38393839 ↵
EDX 00000006 ↵
ESI 0042C16C ↵ Stdctrls::TButton::Click(void)
EDI 0012F7A8 ↵ Stack[ 000003D8]:0012F7A8
EBP 0012F62C ↵ Stack[ 000003D8]:0012F62C
ESP 0012F5D0 ↵ Stack[ 000003D8]:0012F5D0
EIP 004674B4 ↵ _Tprincipal_btnloginClick+DC
EFL 000002C3

```

OF 0
DF 0
IF 1
TF 0
SE 1
ZF 1
AF 0
PF 0
CF 1

Lo que hemos conseguido con este pequeño pero ingenioso cambio, es poder continuar con nuestra singladura ya que el código ahora ha interpretado que la anterior comparación era correcta, aunque en realidad no lo es.

Seguimos traceando hasta llegar a la segunda “Call” address “004674CA” y así poder ver que comparación hace allí con nuestro “User” tipeado.

Ahora con la instrucción “mov eax, [ebp+var_4C]” lo que hace es meter en memoria dentro del registro “EAX” nuestro “User” truco “QwErTy”

```

CODE:0046748F xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:00467498 lea    edx, [ebp+var_48]
CODE:0046749E mov    eax, [ebx+308h]; this
CODE:004674A0 call   @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674A9 mov    eax, [ebp+var_48]
CODE:004674AC mov    edx, [ebp+var_4]
CODE:004674AF call   @System@StrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004674B1 jnz   short loc_4674E2
CODE:004674B6 lea    edx, [ebp+var_4C]
CODE:004674B9 mov    eax, [ebx+304h]; this
CODE:004674BF call   @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674C4 mov    eax, [ebp+var_4C]
CODE:004674C7 mov    edx, [ebp+var_8]
CODE:004674C9 call   @System@StrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004674CF jnz   short loc_4674E2
CODE:004674D1 mov    eax, ds:off_4698F0

```

EAX 00998704 ↵ debug@0:00998704
EBX 00991FCC ↵ debug@0:00991FCC
ECX 7E39882A ↵ user32.dll:user32_GetText+163
EDI 0042C16C ↵ Stdctrls::TButton::Click(void)
EBP 0012F7A8 ↵ Stack[000003D8]:0012F7A8
ESP 0012F62C ↵ Stack[000003D8]:0012F62C
EIP 004674CA ↵ _Tprincipal_btnloginClick+DC
EFL 00000202

Y “mov edx, [ebp+var_8]” guarda en memoria dentro del registro “EDX” el “User” correcto que es “master”

```

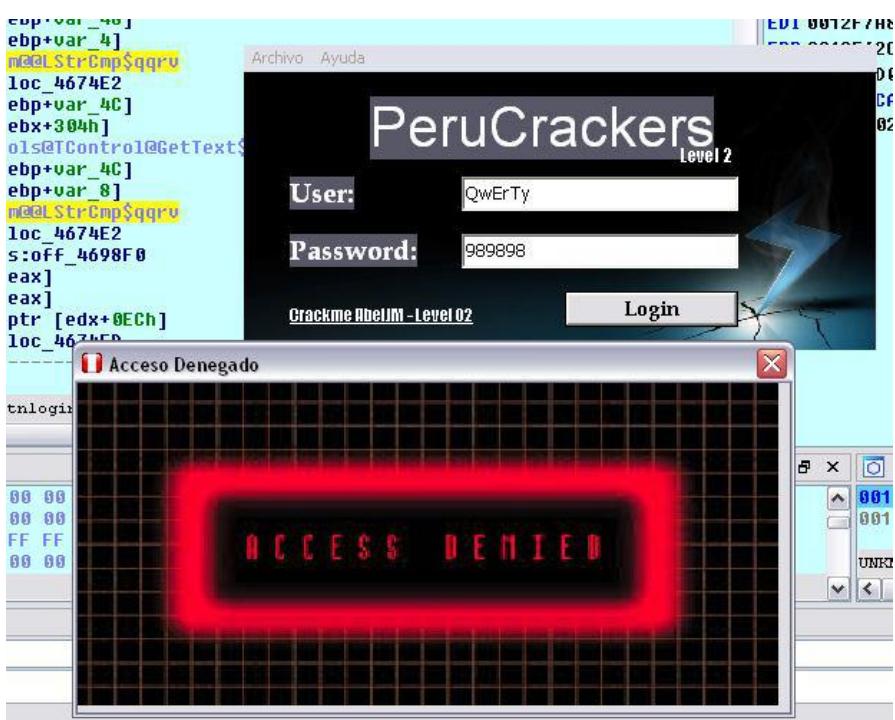
CODE:0046748F xor    ecx, ecx
CODE:00467491 mov    edx, 7
CODE:00467496 call   sub_466E28
CODE:00467498 lea    edx, [ebp+var_48]
CODE:0046749E mov    eax, [ebx+308h]; this
CODE:004674A0 call   @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674A9 mov    eax, [ebp+var_48]
CODE:004674AC mov    edx, [ebp+var_4]
CODE:004674AF call   @System@StrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004674B1 jnz   short loc_467NE2
CODE:004674B6 lea    edx, [ebp+var_4C]
CODE:004674B9 mov    eax, [ebx+304h]; this
CODE:004674BF call   @Controls@TControl@GetText$qqrv ; Controls::TControl::GetText(void)
CODE:004674C4 mov    eax, [ebp+var_4C]
CODE:004674C7 mov    edx, [ebp+var_8]
CODE:004674C9 call   @System@StrCmp$qqrv ; System::__linkproc__ LStrCmp(void)
CODE:004674CF jnz   short loc_467NE2
CODE:004674D1 mov    eax, ds:off_4698F0

```

EAX 00998704 ↵ debug@0:00998704
EBX 00991FCC ↵ debug@0:00991FCC
ECX 7E39882A ↵ user32.dll:user32_GetText+163
EDI 0042C16C ↵ Stdctrls::TButton::Click(void)
EBP 0012F7A8 ↵ Stack[000003D8]:0012F7A8
ESP 0012F62C ↵ Stack[000003D8]:0012F62C
EIP 004674CA ↵ _Tprincipal_btnloginClick+DC
EFL 00000202

Por lo tanto, parece que hemos obtenido lo que buscábamos.

Damos “Start” para que salte de nuevo el “bad boy”.



Reiniciamos el debugger, rellenamos las cajas de texto con los datos obtenidos



Y Crackmefelizmente.....solucionado.

Por fin vemos el mensaje de "Good boy"



ESTUDIANDO LA CUARTA VÍCTIMA

Crackme_AbelJM_Level-3



Lo ejecutamos, y esta vez nos pide que quitemos esta Maldita Nag !



Miramos las Reglas



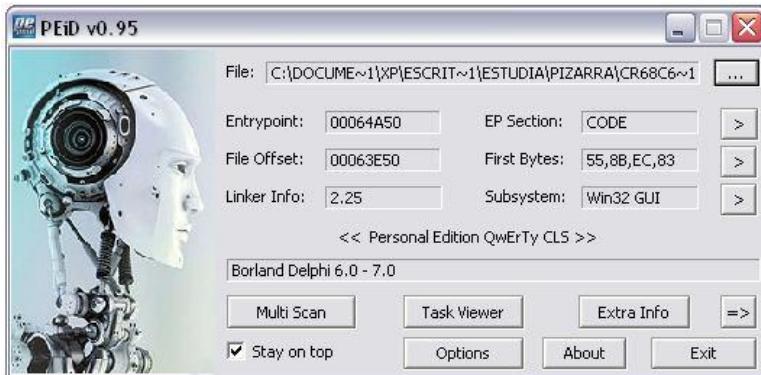
Leemos la indicación del autor.



Pues... sabéis que os digo...., que esta vez NO LE VAMOS HA HACER NI PUÑETERO CASO AL CREADOR DEL CRACKME, y vamos a solucionarlo únicamente con IDA.

CONTINUEMOS ESTUDIANDO LA VÍCTIMA

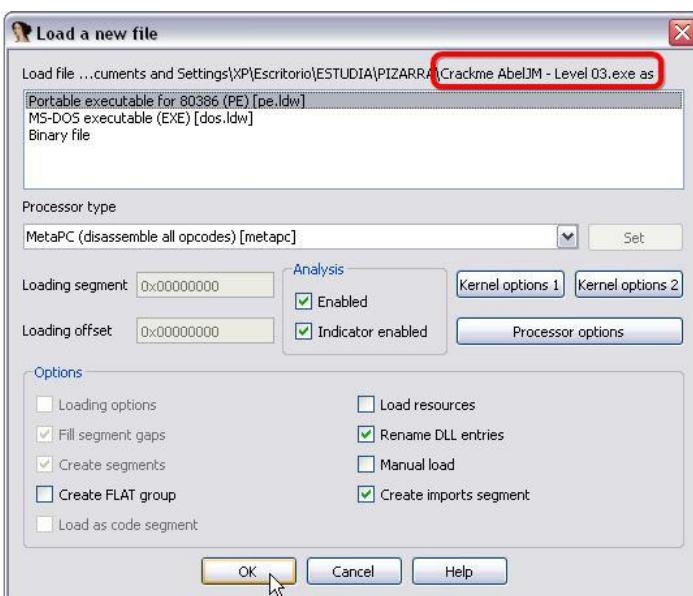
Le pasamos el detector "PEiD"



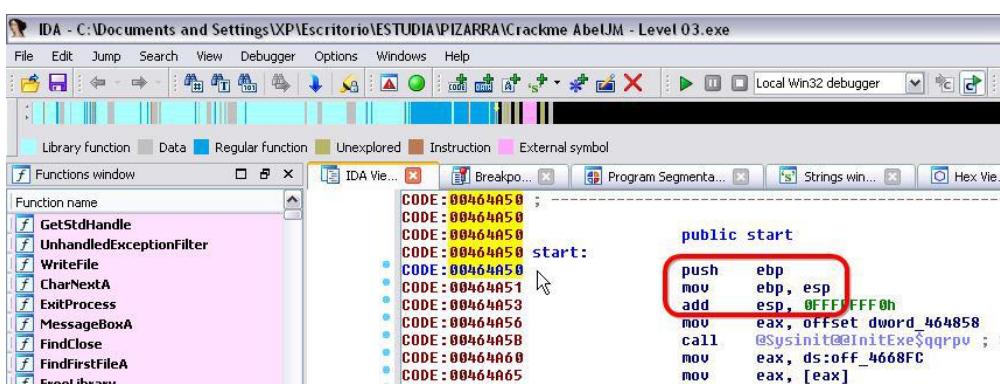
Y otroooo compilado con "Borland Delphi"

VAMOS A POR ELLA

Cargamos nuestro "Crackme_AbelJM_Level-3" con "IDA",

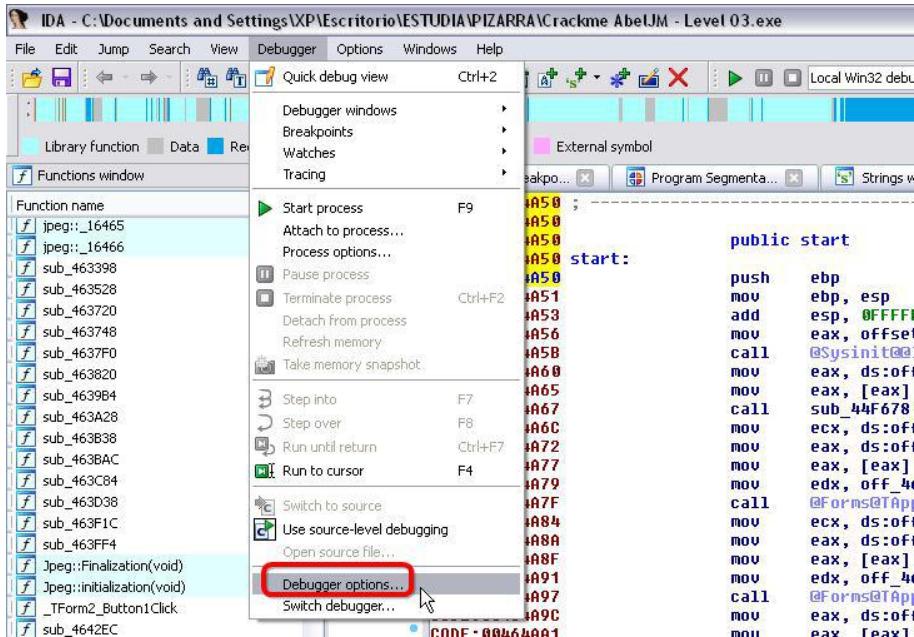


Le damos a "Ok" y nos encontramos parados en el Entry Point address "00464A50", (OEP)

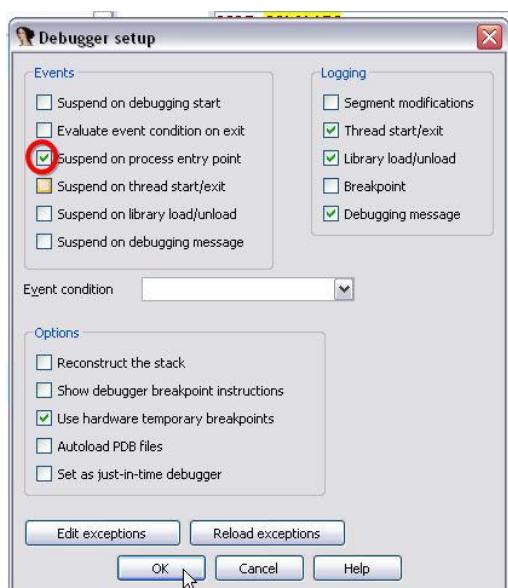


Ahora vamos a configurar el modo debugger de IDA para que cuando carguemos el Crackme, éste pare en el "Entry Point" evitando así que se dispare la "Malvada Nag" a la primera de cambio...

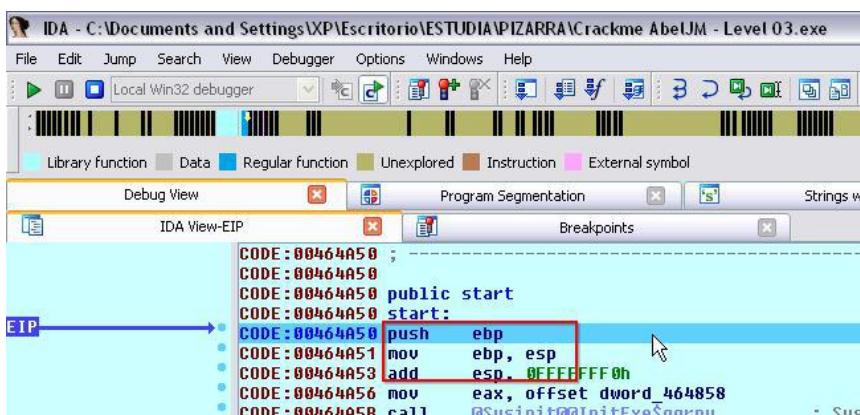
Para ello nos vamos a "Debugger - Debugger options..."



Y aquí tildamos y activamos "Suspend on process entry point"



Le damos a "OK" para validar el cambio, ejecutamos el debugger para que se inicie nuestro Crackme, y nos encontramos parados en el Entry Point (OEP)



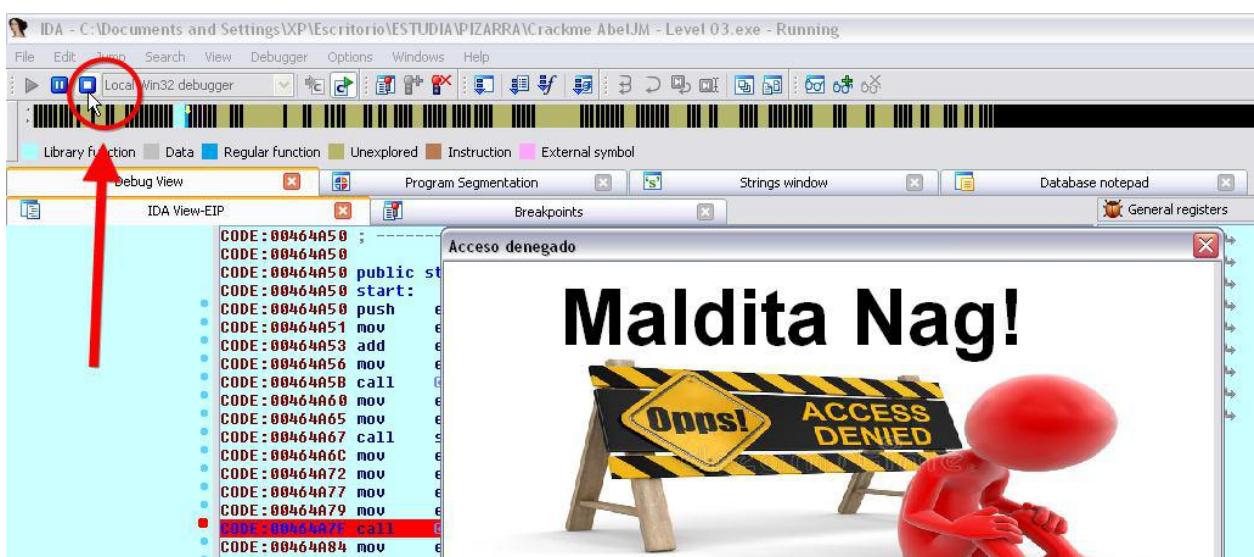
Empezamos a tracear con "F8" hasta que en la tercera Call address "00464A7F" salta la Nag



Le ponemos un "BP"

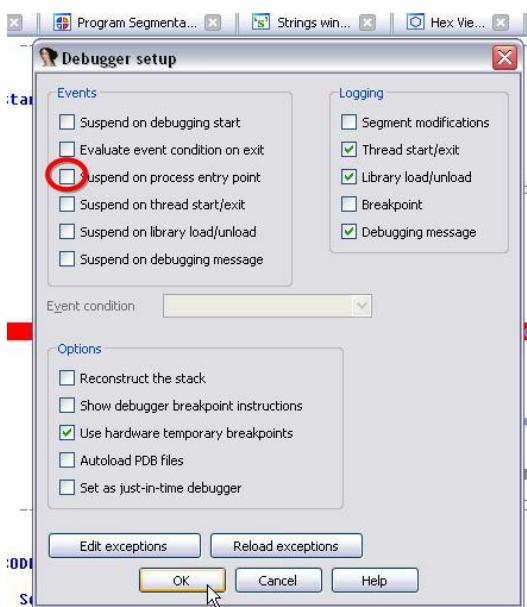


Y le damos a Salir del debugger (Finalizamos el proceso)



Con que ya tenemos un "BP" colocado (Call en la que saltó la Nag), ya podemos deshabilitar el debugger para que no pare en el "Entry Point", ya que nos interesa ir por faena y así pararemos directamente en esa "Call" cuando arranque.

Pues manos a la obra.... Volvemos a la configuración del debugger y destildamos la opción "Suspend on process entry point" para deshabilitarla y nos queda así



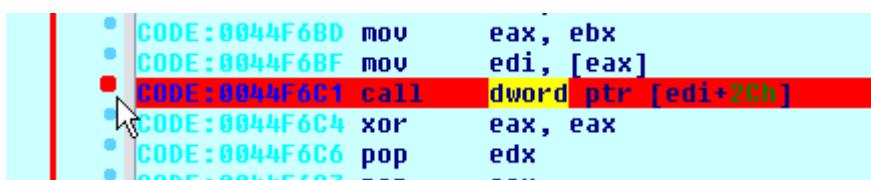
Arrancamos de nuevo en modo debugger y ahora nos encontramos parados en el "BP" que teníamos puesto



Quitamos el "BP" y entramos en la "call" con "F7". Una vez dentro continuamos traceando con "F8" hasta que vuelva a saltar la "Nag", y esto sucede en la address "0044F6C1" donde se encuentra otra llamada "call"



Y en la que ahora le ponemos el "BP"



Volvemos a salir del debugger y volvemos a entrar de nuevo, y ahora nos encontramos parados en esa "call"

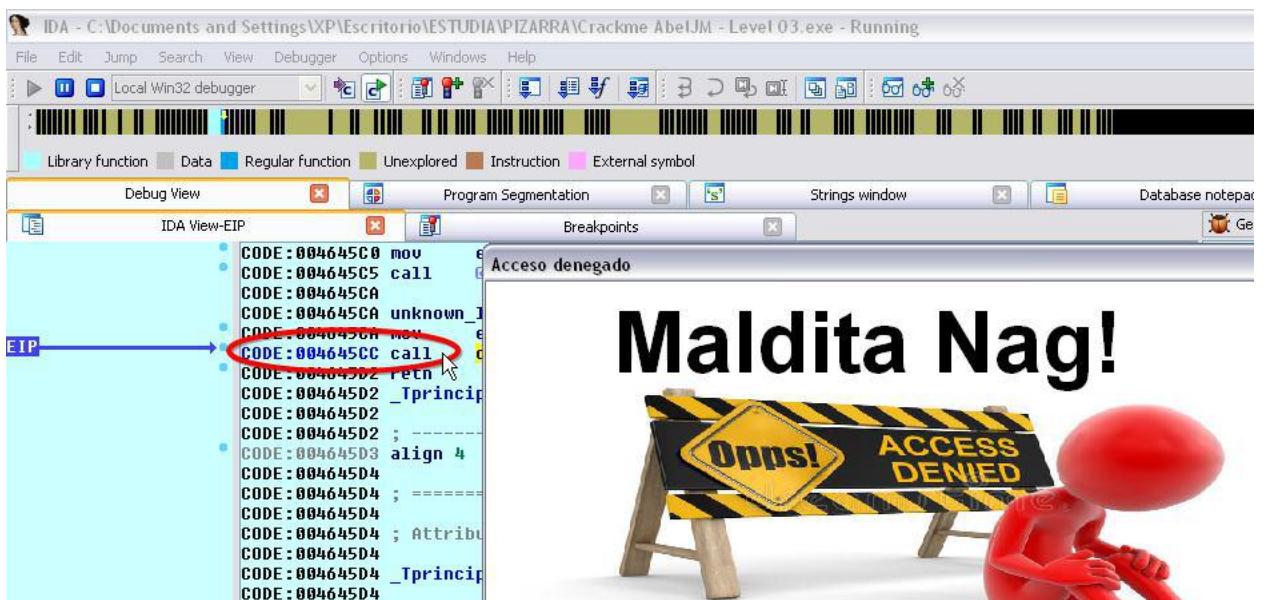
IDA View-EIP Breakpoints

```

CODE:0044F6B2 push    dword ptr fs:[eax]
CODE:0044F6B5 mov     fs:[eax], esp
CODE:0044F6B8 mov     ecx, esi
CODE:0044F6B9 or     edx, 0FFFFFFFh
CODE:0044F6B0 mov     eax, ebx
CODE:0044F6B1 mov     edi, [eax]
CODE:0044F6C1 call    dword ptr [edi+2Ch]
CODE:0044F6C4 xor     eax, eax
CODE:0044F6C6 pop    edx
CODE:0044F6C7 pop    ecx

```

Nuevamente le quitamos el “BP” y entramos con “F7”. Una vez dentro continuamos traceando con “F8” hasta que vuelva a saltar la “Nag”, y esto ahora sucede en la address “004645CC”, donde se encuentra otra “call”



Ya nos vamos acercando a nuestro objetivo, tengan un poco de paciencia.....

Volvemos a poner un “BP” a esta “call”

```

CODE:004645CA unknown_libname_805:
CODE:004645CA mov     edx, [eax]
CODE:004645CC call    dword ptr [edx+8ECh]
CODE:004645D2 retn
CODE:004645D2 _Tprincipal_FormCreate endp

```

Y Salimos del debugger, volvemos a entrar y ahora nos encontramos parados en el “BP”

```

CODE:004645C5 call    @Forms@TCustomForm@$bctr$qqrp18
CODE:004645CA
CODE:004645CA unknown_libname_805:
CODE:004645CA mov     edx, [eax]
CODE:004645CC call    dword ptr [edx+8ECh]
CODE:004645D2 retn
CODE:004645D2 _Tprincipal_FormCreate endp
CODE:004645D2 ; -----

```

Lo quitamos de nuevo, entramos con “F7” y seguimos traceando con “F8” hasta llegar a la address “0044C3E1” donde de golpe las cejas de nuestra cara se contraen al unísono con la mirada de nuestros ojos, y esto es señal inequívoca

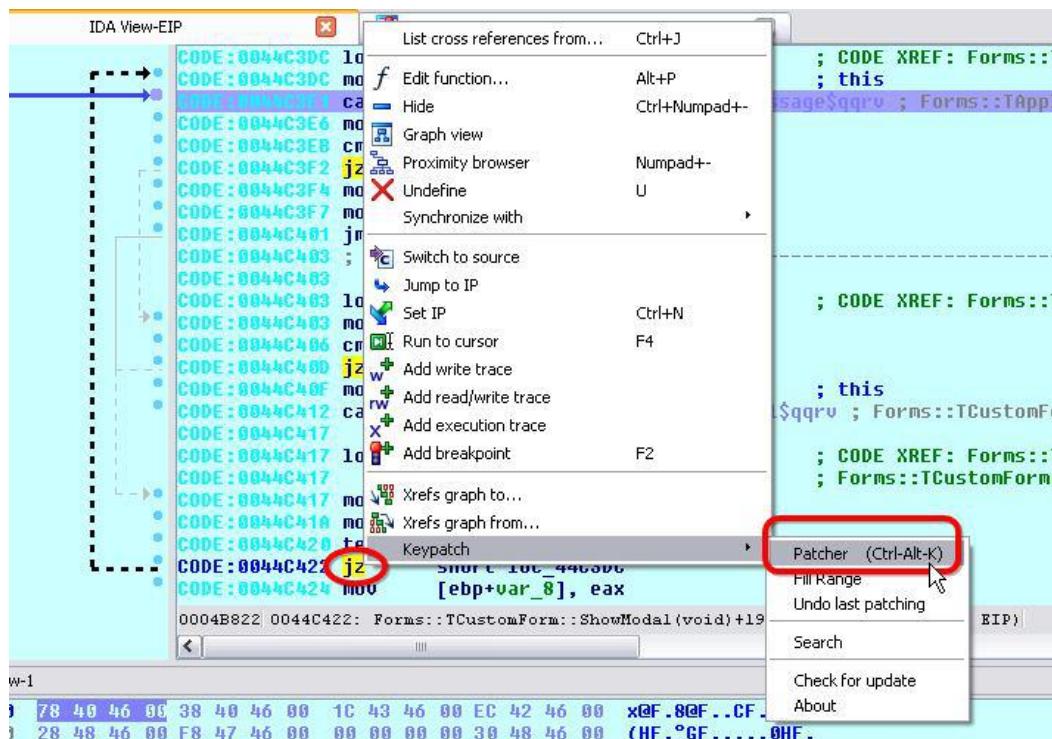
de que nuestro instinto de cracker ha despertado, dentro de nosotros hay algo que nos dice que hemos encontrado la deseada zona caliente que andábamos buscando.

Vemos una especie de Loop muy sospechoso.

```
CODE:0044C3D0 loc_44C3D0:
CODE:0044C3D0    mov    eax, ds:dword_467BB0
CODE:0044C3E1    call   @Forms::TApplication::Message$qqrv ; Forms::TApplication::HandleMessage
CODE:0044C3E6    mov    eax, ds:dword_467BB0
CODE:0044C3E8    cmp    byte ptr [eax+9Ch], 0
CODE:0044C3F2    jz    short loc_44C403
CODE:0044C3F4    mov    eax, [ebp+var_4]
CODE:0044C3F7    mov    dword ptr [eax+24Ch], 2
CODE:0044C401    jmp    short loc_44C417
CODE:0044C403    ;
CODE:0044C403    loc_44C403:
CODE:0044C403    mov    eax, [ebp+var_4]
CODE:0044C406    cmp    dword ptr [eax+24Ch], 0
CODE:0044C408    jz    short loc_44C417
CODE:0044C40F    mov    eax, [ebp+var_4]
CODE:0044C412    call   @Forms::TCustomForm@CloseModal$qqrv ; Forms::TCustomForm::CloseModal
CODE:0044C417    loc_44C417:
CODE:0044C417    mov    eax, [ebp+var_4]
CODE:0044C418    mov    eax, [eax+24Ch]
CODE:0044C420    test   eax, eax
CODE:0044C422    jz    short loc_44C3D0
CODE:0044C424    mov    [ebp+var_8], eax
```

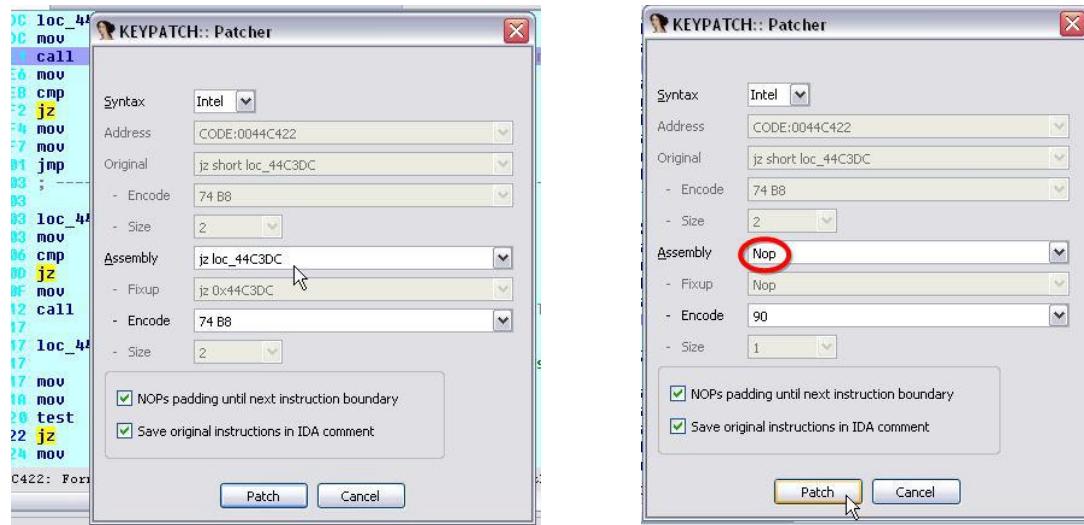
Aquí podríamos hacer varias cosas, pero yo voy a decantarme por nopear el tercer "jz" que se encuentra en la dirección "0044C422" para que el código no vuelva nunca hacia atrás.

Para ello, nos colocamos sobre el mismo salto condicional, click derecho de ratón, "Keypatch" y "Patcher"



Y cambiamos

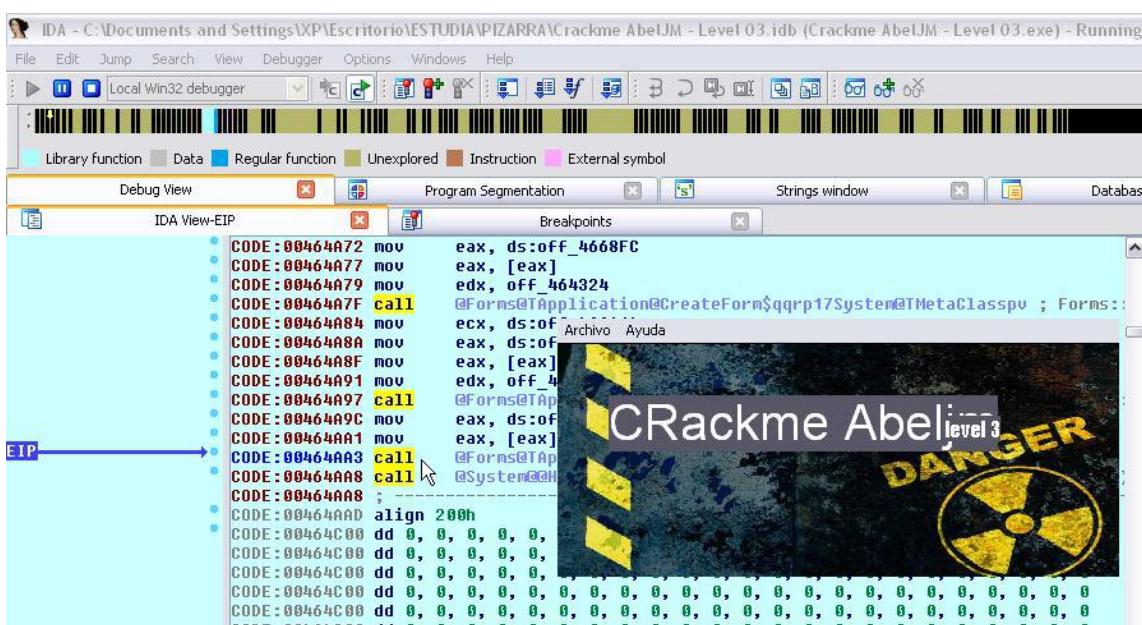
por



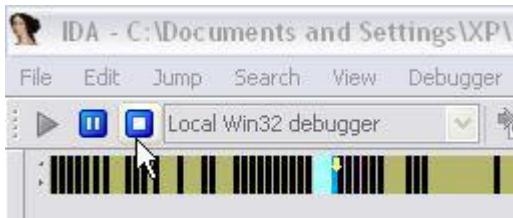
Y nos queda así

```
CODE:0044C3EB cmp    byte ptr [eax+9Ch], 0
CODE:0044C3F2 jz     short loc_44C403
CODE:0044C3F4 mov    eax, [ebp+var_4]
CODE:0044C3F7 mov    dword ptr [eax+24Ch], 2
CODE:0044C401 jmp    short loc_44C417
CODE:0044C403 ;
CODE:0044C403 loc_44C403:
CODE:0044C403 mov    eax, [ebp+var_4]
CODE:0044C406 cmp    dword ptr [eax+24Ch], 0
CODE:0044C40D jz     short loc_44C417
CODE:0044C40F mov    eax, [ebp+var_4] ; this
CODE:0044C412 call   @Forms@TCustomForm@CloseModal$qqrv ; Forms::TCustomForm::CloseModal
CODE:0044C417 loc_44C417:
CODE:0044C417 mov    eax, [ebp+var_4]
; CODE XREF: Forms::TCustomForm::`rsrc@0
CODE:0044C417 mov    eax, [eax+24Ch]
CODE:0044C41A test   eax, eax
CODE:0044C420 test   eax, eax
CODE:0044C422 nop
; Keypatch modified this from:
; jz short loc_44C3DC
; Keypatch padded NOP to next bound
CODE:0044C423 nop
CODE:0044C424 mov    [ebp+var_8], eax
CODE:0044C427 push   0 ; lParam
0004B824 0044C424: Forms::TCustomForm::ShowModal(void)+1A0 (Synchronized with EIP)
```

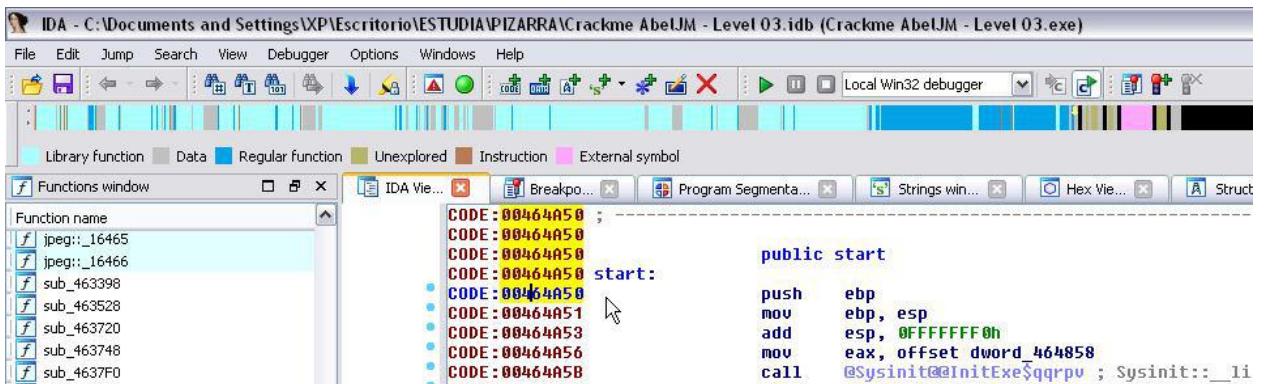
Seguimos traceando y al llegar a la address 00464AA3 observamos que por fin hemos logrado nuestro objetivo, ya no aparece la Nag



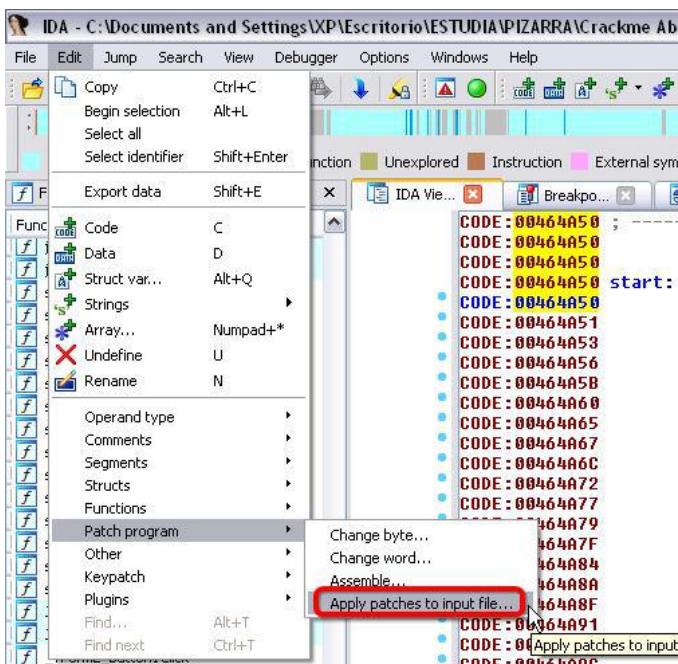
Salimos del modo debugger



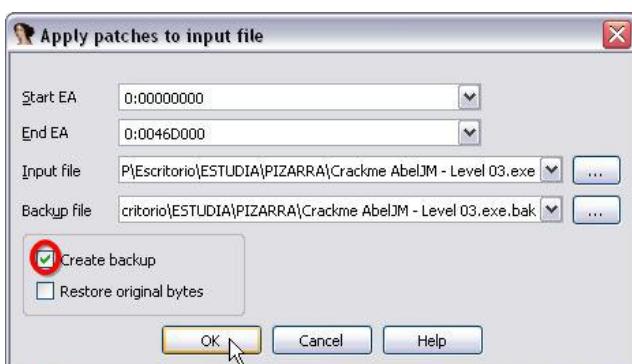
Y aparecemos en el modo "Loader"



Guardamos cambios



Tildamos "Create backup" para que nos guarde una copia del original

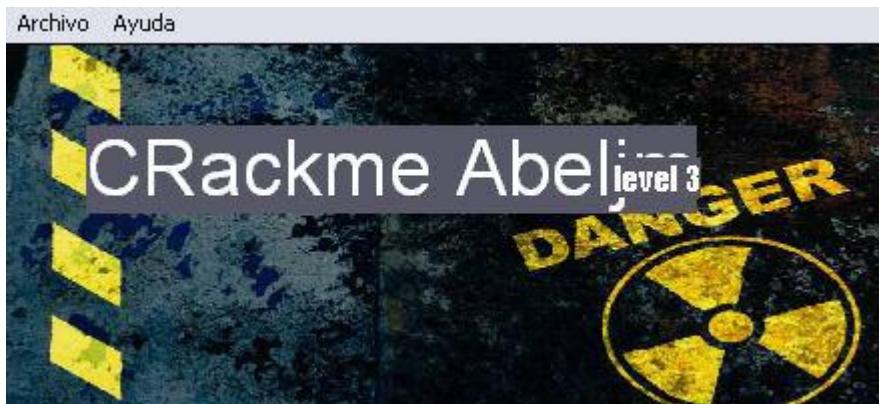


Le damos a "OK" y ya podemos salir totalmente de IDA.

Nos vamos a la ruta donde tenemos nuestro Crackme ahora modificado y la copia



Ejecutamos el "Crackme AbelJM - Level 03" modificado y observamos que funciona a la perfección.



Salimos del Crackme, le cambiamos el nombre para no confundirnos, yo le llamaré "Crackme AbelJM - Level 03_Patched", y a la copia le quitamos el formato ".bak" para tener el original, y eso es todo.

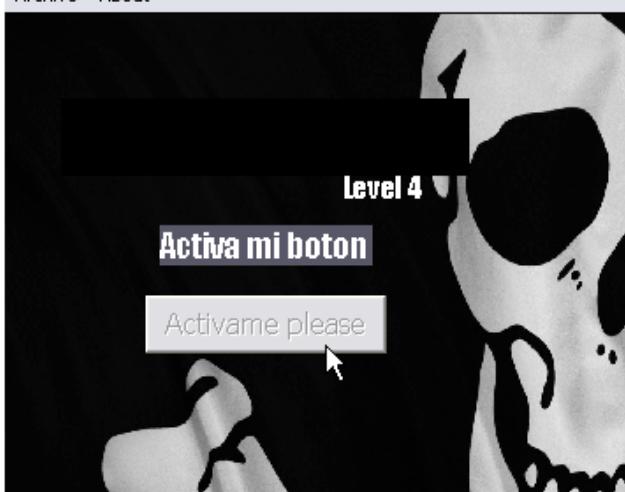


ESTUDIANDO LA QUINTA VÍCTIMA

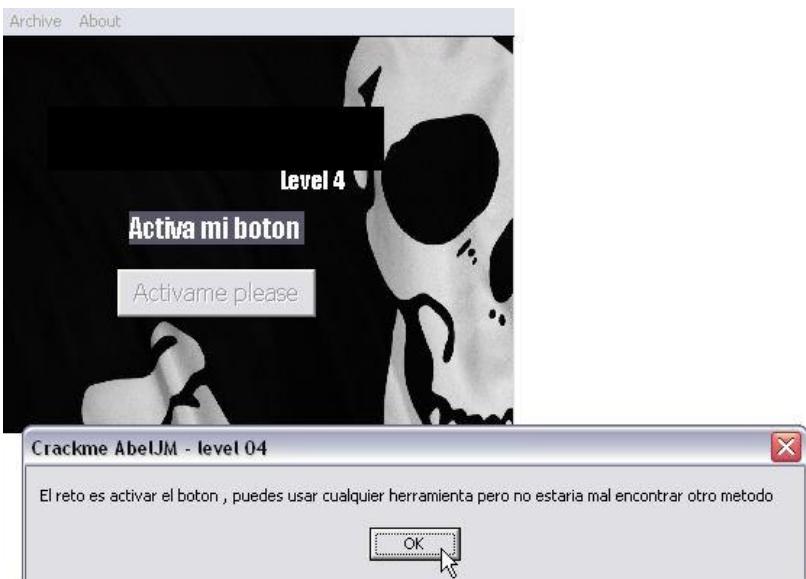
Crackme_AbelJM_Level-4



Lo ejecutamos y nos pide que por favor activemos un "button"



Miramos las Reglas y las leemos

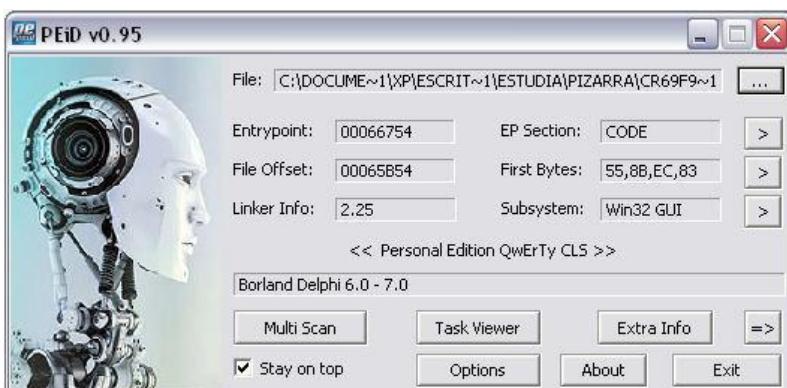


Este AbelJM es de miedo.... dice que podemos usar cualquier herramienta pero no estaría nada mal encontrar otro método.

Pues la verdad, no se a que otro método se refiere el autor, por que sin ninguna tool creo que es imposible, al menos yo no estoy preparado para solucionarlo simplemente mirándolo con el poder mental.....

CONTINUEMOS ESTUDIANDO LA VÍCTIMA

Le pasamos el detector



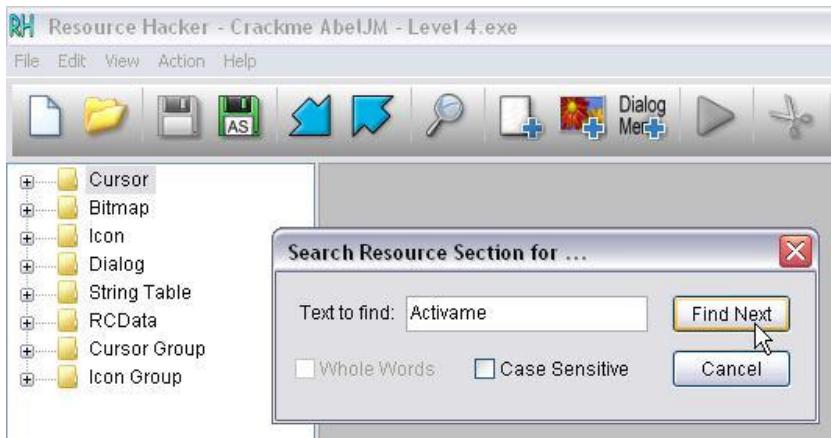
y para variar oooooootro "Borland Delphi" .

VAMOS A POR ELLA

Lo abrimos con el "Resource Hacker"



Buscamos una palabra clave "Activame"



Y aparecemos aquí

```
2857 end
2858 object Button1: TButton
2859   Left = 80
2860   Top = 160
2861   Width = 137
2862   Height = 33 ↑
2863   Caption = 'Activame please'
2864   Enabled = False _____
2865   Font.Charset = DEFAULT_CHARSET
2866   Font.Color = clWindowText
2867   Font.Height = -16
2868   Font.Name = 'Tahoma'
2869   Font.Style = []
2870   ParentFont = False
2871   TabOrder = 0
2872   OnClick = Button1Click
2873 end
```

Cambiamos directamente el "False" por "True"

```

2857 end
2858 object Button1: TButton
2859   Left = 80
2860   Top = 160
2861   Width = 137
2862   Height = 33
2863   Caption = 'Activame please'
2864   Enabled = True
2865   Font.Charset = DEFAULT_CHARSET
2866   Font.Color = clWindowText
2867   Font.Height = -16
2868   Font.Name = 'Tahoma'
2869   Font.Style = []
2870   ParentFont = False
2871   TabOrder = 0
2872   OnClick = Button1Click
2873 end

```

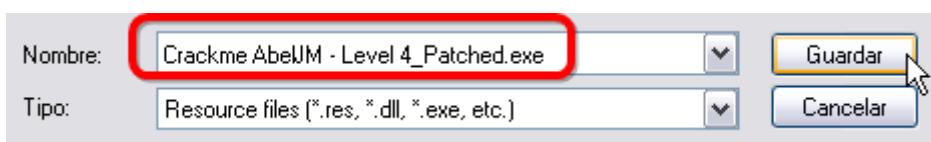
Compilamos



Guardamos como



Cambiamos el nombre. Yo lo guardo como "Crackme AbelJM-Level 4_Patched.exe"



Salimos del "Resource Hacker". Nos dirigimos a la ruta donde hemos guardado nuestra fechoría



lo ejecutamos y vemos con satisfacción que el "button" ha quedado activado



Lo pulsamos y trabajo conseguido....



ESTUDIANDO LA SEXTA Y ÚLTIMA VÍCTIMA

Crackme_AbelJM_Level-5



Crackme
AbelJM - Level
5

Lo ejecutamos y también leemos con atención las indicaciones del autor.



CONTINUEMOS ESTUDIANDO LA VÍCTIMA

Ya paso de pasarle el detector de ejecutables, por que seguro que también está compilado con "Borland Delphi"

VAMOS A POR ELLA

Como buenos observadores, vemos que al ejecutar el crackme, automáticamente se ha creado en la misma ruta un archivo ".ini" con título "MiConfiguracion"



Pues lo primero que se me ocurre es mirar que hay dentro de ese archivo. Lo abrimos como "Bloc de notas" y aquí tenemos su contenido.

A screenshot of a 'MiConfiguracion - Bloc de notas' window. The menu bar includes Archivo, Edición, Formato, Ver, and Ayuda. The main text area contains the following configuration data:

```
[Usuario]
Name=mal
Password=cracker
[Registro]
User=Usuario no Registrado
```

Más fácil, imposible.....

Lo cambiamos directamente por

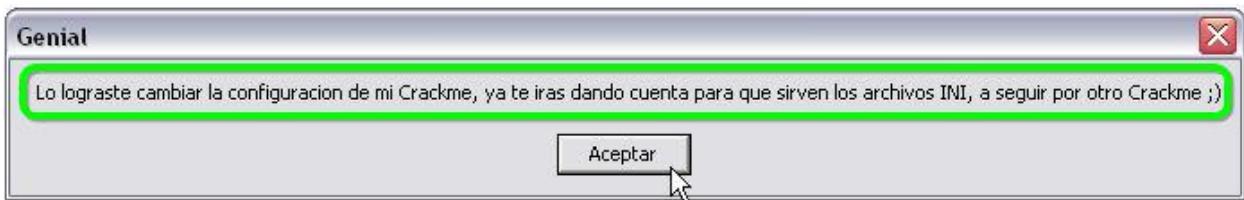
A screenshot of a 'MiConfiguracion - Bloc de notas' window. The menu bar includes Archivo, Edición, Formato, Ver, and Ayuda. The main text area contains the following configuration data:

```
[usuario]
Name=QwErTy
Password=CracksLatinos
[Registro]
User=Usuario Registrado
```

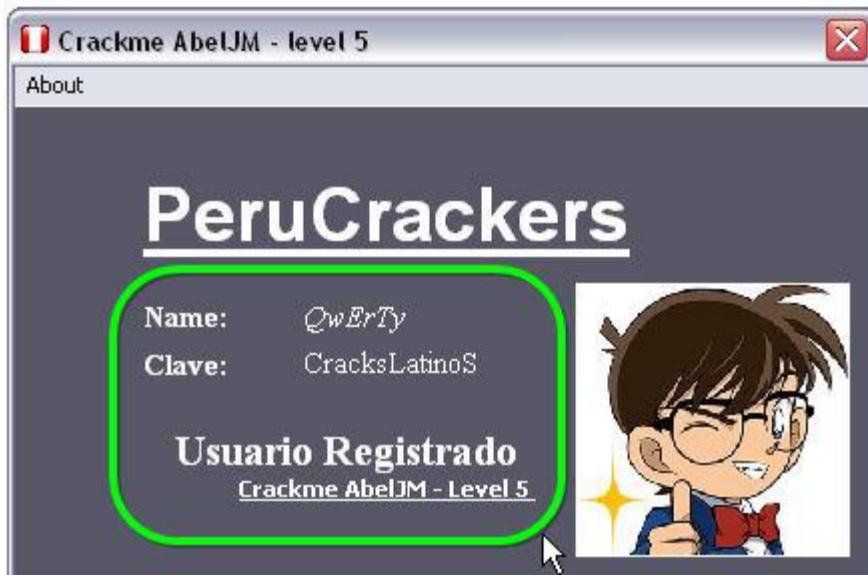
Guardamos los cambios conservando su nombre original



Abrimos nuestro Crackme y.....

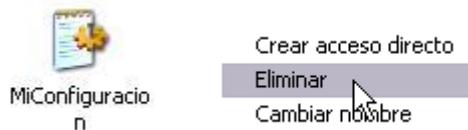


Le damos a "Aceptar" y....

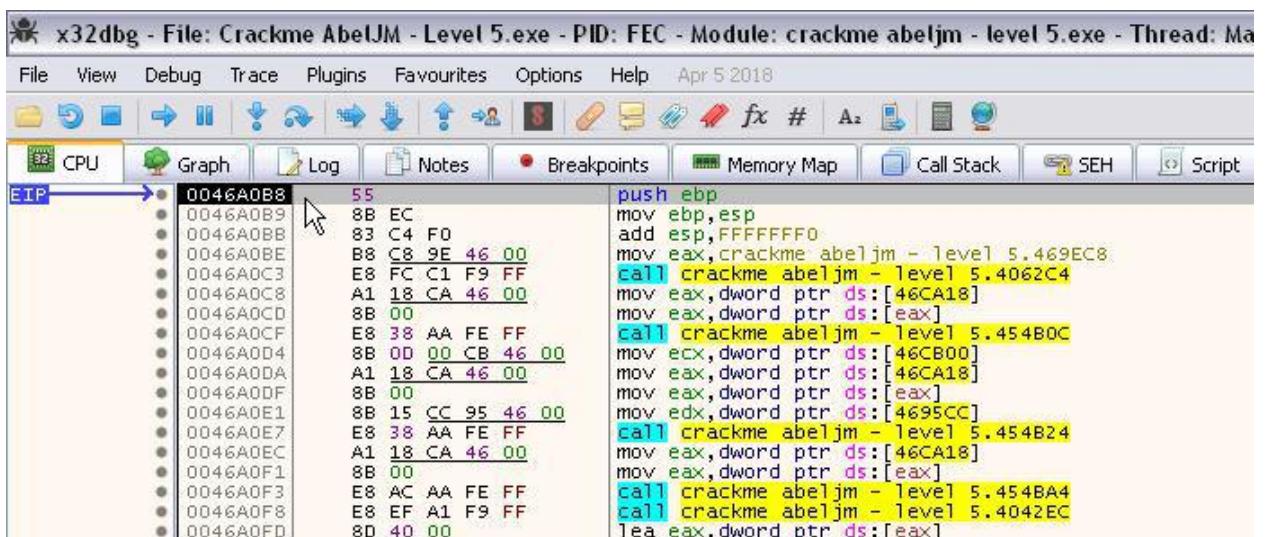


Ahora vamos a intentar modificar las entrañas del ejecutable para que cuando cree el archivo ".ini", lo haga directamente con nuestros datos, y nos muestre que estamos Registrados.

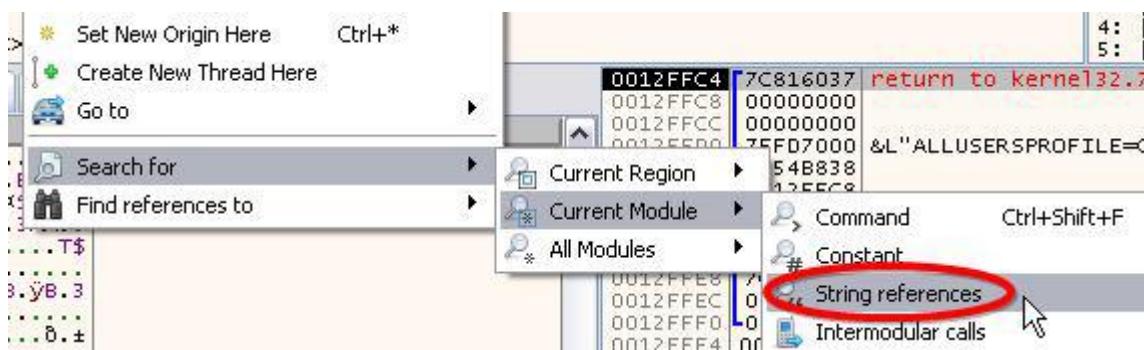
Eliminamos el archivo "MiConfiguracion.ini" ,



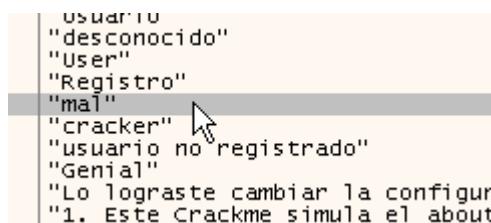
Cargamos el Crackme con el "x32dbg", y aparecemos aquí:



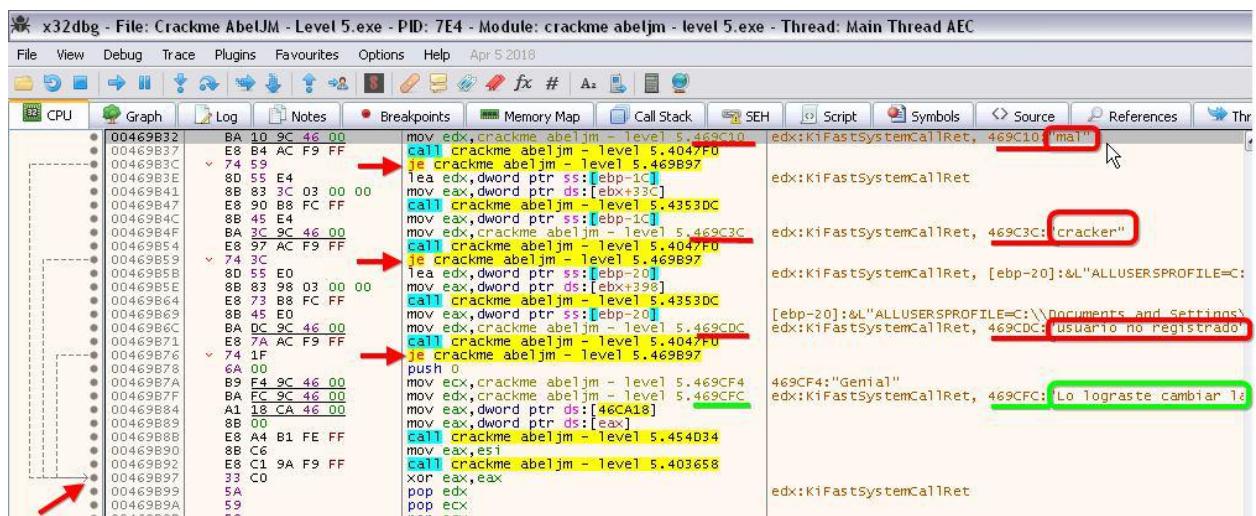
Buscamos las "String references"



Nos posicionamos sobre "mal",



Dos clicks Izquierdo de ratón y aquí estamos



En la imagen superior vemos muchas cosas interesantes:

En "469C10" guarda la string "mal"

En "469C3C" guarda la string "cracker"

En "469CDC" guarda la string "usuario no registrado"

En "469CFC" guarda la string "Lo lograste cambiar la configuración.bla,bla,bla"

Tambien vemos tres saltos condicionales "je" unas instrucciones "lea", "mov" y unas "call"

Resumiendo, estamos en la zona caliente....donde se comparará que si las Strigs "mal" y "cracker" no han sufrido cambios, nos llevarán directos a la zona de "bad boy", pero si han sufrido cambios nos llevarán a "Good boy".

Ahora hacemos "scroll" hacia arriba y vemos lo siguiente:

Breakpoints Memory Map Call Stack SEH Script Symbols Source References TI

```

mov eax,dword ptr ds:[eax]
call crackme abeljm - level 5.455264
mov eax,dword ptr ss:[ebp-14]
lea edx,dword ptr ss:[ebp-10]
call crackme abeljm - level 5.4089A8
lea eax,dword ptr ss:[ebp-10]
mov edx,crackme abeljm - level 5.469B08
call crackme abeljm - level 5.4046AC
mov ecx,dword ptr ss:[ebp-10]
mov dx,dword ptr ss:[ebp-10]
mov dl,1
mov eax,dword ptr ds:[4299BC]
call crackme abeljm - level 5.429A6C
mov esi,eax
mov eax,crackme abeljm - level 5.469BF4
call crackme abeljm - level 5.408910
test al,al
jne crackme abeljm - level 5.469AAC
push crackme abeljm - level 5.469C10
mov ecx,crackme abeljm - level 5.469C1C
mov edx,crackme abeljm - level 5.469C2C
mov eax,esi
mov edi,dword ptr ds:[eax]
call dword ptr ds:[edi+4]
push crackme abeljm - level 5.469C3C
mov ecx,crackme abeljm - level 5.469C4C
mov edx,crackme abeljm - level 5.469C2C
mov eax,esi
mov edi,dword ptr ds:[eax]
call dword ptr ds:[edi+4]
push crackme abeljm - level 5.469C60
mov ecx,crackme abeljm - level 5.469C80
mov edx,crackme abeljm - level 5.469C90
mov eax,esi
mov edi,dword ptr ds:[eax]

```

469BF4:"Miconfiguracion.ini"

edx:KiFastSystemCallRet, 469BD8:"MiConfiguracion.ini"

469C10:"mal"
469C1C:"Name"
edx:KiFastSystemCallRet, 469C2C:"Usuario"

469C3C:"cracker"
469C4C:"Password"
edx:KiFastSystemCallRet, 469C2C:"Usuario"

469C60:"Usuario no Registrado"
469C80:"User"
edx:KiFastSystemCallRet, 469C90:"Registro"

El nombre que tomará el archivo ".ini" (MiConfiguracion), y después los tres datos que contendrá dicho archivo por defecto:

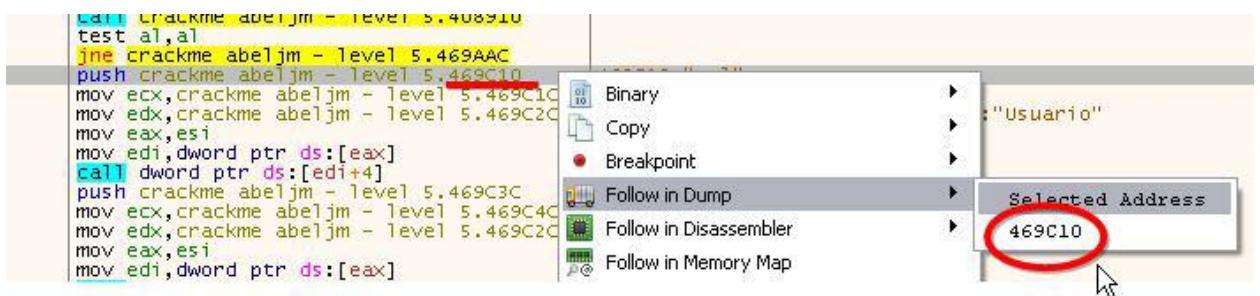
"mal"

"cracker"

"Usuario no Registrado"

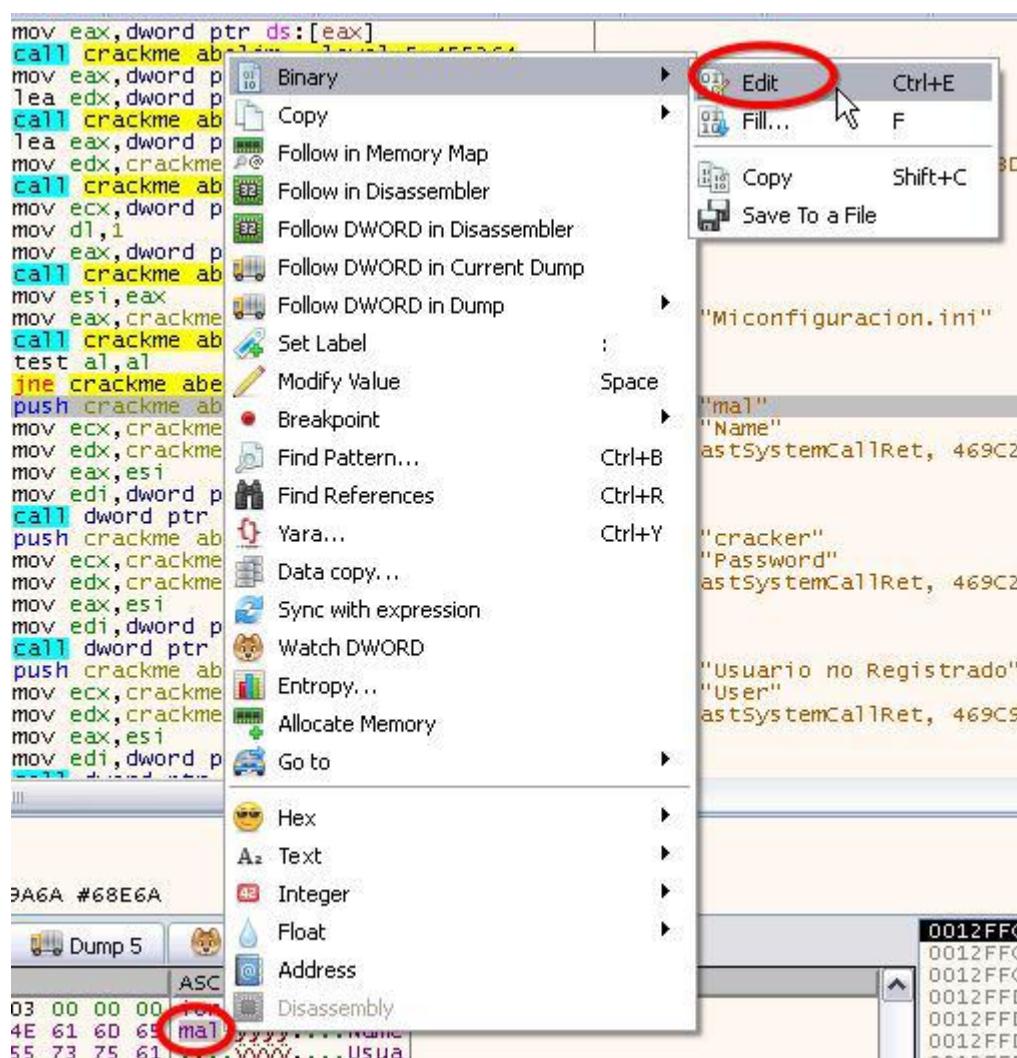
Vamos a cambiar esos valores de la siguiente manera:

Nos colocamos sobre "469C10"

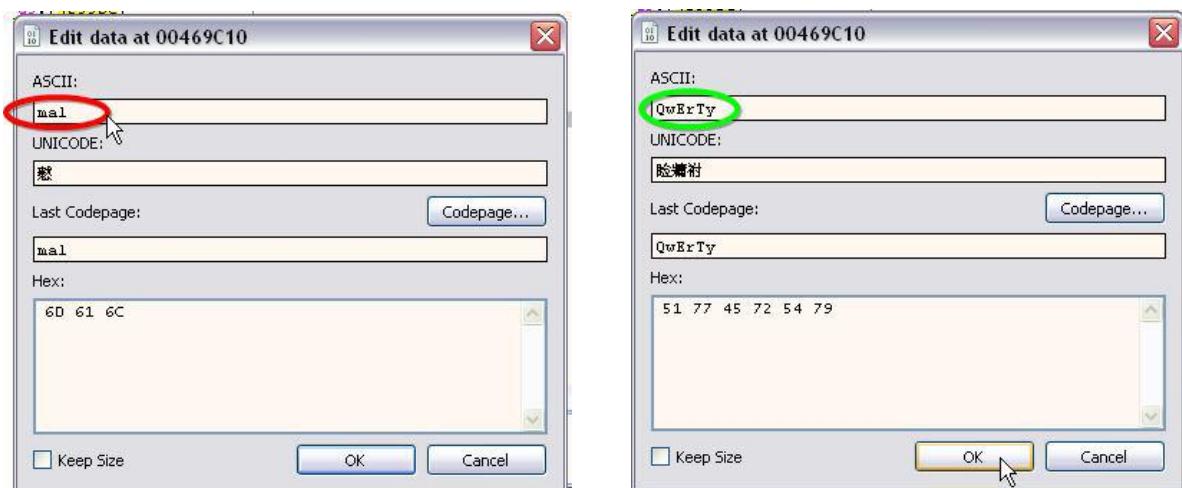


Click derecho de ratón y "Follow in Dump" - "469C10" para que nos muestre en la ventana "Dump" lo que guarda dentro y efectivamente vemos la palabra "mal".

En el mismo "Dump" marcamos toda la palabra "mal", y posicionados sobre ella, click derecho de ratón y "binary"- "Edit"



Y lo editamos cambiando "mal" por "QwErTy"



Una vez realizado el cambio, observamos en rojo el binario

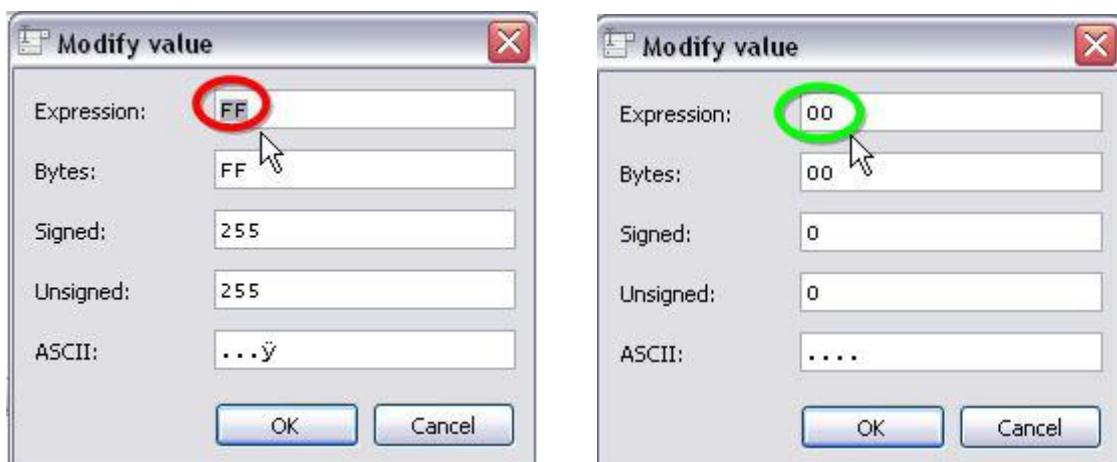
Address	Hex	ASCII
00469C00	69 6F 6E 2E 69 6E 69 00	ion.ini.yyyy....
00469C10	51 77 45 72 54 79 FF FF	QwErTy...Name
00469C20	00 00 00 00 FF FF FF FFUsua
00469C30	72 69 6F 00 FF FF FF FF	rio.yyyy....crac
00469C40	6B 65 72 00 FF FF FF FF	ker.yyyy....Pass
00469C50	77 6F 72 64 00 00 00 00	word....yyyy....
00469C60	55 73 75 61 72 69 6F 20	65 67 69 73 Usuario no Regis
00469C70	74 72 61 64 6F 00 00 00	trado....yyyy....
00469C80	55 73 65 72 00 00 00 00	User....yyyy....
00469C90	52 65 67 69 73 74 72 6F	Registro....yyyy....
00469CA0	08 00 00 00 64 65 73 63	69 64 6F 00desconocido.
00469CBA	FF FF FF FF 04 00 00 00	6E 61 6D 65 00 00 00 00 yyyy....name....
00469CC0	FF FF FF FF 04 00 00 00	77 77 77 77 77 77 77 77

Acto seguido vamos a limpiar los dos símbolos ASCII "ÿ" para que no nos molesten simplemente colocándonos sobre su valor binario "FF" "FF" y cambiándolos por "00" "00".

Nos colocamos sobre el primer "FF"

Address	Hex	ASCII
00469C00	69 6F 6E 2E 69 6E 69 00	ion.ini.ÿÿÿ...
00469C10	51 77 45 72 54 79 FF FF	QwErTyÿÿ...Name
00469C20	00 00 00 00 FF FF FF FF	...ÿÿÿ...Usua
00469C30	72 69 6F 00 FF FF FF FF	rio.ÿÿÿ...crac
00469C40	6B 65 72 00 FF FF FF FF	ker.ÿÿÿ...Pass
00469C50	77 6F 72 64 00 00 00 00	word.ÿÿÿ...
00469C60	55 73 75 61 72 69 6F 20	Usuario no Regis
00469C70	74 72 61 64 6F 00 00 00	trado.ÿÿÿ...
00469C80	55 73 65 72 00 00 00 00	User.ÿÿÿ...
00469C90	52 65 67 69 73 74 72 6F	Registro.ÿÿÿ...
00469CA0	0B 00 00 00 64 65 73 63	desconocido.
00469CBO	FF FF FF FF 04 00 00 00	ÿÿÿ...name...

Damos dos Clicks Izquierdo de ratón y cambiamos directamente su valor



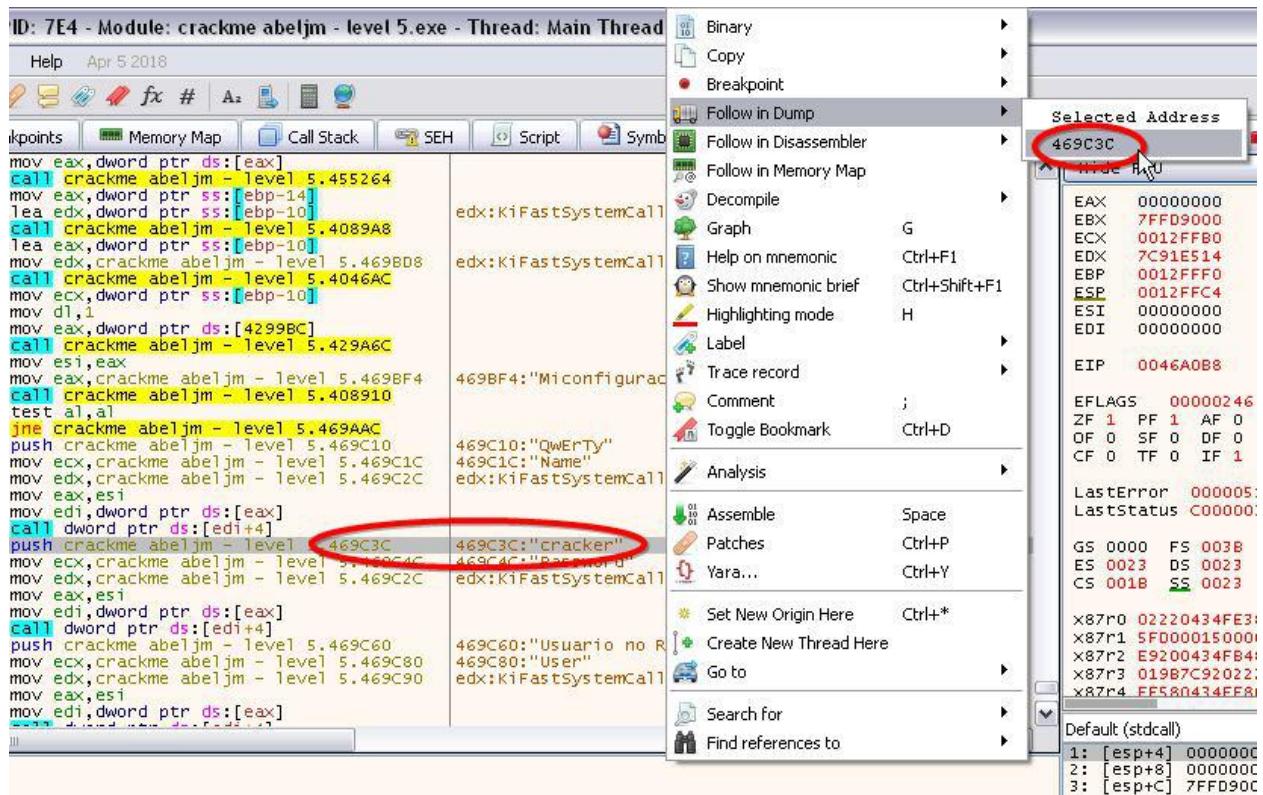
y nos queda así

Address	Hex	ASCII
00469C00	69 6F 6E 2E 69 6E 69 00	ion.ini.ÿÿÿ...
00469C10	51 77 45 72 54 79 00 00	QwErTy.ÿ...Name
00469C20	00 00 00 00 FF FF FF FF	...ÿÿÿ...Usua
00469C30	72 69 6F 00 FF FF FF FF	rio.ÿÿÿ...crac
00469C40	6B 65 72 00 FF FF FF FF	ker.ÿÿÿ...Pass
00469C50	77 6F 72 64 00 00 00 00	word.ÿÿÿ...
00469C60	55 73 75 61 72 69 6F 20	Usuario no Regis
00469C70	74 72 61 64 6F 00 00 00	trado.ÿÿÿ...
00469C80	55 73 65 72 00 00 00 00	User.ÿÿÿ...
00469C90	52 65 67 69 73 74 72 6F	Registro.ÿÿÿ...
00469CA0	0B 00 00 00 64 65 73 63	desconocido.
00469CBO	FF FF FF FF 04 00 00 00	ÿÿÿ...name...

Hacemos lo mismo con el segundo valor y nos queda así:

00469C10 "QwErTy"		
CODE:00469A6A crackme abeljm - level 5.exe:\$69A6A #68E6A		
Address	Hex	ASCII
00469C00	69 6F 6E 2E 69 6E 69 00	ion.ini.ÿÿÿ...
00469C10	51 77 45 72 54 79 00 00	QwErTy.ÿ...Name
00469C20	00 00 00 00 FF FF FF FF	...ÿÿÿ...Usua
00469C30	72 69 6F 00 FF FF FF FF	rio.ÿÿÿ...crac
00469C40	6B 65 72 00 FF FF FF FF	ker.ÿÿÿ...Pass
00469C50	77 6F 72 64 00 00 00 00	word.ÿÿÿ...
00469C60	55 73 75 61 72 69 6F 20	Usuario no Regis
00469C70	74 72 61 64 6F 00 00 00	trado.ÿÿÿ...
00469C80	55 73 65 72 00 00 00 00	User.ÿÿÿ...
00469C90	52 65 67 69 73 74 72 6F	Registro.ÿÿÿ...
00469CA0	0B 00 00 00 64 65 73 63	desconocido.
00469CBO	FF FF FF FF 04 00 00 00	ÿÿÿ...name...

Ahora hacemos la misma operación posicionamos sobre "469C3C", Click derecho de ratón y "Follow in Dump" - "469C3C" para que nos muestre en la ventana "Dump" lo que guarda dentro y aparece la palabra "cracker"



Repetimos todo los pasos para cambiar la string por defecto "cracker" por "CracksLatinoS" y nos debería quedar así

Address	Hex	ASCII
00469BEC	FF FF FF FF 13 00 00 00 4D 69 63 6F 6E 66 69 67	yyyy...Miconfig
00469BFC	75 72 61 63 69 6F 6E 2E 69 6E 69 00	uracion.ini.yyyy
00469C00	03 00 00 00 51 77 45 72 54 79 00 00QwErTy....
00469C1C	4E 61 6D 65 00 00 00 00 FF FF FF FF 07 00 00 00	Name....yyyy....
00469C2C	55 73 75 61 72 69 6F 00 FF FF FF FF 07 00 00 00	Usuario.yyyy....
00469C3C	43 72 61 63 6B 73 4C 61 74 69 6E 6F 53 08 00 00	CracksLatinoS....
00469C4C	50 61 73 73 77 6F 72 64 00 00 00 00 FF FF FF FF	Password....yyyy
00469C5C	15 00 00 00 55 73 75 61 72 69 6F 20 6E 6F 20 52Usuario no R
00469C6C	65 67 69 73 74 72 61 64 6F 00 00 00 FF FF FF FF	egistrado....yyyy
00469C7C	04 00 00 00 55 73 65 72 00 00 00 00 FF FF FF FFUser....yyyy
00469C8C	08 00 00 00 52 65 67 69 73 74 72 6F 00 00 00 00Registro....
00469C9C	FF FF FF FF 0B 00 00 00 64 65 73 63 6F 6E 6F 63	yyyy....desconocido
00469CA0	C0 C4 CF 00 FF FF FF 04 00 00 00 70 C1 77 77 FF FF FFname.....password

Y lo mismo de lo mismo para cambiar la string por defecto "Usuario no Registrado" por "Usuario Registrado" y también nos debería quedar así

Address	Hex	ASCII
00469C00	69 6F 6E 2E 69 6E 69 00 FF FF FF FF 03 00 00 00	ion.ini.yyyy....
00469C10	51 77 45 72 54 79 00 00 04 00 00 00 4E 61 6D 65	QwErTy.....Name
00469C20	00 00 00 00 FF FF FF FF 07 00 00 00 55 73 75 61yyyy....Usua
00469C30	72 69 6F 00 FF FF FF FF 07 00 00 00 43 72 61 63	rio.yyyy....Crac
00469C40	6B 73 4C 61 74 69 6E 6F 53 08 00 00 50 61 73 73	ksLatinos...Pass
00469C50	77 6F 72 64 00 00 00 00 FF FF FF FF 15 00 00 00	word....yyyy....
00469C60	55 73 75 61 72 69 6F 20 52 65 67 69 73 74 72 61	Usuario Registrado
00469C70	64 6F 00 00 00 00 00 00 FF FF FF FF 04 00 00 00	do....yyyy....
00469C80	55 73 65 72 00 00 00 00 FF FF FF FF 08 00 00 00	User....yyyy....
00469C90	52 65 67 69 73 74 72 6F 00 00 00 00 FF FF FF FF	Registro....yyyy
00469CA0	0B 00 00 00 64 65 73 63 6F 6E 6F 63 69 64 6F 00desconocido
00469CB0	FF FF FF FF 04 00 00 00 6E 61 6D 65 00 00 00 00	yyyy....name....
00469CC0	FF FF FF FF 02 00 00 00 70 C1 77 77 FF FF FF	password

Con los tres valores cambiados, vamos a poner un "BP", por ejemplo en la address "00469A5C"

```

00469A3E 8D 45 F0          lea eax,dword ptr ss:[ebp-10]
00469A41 BA D8 9B 46 00    mov edx,crackme_abeljm - level 5.469BD8
00469A46 E8 61 AC FF      call crackme_abeljm - level 5.4046AC
00469A4B 8B 40 F0          mov eax,dword ptr ss:[ebp-10]
00469A4E B2 01             mov d1,1
00469A50 E8 12 00 FC FF    mov eax,dword ptr ds:[4299BC]
00469A55 E8 12 00 FC FF    call crackme_abeljm - level 5.429A6C
00469A5C B8 F4 9B 46 00    mov eax,esi
00469A61 E8 AA EE F9 FF    call crackme_abeljm - level 5.408910
00469A66 84 C0             test al,al
00469A68 75 42             jne crackme_abeljm - level 5.408910
00469A6A 68 10 9C 46 00    push crackme_abeljm - level 5.469C10
00469A6B B9 1C 9C 46 00    mov ecx,crackme_abeljm - level 5.469C1C
00469A74 BA 2C 9C 46 00    mov edx,crackme_abeljm - level 5.469C2C
00469A79 8B C6             mov eax,esi
00469A7B 8B 38             mov edi,dword ptr ds:[eax]

```

Le damos a "run" para que arranque el Crackme y nos encontramos parados en nuestro "BP"

```

00469A50 A1 BC 99 42 00    mov eax,dword ptr ds:[4299BC]
00469A55 E8 12 00 FC FF    call crackme_abeljm - level 5.429A6C
00469A5A B8 F0             mov eax,esi
00469A61 E8 AA EE F9 FF    call crackme_abeljm - level 5.408910
00469A66 84 C0             test al,al
00469A68 75 42             jne crackme_abeljm - level 5.408910

```

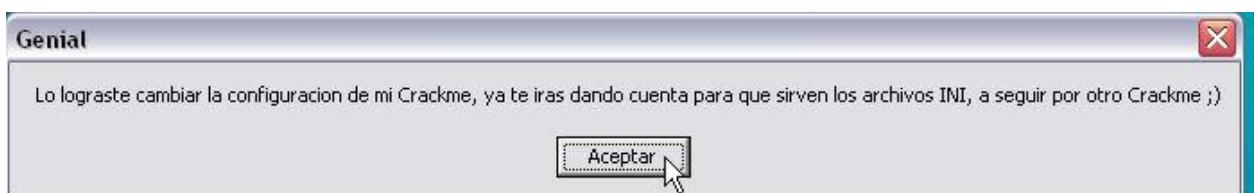
Vamos traceando y observando....

```

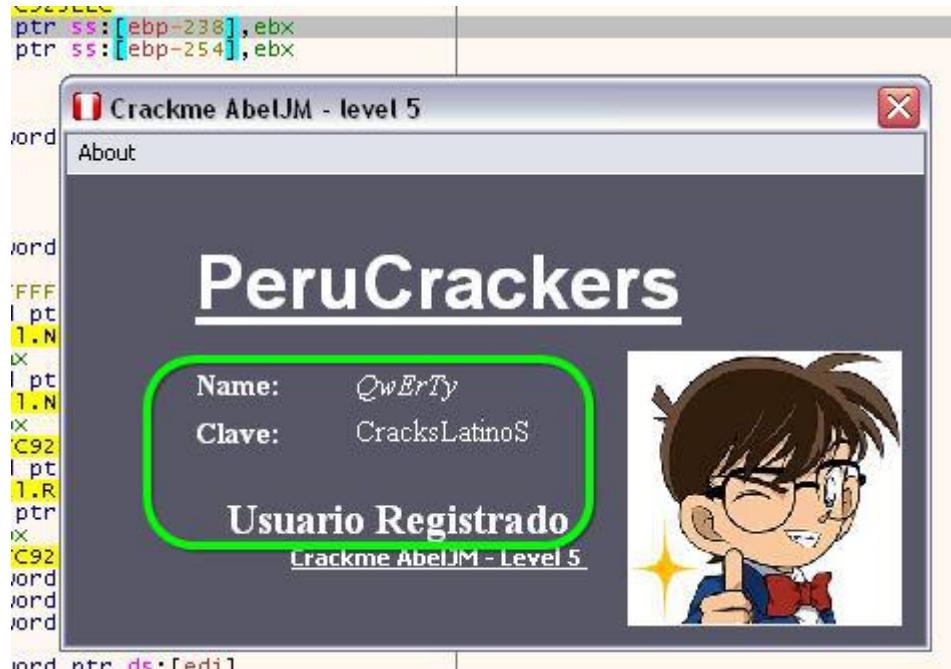
00469A50 FF 17             call dword ptr ds:[edi]
00469A55 FC               mov edx,dword ptr ss:[ebp-4]
00469A57 8B 55 FC          mov eax,dword ptr ds:[ebp+18]
00469A5A 00 00 00 00        call crackme_abeljm - level 5.43540C
00469B00 E8 07 B9 FC FF    mov edx,dword ptr ss:[ebp-8]
00469B05 8B 55 F8          mov eax,dword ptr ds:[ebp+33C]
00469B08 8B 83 3C 03 00 00  call crackme_abeljm - level 5.43540C
00469B0E E8 F9 B8 FC FF    mov edx,dword ptr ss:[ebp-C]
00469B13 8B 55 F4          mov eax,dword ptr ds:[ebp+98]
00469B16 8B 83 98 03 00 00  call crackme_abeljm - level 5.43540C
00469B1C E8 E8 B8 FC FF    mov eax,dword ptr ds:[ebp+18]
00469B21 8D 55 E8          lea edx,dword ptr ss:[ebp-18]
00469B24 8B 83 18 03 00 00  mov eax,dword ptr ds:[ebp+318]
00469B2A E8 AD B8 FC FF    call crackme_abeljm - level 5.4353DC
00469B2F 8B 45 E8          mov eax,dword ptr ss:[ebp-18]
00469B32 BA 10 9C 46 00     mov edx,crackme_abeljm - level 5.469C10
00469B37 E8 B4 AC F9 FF    call crackme_abeljm - level 5.4047F0
00469B3C 74 59             je crackme_abeljm - level 5.469B97
00469B3E 8D 55 E4          lea edx,dword ptr ss:[ebp-1C]
00469B41 8B 83 3C 03 00 00  mov eax,dword ptr ds:[ebp+33C]
00469B47 E8 B0 B8 FC FF    call crackme_abeljm - level 5.4353DC
00469B4C 8B 45 E4          mov eax,dword ptr ss:[ebp-1C]
00469B4F BA 2C 9C 46 00    mov edx,crackme_abeljm - level 5.469C3C
00469B54 E8 97 AC F9 FF    call crackme_abeljm - level 5.4047F0
00469B59 74 3C             je crackme_abeljm - level 5.469B97
00469B5B 8D 55 E0          lea edx,dword ptr ss:[ebp-20]
00469B5E 8B 83 98 03 00 00  mov eax,dword ptr ds:[ebp+398]
00469B64 E8 73 B8 FC FF    call crackme_abeljm - level 5.4353DC
00469B69 8B 45 E0          mov eax,dword ptr ss:[ebp-20]
00469B6C BA DC 9C 46 00    mov edx,crackme_abeljm - level 5.469CDC
00469B71 E8 7A AC F9 FF    call crackme_abeljm - level 5.4047F0
00469B76 74 1F             je crackme_abeljm - level 5.469B97
00469B78 6A 00             push 0
00469B7A B9 F4 9C 46 00    mov ecx,crackme_abeljm - level 5.469CF4
00469B7F BA FC 9C 46 00    mov edx,crackme_abeljm - level 5.469CF0
00469B84 A1 18 CA 46 00    mov eax,dword ptr ds:[46CA18]
00469B89 8B 00             mov eax,dword ptr ds:[eax]
00469B8B E8 A4 B1 FE FF    call crackme_abeljm - level 5.454D34
00469B90 8B C6             mov eax,esi

```

Y seguimosss traceando.... Y nos salta la siguiente ventana



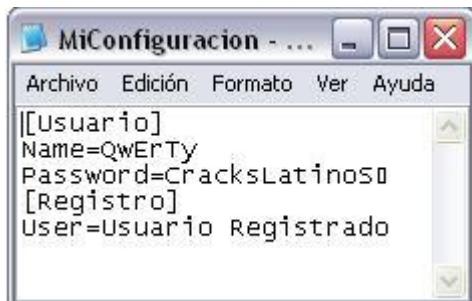
Aceptamos y....seguimos traceando hasta que por fin



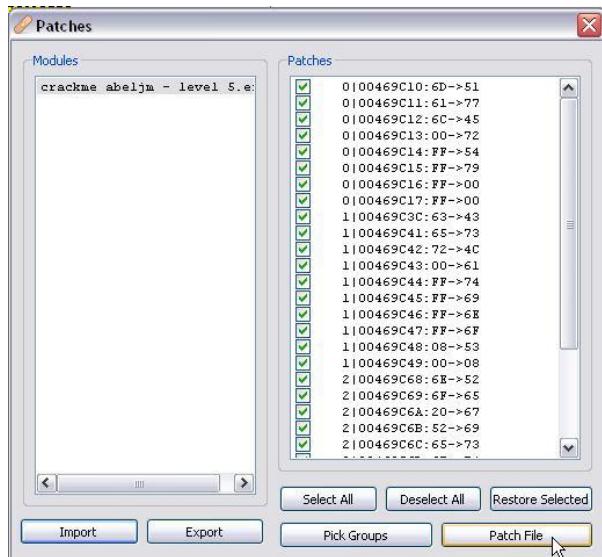
Miramos el contenido del archivo automáticamente generado "MiConfiguracion.ini"



Y comprobamos que realmente tiene los datos perfectamente modificados



Ya solo queda guardar nuestro parche



Le cambiamos el Nombre



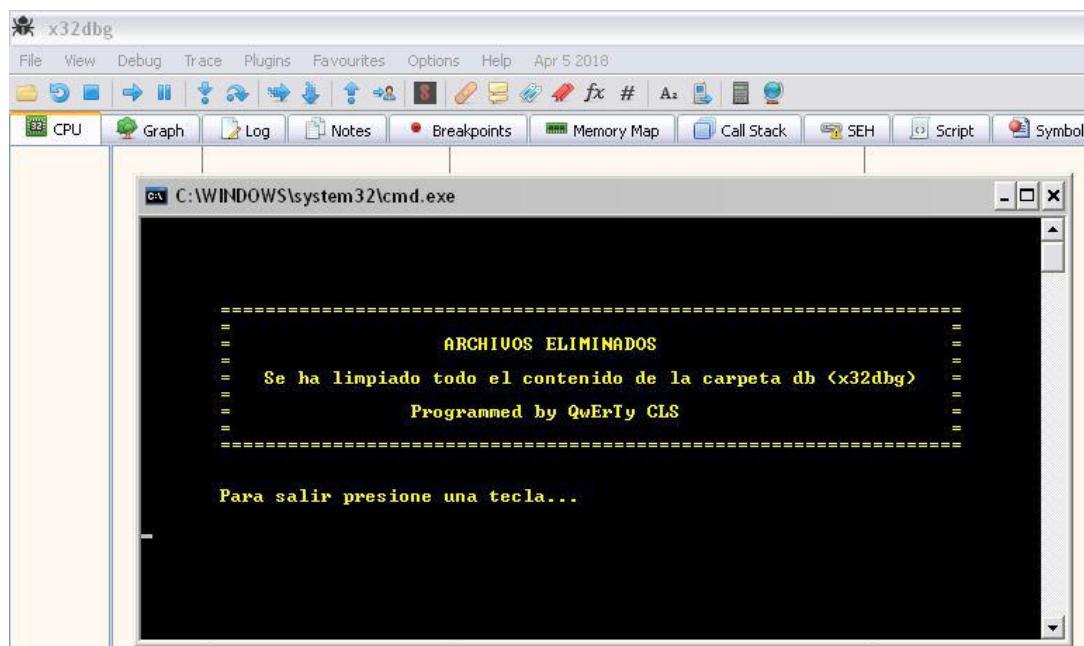
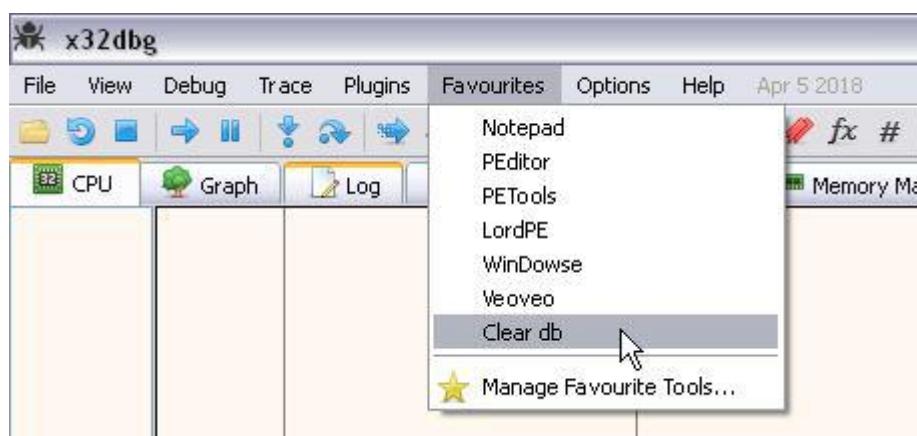
Y lo guardamos.



Ahora se me ocurre otra diablura...je,je,je....

Eliminamos de nuevo el archivo "MiConfiguracion.ini"

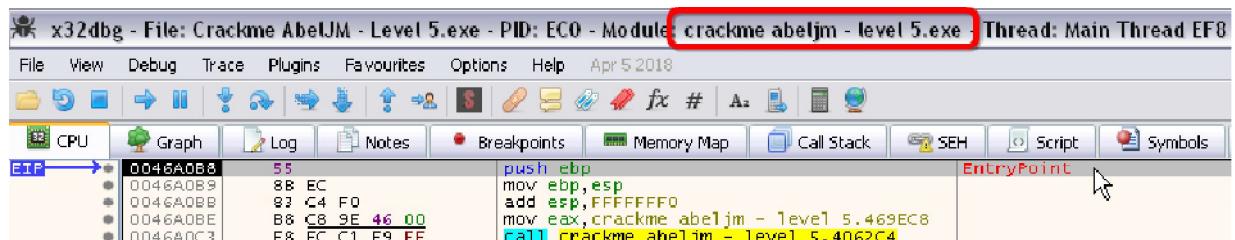
Cerramos el "x32dbg" me voy a favoritos, le doy a mi "Clear db" que yo mismo programé para que me borre todo el contenido que se guarda en la carpeta "db" de nuestro "X32dbg" de forma automática



Cargamos de nuevo con el "x32dbg", el Crackme original



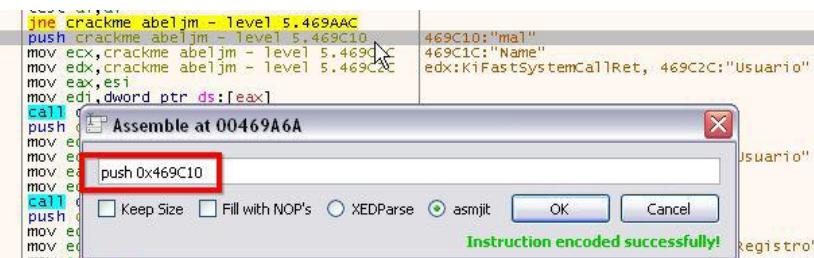
Y aparecemos aquí



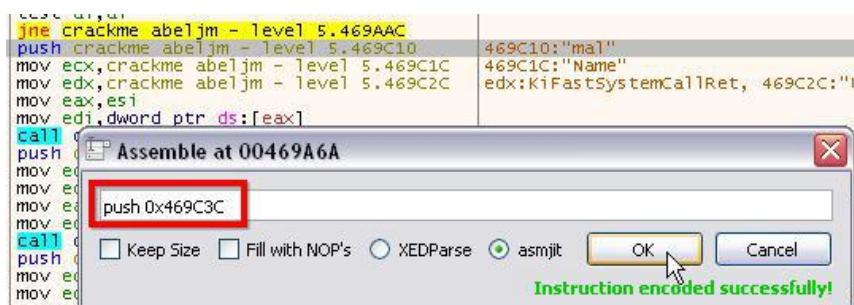
Miramos las "Strings" y las address que las contienen

0046957A	mov edx,crackme abeljm - level 5.4695C8	"jpg"
00469A41	mov edx,crackme abeljm - level 5.469BD8	"MiConfiguracion.ini"
00469A5C	mov eax,crackme abeljm - level 5.469BF4	"ma"
00469A6A	push crackme abeljm - level 5.469C10	"Name"
00469A6F	mov ecx,crackme abeljm - level 5.469C1C	"Usuario"
00469A74	mov edx,crackme abeljm - level 5.469C2C	"cracker"
00469A80	push crackme abeljm - level 5.469C3C	"Password"
00469A85	mov ecx,crackme abeljm - level 5.469C4C	"Usuario"
00469A86	mov edx,crackme abeljm - level 5.469C2C	"Usuario no Registrado"
00469A96	push crackme abeljm - level 5.469C60	"User"
00469A9B	mov ecx,crackme abeljm - level 5.469C80	"Registro"
00469AA0	mov edx,crackme abeljm - level 5.469C90	"desconocido"
00469AAC	push crackme abeljm - level 5.469CA4	"name"
00469AB5	mov ecx,crackme abeljm - level 5.469CB8	"Usuario"
00469ABA	mov edx,crackme abeljm - level 5.469C2C	"desconocido"
00469AC5	push crackme abeljm - level 5.469CA4	"password"
00469ACE	mov ecx,crackme abeljm - level 5.469CC8	"Usuario"
00469AD3	mov edx,crackme abeljm - level 5.469C2C	"desconocido"
00469ADE	push crackme abeljm - level 5.469CA4	"User"
00469AE7	mov ecx,crackme abeljm - level 5.469C80	"Registro"
00469AEC	mov edx,crackme abeljm - level 5.469C90	"ma"
00469B32	mov edx,crackme abeljm - level 5.469C10	"cracker"
00469B4F	mov edx,crackme abeljm - level 5.469C3C	"usuario no registrado"
00469B6C	mov edx,crackme abeljm - level 5.469CDC	"Genial"
00469B7A	mov ecx,crackme abeljm - level 5.469CE4	"Lo registraste cambiando la configuración de i1. Este Crackme simula el about de un pr
00469B7F	mov edx,crackme abeljm - level 5.469CF8	
00469D98	mov eax,crackme abeljm - level 5.469DAC	

Nos vamos a la pantalla principal del debugger y vamos a cambiarlas así:



Por



Segundo

```
11 dword ptr ds:[edi+4]
ish crackme abeljm - level 5.469C3C
iv ecx,crackme abeljm - level 5.469C4C
iv edx,crackme abeljm - level 5.469C2C
iv eax
iv edi
11 dw
ish cr
iv ecx
iv edx
iv eax
iv edi
11 dw
ish cr
ea eax
11 dw
ish cr
ea eax
```

Assemble at 00469A80

push 0x469C3C

Keep Size Fill with NOP's XEDParse asmjit

OK Cancel

Instruction encoded successfully!

Por

```
call dword ptr ds:[edi+4]
push crackme abeljm - level 5.469C3C
mov ecx,crackme abeljm - level 5.469C4C
mov edx,crackme abeljm - level 5.469C2C
mov eax
mov edi
call dw
push cr
mov ecx
mov edx
mov eax
mov edi
call dw
push cr
lea eax
push eax
```

Assemble at 00469A80

push 0x469C2C

Keep Size Fill with NOP's XEDParse asmjit

OK Cancel

Instruction encoded successfully!

Y tercero

```
call dword ptr ds:[edi+4]
push crackme abeljm - level 5.469C60
mov ecx,crackme abeljm - level 5.469C80
mov edx,crackme abeljm - level 5.469C90
mov eax,esi
mov edi
call dw
push cr
lea eax
push e
mov ecx
mov edx
mov eax
mov edi
call dw
push crackme abeljm - level 5.469C94
```

Assemble at 00469A96

push 0x469C60

Keep Size Fill with NOP's XEDParse asmjit

OK Cancel

Instruction encoded successfully!

Por

```
call dword ptr ds:[edi+4]
push crackme abeljm - level 5.469C60
mov ecx,crackme abeljm - level 5.469C80
mov edx,crackme abeljm - level 5.469C90
mov eax,esi
mov edi
call dw
push cr
lea eax
push e
mov ecx
mov edx
mov eax
mov edi
call dw
push crackme abeljm - level 5.469C94
```

Assemble at 00469A96

push 0x469CF4

Keep Size Fill with NOP's XEDParse asmjit

OK Cancel

Instruction encoded successfully!

Y con nuestros cambios de direcciones efectuados, tiene que tener esta pinta

x32dbg - File: Crackme AbelJM - Level 5.exe - PID: ECO - Module: crackme AbelJM - level 5.exe - Thread: Main Thread EF8

File View Debug Trace Plugins Favourites Options Help Apr 5 2018

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source Referer

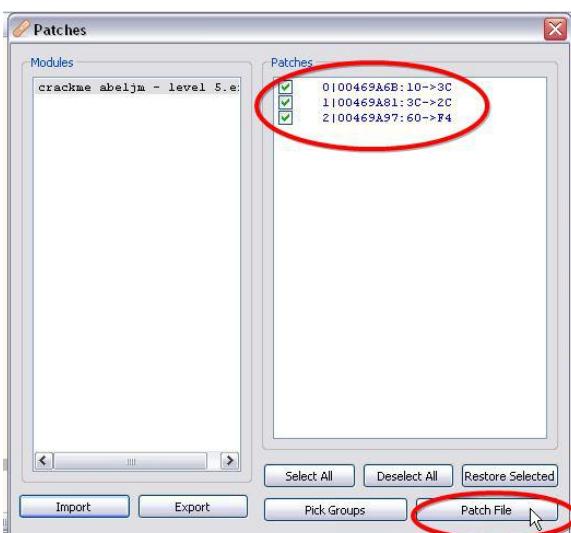
```

00469A3E 8D 45 F0      lea eax,dword ptr ss:[ebp-10]
00469A41 BA D8 9B 46 00 mov edx,crackme AbelJM - level 5.469BD8
00469A44 E8 61 AC F9 FF call crackme AbelJM - level 5.4046AC
00469A4B 8B 40 F0      mov ecx,dword ptr ss:[ebp-10]
00469A4E B2 01          mov d1,i
00469A50 A1 BC 99 42 00 mov eax,dword ptr ds:[4299BC]
00469A53 E8 12 00 FC FF call crackme AbelJM - level 5.429A6C
00469A5A 8B F0          mov esi,ecx
00469A5C B8 F4 9B 46 00 mov eax,crackme AbelJM - level 5.469BF4
00469A61 E8 AA EE F9 FF call crackme AbelJM - level 5.408910
00469A66 84 C0          test al,al
00469A68 75 42          jne crackme AbelJM - level 5.469AAC
00469A6A 68 3C 9C 46 00 push crackme AbelJM - level 5.469C3C
00469A6F B9 1C 9C 46 00 mov ecx,crackme AbelJM - level 5.469C1C
00469A74 BA 2C 9C 46 00 mov edx,crackme AbelJM - level 5.469C2C
00469A79 8B C6          mov eax,esi
00469A7B 8B 38          mov edi,dword ptr ds:[eax]
00469A7D FF 57 04          call dword ptr ds:[edi+4]
00469A80 68 2C 9C 46 00 push crackme AbelJM - level 5.469C2C
00469A85 B9 4C 9C 46 00 mov ecx,crackme AbelJM - level 5.469C1C
00469A88 BA 2C 9C 46 00 mov edx,crackme AbelJM - level 5.469C2C
00469A8F 8B C6          mov eax,esi
00469A91 8B 38          mov edi,dword ptr ds:[eax]
00469A93 FF 57 04          call dword ptr ds:[edi+4]
00469A96 68 F4 9C 46 00 push crackme AbelJM - level 5.469CF4
00469A9B B9 80 9C 46 00 mov ecx,crackme AbelJM - level 5.469C80
00469AA0 BA 90 9C 46 00 mov edx,crackme AbelJM - level 5.469C90
00469AA5 8B C6          mov eax,esi
00469AA7 8B 38          mov edi,dword ptr ds:[eax]

```

edx:KiFastSystemCallRet, 469BD8:"MiConfigurac
469BF4:"Miconfiguracion.ini"
469C3C:'cracker'
469C1C:'Name'
edx:KiFastSystemCallRet, 469C2C:"Usuario"
469C2C: 'Usuario'
469C4C: 'password'
edx:KiFastSystemCallRet, 469C2C:"Usuario"
469CF4: 'Genial'
469C80: 'user'
edx:KiFastSystemCallRet, 469C90:"Registro"

Guardamos el "patch"



Le ponemos otro nombre

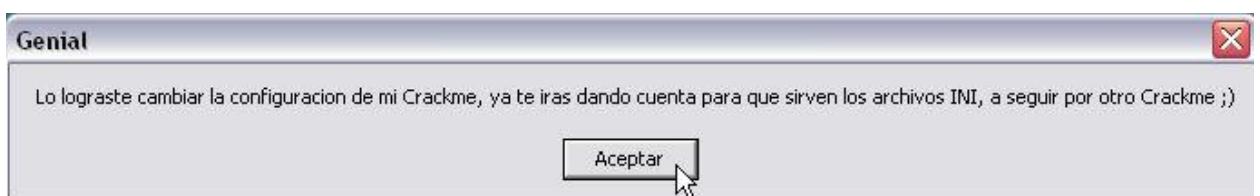


Salimos totalmente del "x32dbg"

Nos vamos a la ruta donde tenemos guardada nuestra nueva creación



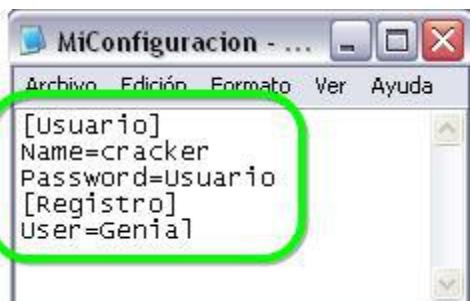
Lo ejecutamos y....



Le damos a aceptar....



Miramos el contenido del archivo "MiConfiguracion.ini" que se ha creado automáticamente y...



nuestro "Otro patched" también funciona correctamente.

+++++ +++++ +++++ +++++ +++++ +++++ +++++ +++++ +++++ +++++

Seguro que hay más y mejores soluciones, pero bueno..., yo me doy por satisfecho con el trabajo que les he mostrado dando por finalizado este entretenido tute de seis Crackmes programados por nuestro compañero **AbelJM**, a quien también aprovecho muy especialmente para saludarlo.

* * * * *

* * * * * * * * * * * THE END * * * * * * * * * * *

.....MISIÓN CUMPLIDA.....



Mis agradecimientos infinitos a

CracksLatinoS

EL QUE APRENDE Y APRENDE Y NO PRACTICA LO QUE SABE,
ES COMO EL QUE ARA Y ARA Y NO SIEMBRA.

Platón

Salu2

<< QwErTy CLS >>

20 de Mayo de 2018