RegexBuddy 3 – by **+xcdfgt**

Programa	**RegexBuddy v 3.6.1**
Download	http://descargar.cnet.com/RegexBuddy/3000-2352_4-10309844.html
Descripción	Permite probar expresiones regulares
Herramientas	Olly 1.10, PeiD, RDG Packer Detector v0.7.3
Dificultad	Newbie
Compresor/Compilador	Borland Delphi 6.0 – 7.0
Protección	No se encuentra empacado o protegido
Objetivos	Quitar nag de inicio y bloqueo en texbox
Cracker	**xcdfgt** Fecha: 26/05/2015
Tutorial nº	2

Introducción

Lo que se busca con este proceso de cracking es:

1. Quitar Nag de inicio
2. Suprimir Bloqueo de textbox, que impide probar nuestras expresiones regulares.

Comencemos

Como siempre, antes de comenzar a trabajar, le pasamos el PeiD a la víctima para ver qué es lo que nos trae:

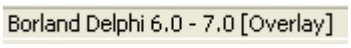
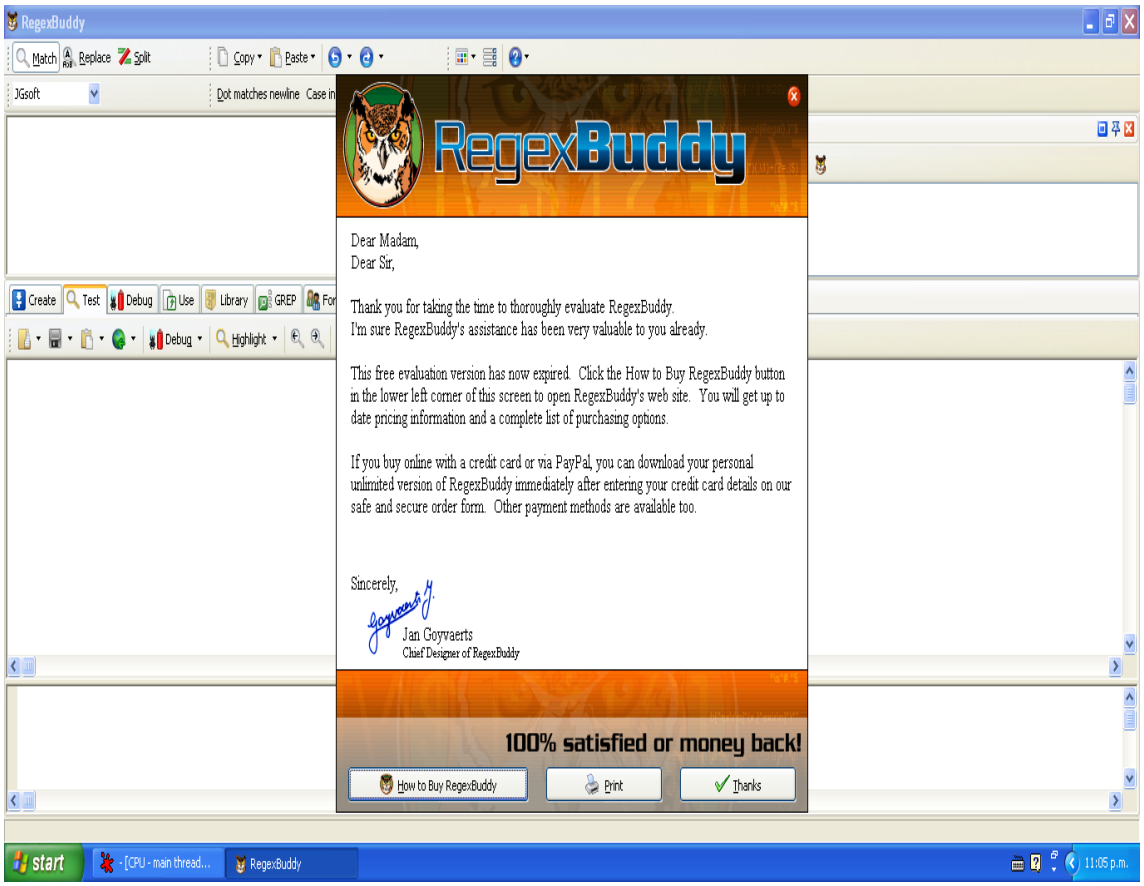


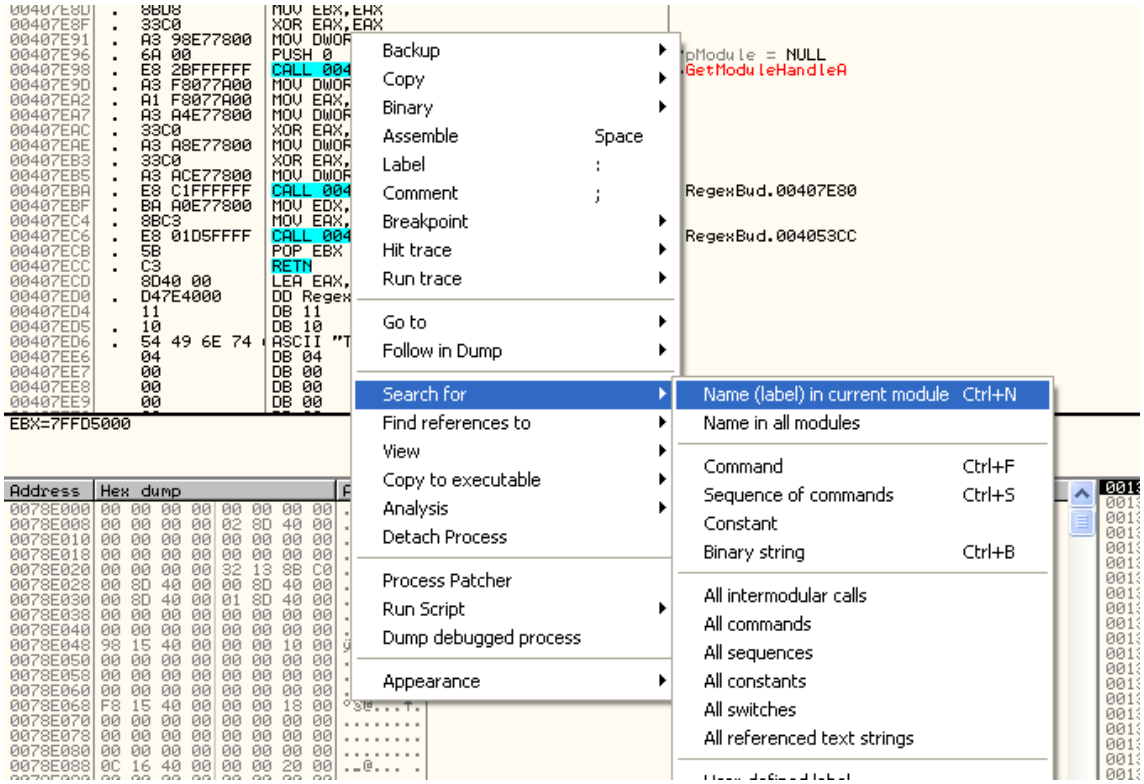
Imagen de NAG

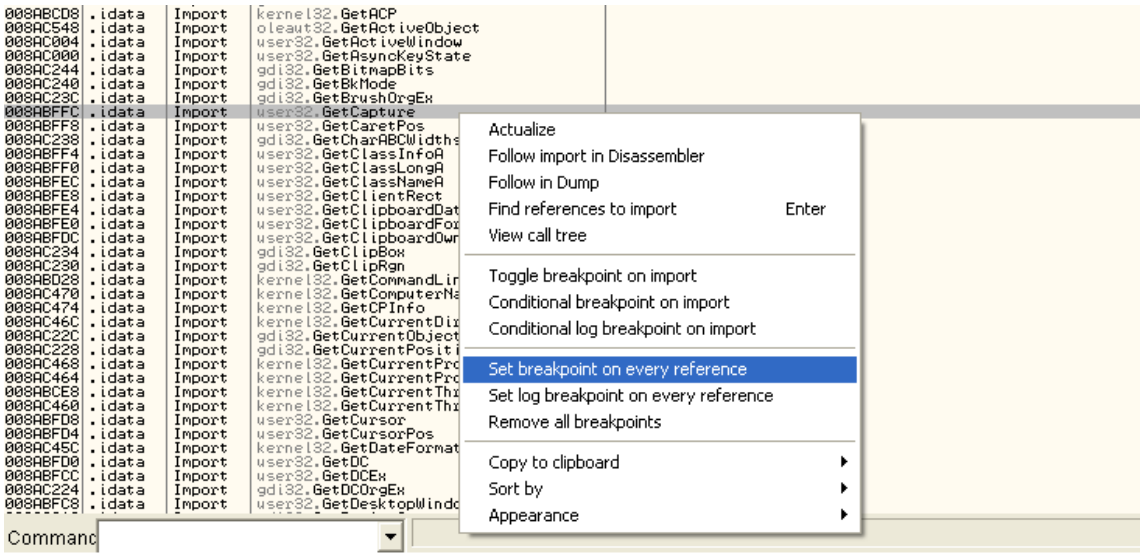


Para suprimir esta pantalla inicial, debemos buscar un call , el cual es responsable de esta fastidiosa ventana.
A claro que este comportamiento se replica en programas compilados en Borlan Delphy Tal como se menciona en el **manual COMO ELIMINAR NAGS EN PROGRAMAS DELPHI.doc**

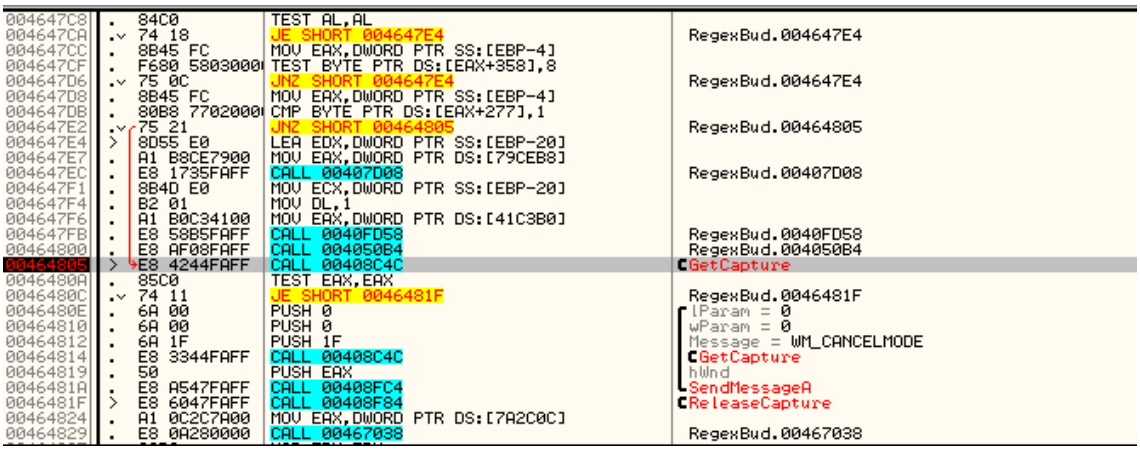
Lo primero que tenemos que hacer es colocar un breakpoint en todas al referencias al Api GetCapture , lo cual hacemos de la siguiente manera .

Obviamente esto lo tenemos que hacer parados en el **Entry ponit** , para este caso el olly nos deja , el solo en este punto, solo es abrir el ejecutable y listo.
Seguido esto, realizamos lo siguiente.



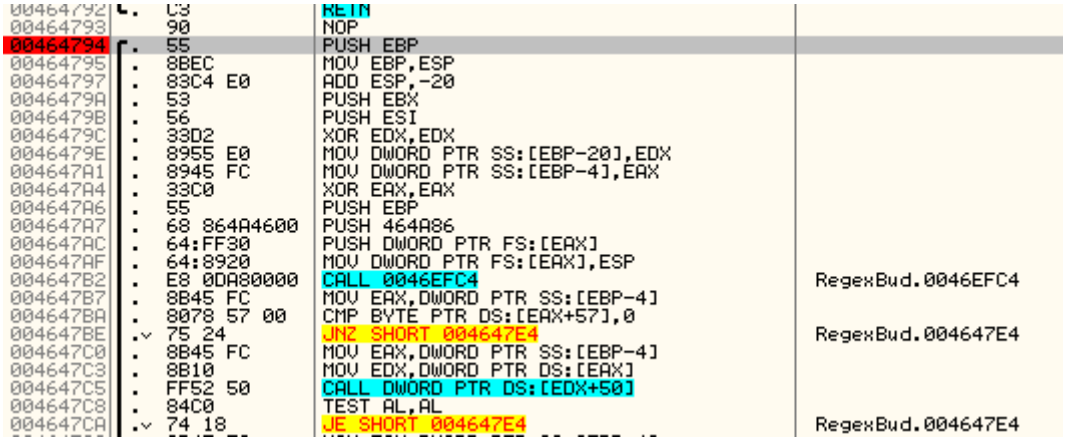


Al hacer esto, nuestro programa se va a detener en todas las llamadas al Api GetCapture.
Puede haber muchas paradas a esta Api, pero lo que tenemos que hacer es buscar un llamada la cual tenga el siguiente aspecto:

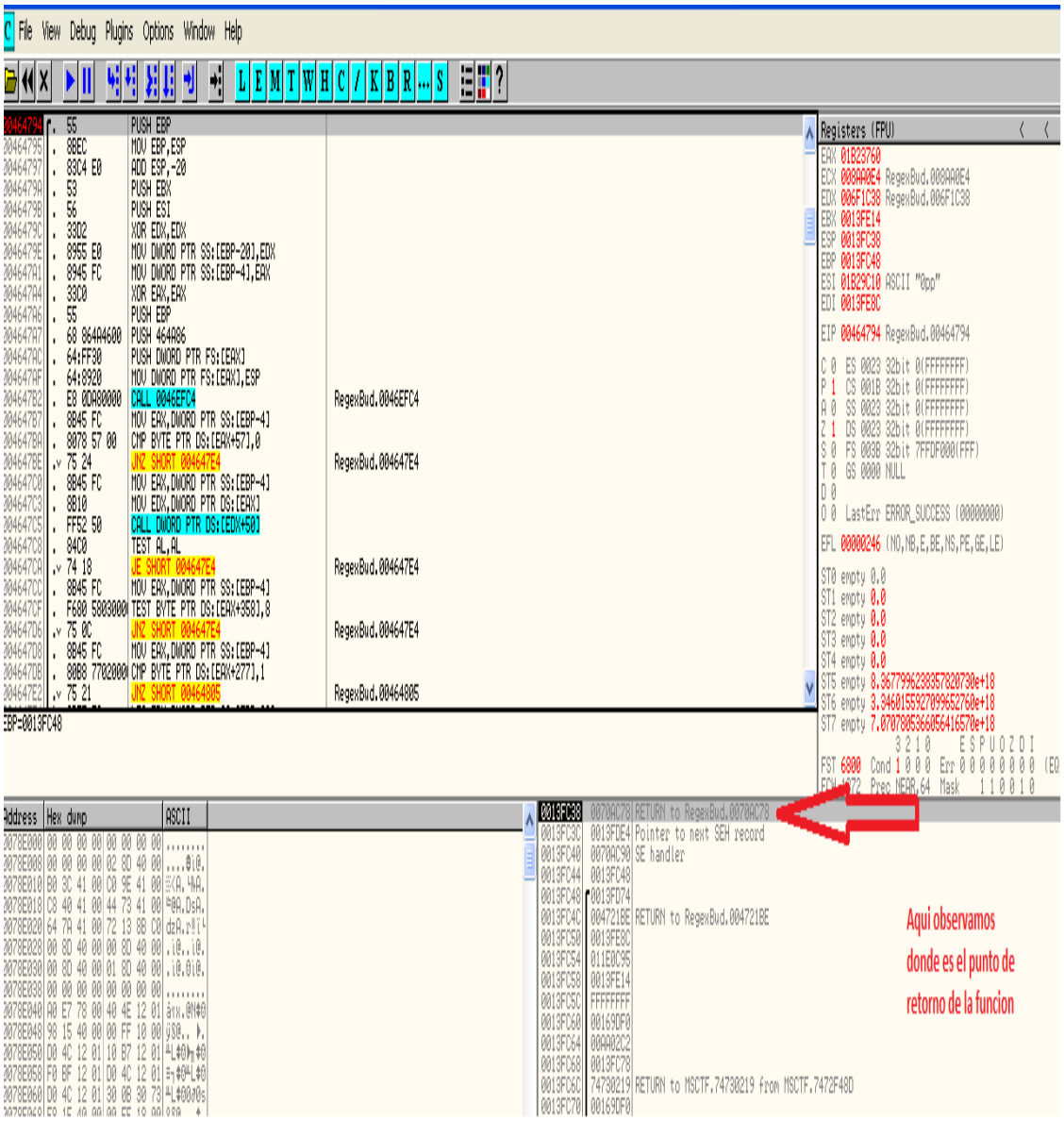


Lo que tenemos que tener en cuenta, es que nuestra llamada al Api **GetCapture** , tenga abajo llamadas a las Apis **SenMessageA** y **ReleaseCapture ,** tal como se muestra en la imagen.

Después de localizar esta llamada, debemos subir hasta encontrar el punto de inicio donde comienza nuestra llamada, es decir un PUSH EBP ,el cual deber estar un poco más arriba de la llamada al Api GetCapture.



Llegado a este punto, colocamos un breakpoint, damos reiniciar y esta vez debería detenerse en el último breakpoint colocado .
Antes de volverlo a correr quitamos todos los breakpoints colocados anteriormente, o los deshabilitamos y damos RUN.



Al llegar al breakpoint, observamos la pila , y en ella podemos visualizar una dirección de retorno

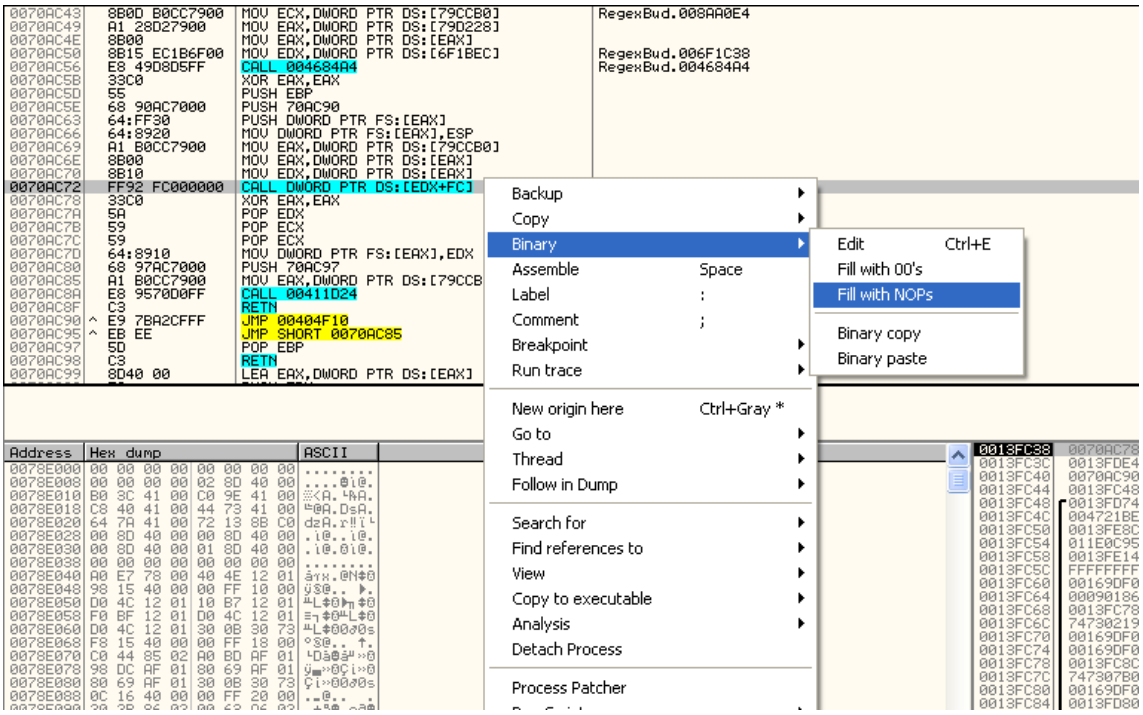
RETURN to RegexBud 0072AC78.

Vamos a esta dirección

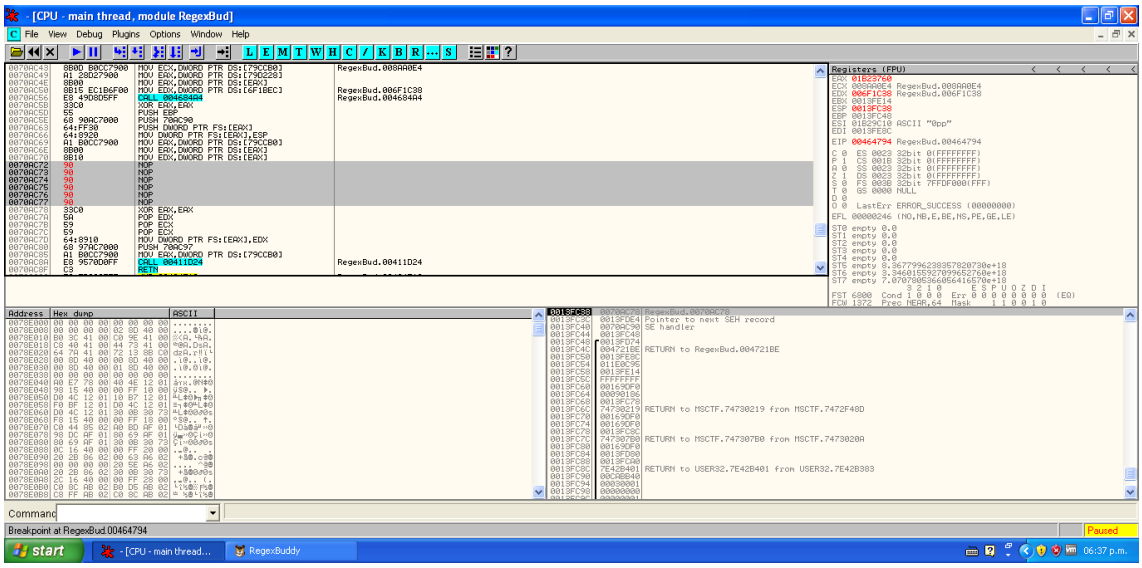
```
0070AC43    8B0D B0CC7900   MOV ECX,DWORD PTR DS:[79CCB0]      RegexBud.008AA0E4
0070AC49    A1 28D27900     MOV EAX,DWORD PTR DS:[79D228]
0070AC4E    8B00            MOV EAX,DWORD PTR DS:[EAX]
0070AC50    8B15 EC1B6F00   MOV EDX,DWORD PTR DS:[6F1BEC]      RegexBud.006F1C38
0070AC56    E8 49D8D5FF     CALL 004684A4                      RegexBud.004684A4
0070AC5B    33C0            XOR EAX,EAX
0070AC5D    55              PUSH EBP
0070AC5E    68 90AC7000     PUSH 70AC90
0070AC63    64:FF30         PUSH DWORD PTR FS:[EAX]
0070AC66    64:8920         MOV DWORD PTR FS:[EAX],ESP
0070AC69    A1 B0CC7900     MOV EAX,DWORD PTR DS:[79CCB0]
0070AC6E    8B00            MOV EAX,DWORD PTR DS:[EAX]
0070AC70    8B10            MOV EDX,DWORD PTR DS:[EAX]
0070AC72    FF92 FC000000   CALL DWORD PTR DS:[EDX+FC]
0070AC78    33C0            XOR EAX,EAX
0070AC7A    5A              POP EDX
0070AC7B    59              POP ECX
0070AC7C    59              POP ECX
0070AC7D    64:8910         MOV DWORD PTR FS:[EAX],EDX
0070AC80    68 97AC7000     PUSH 70AC97
0070AC85    A1 B0CC7900     MOV EAX,DWORD PTR DS:[79CCB0]
0070AC8A    E8 9570D0FF     CALL 00411D24                      RegexBud.00411D24
0070AC8F    C3              RETN
0070AC90  ^ E9 7BA2CFFF     JMP 00404F10                       RegexBud.00404F10
0070AC95  ^ EB EE           JMP SHORT 0070AC85                 RegexBud.0070AC85
0070AC97    5D              POP EBP
0070AC98    C3              RETN
0070AC99    8D40 00         LEA EAX,DWORD PTR DS:[EAX]
```

Arriba de la posición mencionada, podemos ver el siguiente **CALL DWORD PTR DS:[EDX+FC]** , el cual es el responsable de la llama a nuestra API GetCapture.

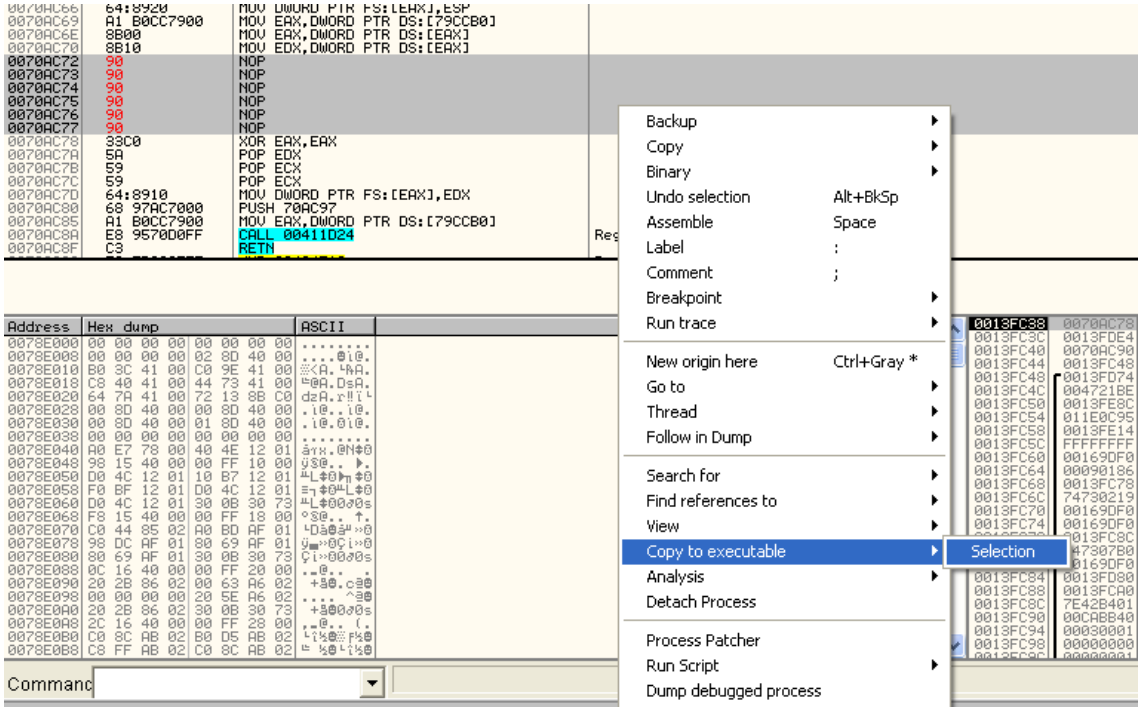
A continuación de esto, vamos a nopear la llamada a este call.



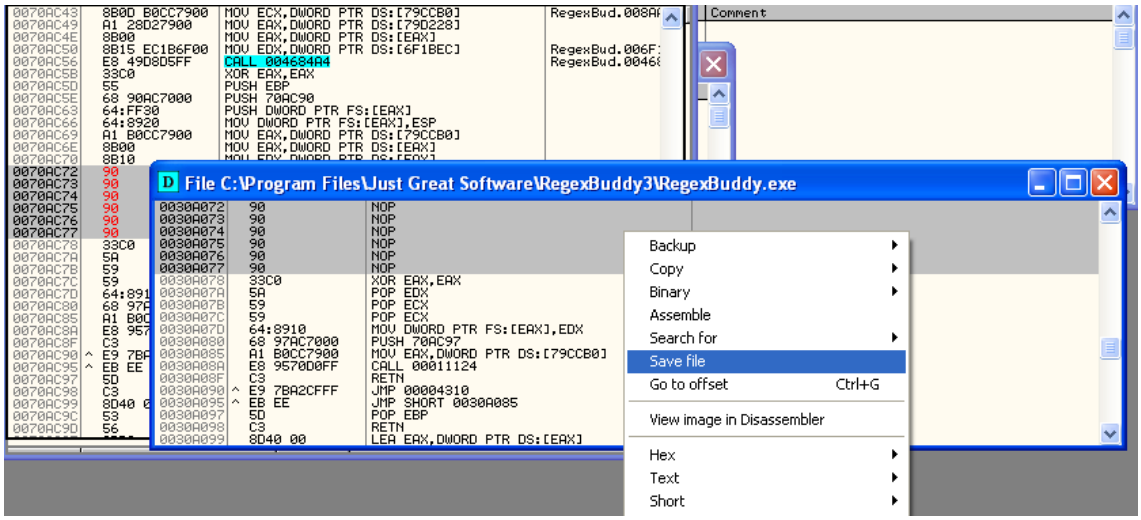
Después del procedimiento quedaría así



Guardamos los cambios realizados

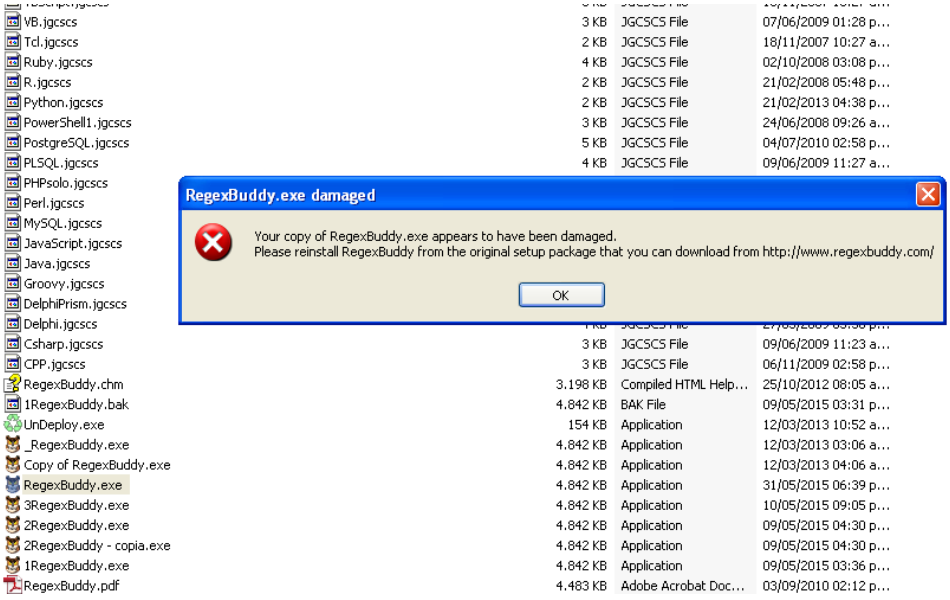


Seguido esto



Y con esto guardamos los cambios.

Después intentamos abrir nuevamente el ejecutable, pero esta vez desde el explorador de Windows, lo buscamos y damos doble click, pero para o sorpresa nos sale el siguiente mensaje



Confieso que al principio pensé que este mensaje, lo único que me está notificando era que el ejecutable se había descompuesto , por realizar los cambios ya mencionados , pero luego de mirar con detenimiento , pude evidenciar , que esta mensaje no provenía del sistema operativo , sino del propio Regexbuddy , el cual de alguna forma se estaba dando cuenta de que había manipulado el ejecutable ,y activaba una rutina de validación que me sacaba de la ejecución normal y me impedía correr el ejecutable.

Por lo tanto después de esta ejecución, me di a la tarea de hallar esta rutina, y así poder invalidarla y poder continuar.

Mencionado lo anterior, nos dimos a la tarea, para ello volvemos a reiniciar el ejecutable y vamos haciendo un paso a paso, hasta llegar al punto

Con la tecla F8 vamos dándole hasta llegar al siguiente punto

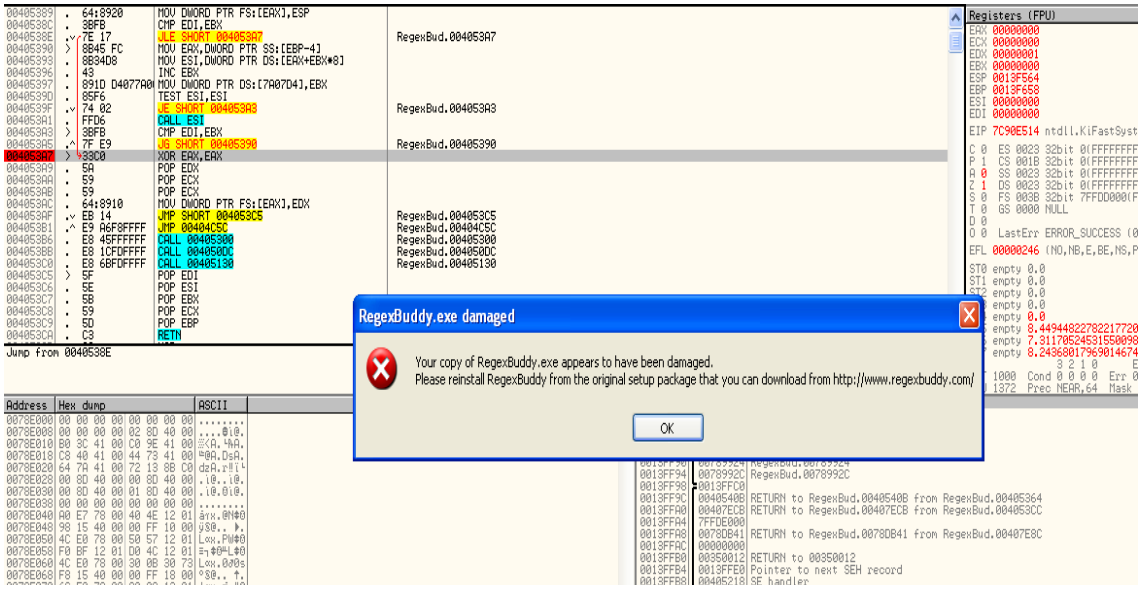
```
00405389  .   64:8920        MOV DWORD PTR FS:[EAX],ESP
0040538C  .   3BFB           CMP EDI,EBX
0040538E  .v  7E 17          JLE SHORT 004053A7                  RegexBud.004053A7
00405390  >   8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
00405393  .   8B34D8         MOV ESI,DWORD PTR DS:[EAX+EBX*8]
00405396  .   43             INC EBX
00405397  .   891D D4077A0(  MOV DWORD PTR DS:[7A07D4],EBX
0040539D  .   85F6           TEST ESI,ESI
0040539F  .v  74 02          JE SHORT 004053A3                   RegexBud.004053A3
004053A1  .   FFD6           CALL ESI
004053A3  >   3BFB           CMP EDI,EBX
004053A5  .^  7F E9          JG SHORT 00405390                   RegexBud.00405390
004053A7  >   33C0           XOR EAX,EAX
004053A9  .   5A             POP EDX
004053AA  .   59             POP ECX
004053AB  .   59             POP ECX
004053AC  .   64:8910        MOV DWORD PTR FS:[EAX],EDX
```

Llegando aquí, colocamos un breakpoint en el punto
004053A7 > \33C0 XOR EAX,EAX

Así como se muestra en la siguiente imagen

```
00405389  .   64:8920        MOV DWORD PTR FS:[EAX],ESP
0040538C  .   3BFB           CMP EDI,EBX
0040538E  .v  7E 17          JLE SHORT 004053A7                  RegexBud.004053A7
00405390  >   8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
00405393  .   8B34D8         MOV ESI,DWORD PTR DS:[EAX+EBX*8]
00405396  .   43             INC EBX
00405397  .   891D D4077A0(  MOV DWORD PTR DS:[7A07D4],EBX
0040539D  .   85F6           TEST ESI,ESI
0040539F  .v  74 02          JE SHORT 004053A3                   RegexBud.004053A3
004053A1  .   FFD6           CALL ESI
004053A3  >   3BFB           CMP EDI,EBX
004053A5  .^  7F E9          JG SHORT 00405390                   RegexBud.00405390
004053A7  >   33C0           XOR EAX,EAX
```

Después de poner le breakpoint damos F9, a ver qué pasa, después de esto veo que nos sale el mensaje de validación



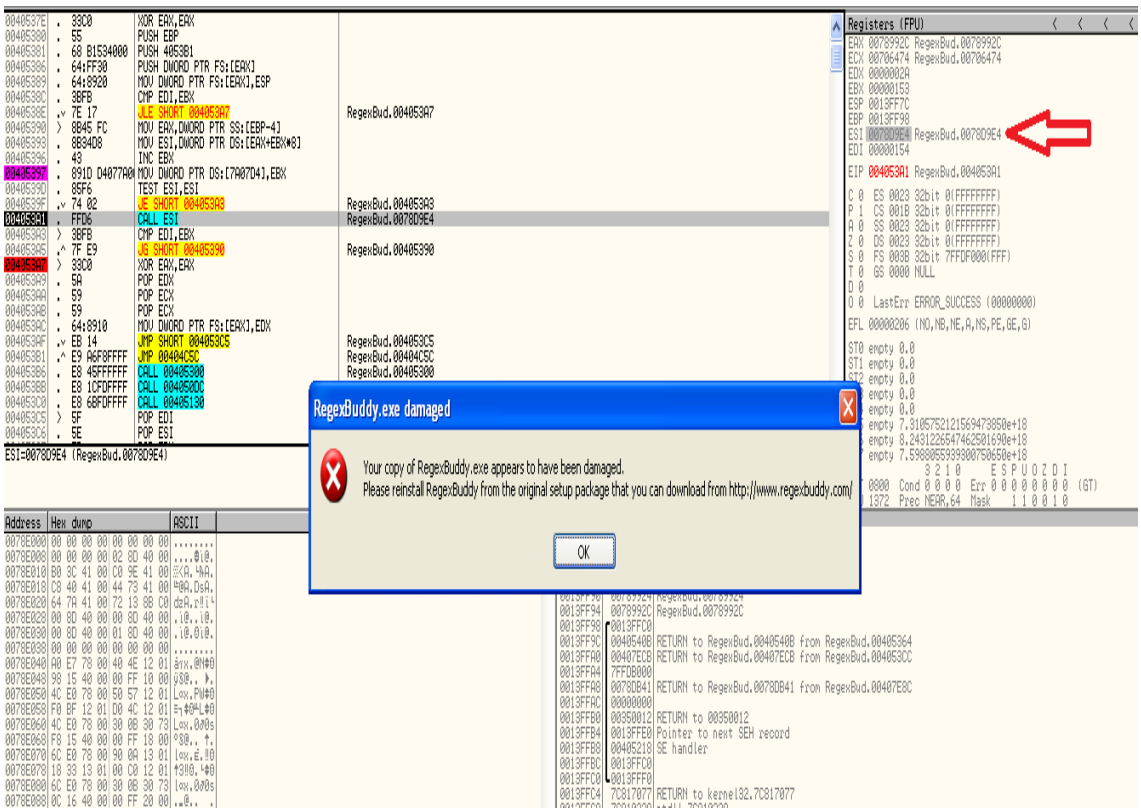
Con esto, nos podemos dar cuenta, de que nuestro mensaje de validación se está ejecutando en algún momento del ciclo, el cual podemos ver arriba, lo importante a conocer aquí, es cuando se da esta condición, y posteriormente anularla.

Como sabes este cartel se ejecuta en algún momento del ciclo, a continuación yo voy a comentar como llegue al punto, sin embargo también es posible hacerlo atreves de múltiples métodos, lo que hice fue colocar un breakpoint condicional cuando lo registro **ECX** del procesador tuviera el valor de **0013FF74,** es claro que esto valores puedan cambiar dependiendo de la maquina donde estén,

Esto se hace con el fin de no tener que llegar al punto de validación, usando la tecla F8 indefinidamente, esto breakpoint lo que hizo fue llevarme mas cercar del punto donde se ejecuta ,

Otra cosa que hice, fue que analizando el código del ciclo veo por lógica, que la llamada **CALL ESI** es la que está ejecutando la validación, el desafío es saber cuándo lo hace.
Después de llegar al breakpoint condicional, y dar unos cuantos F8 vemos que antes de ejecutar el cartel que nos saca de la ejecución normal.
El registro **ESI** vale **0078D9E4,** ósea que esta es punto es inmediatamente anterior a la ejecución de la validación.

Seguido esto colocamos un breakponit condicional que nos detenga cuando el registro **ESI** valga **0078D9E4.**
Colocado el breakpoint, damos reiniciar y esperemos que pare en el punto mencionado, quitamos todos los breakpoints y solo dejamos este.



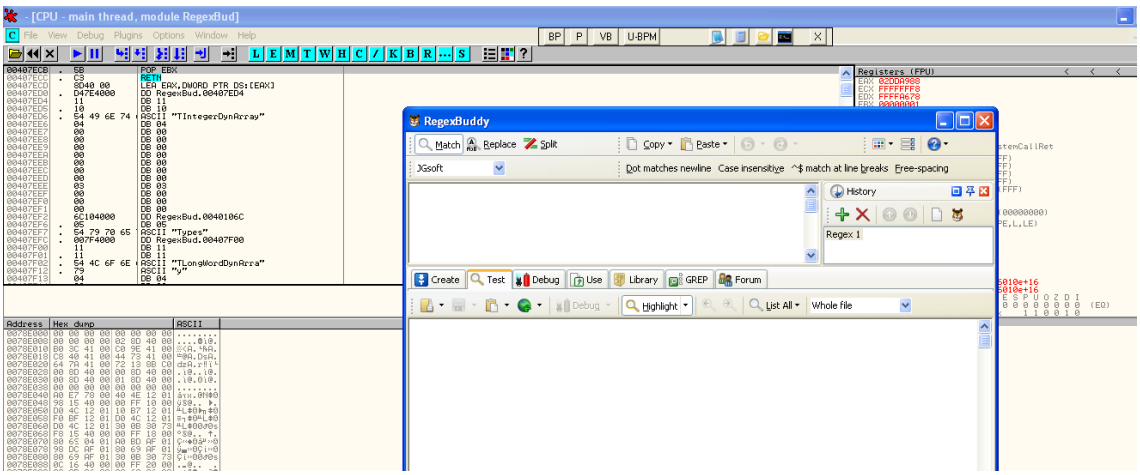
Después de volver a ejecutarlo, y parar en el breakpoint, cuando **ESI** valga 0078D9E4 vemos que el CALL ESI ejecuta la validación que nos saca de la ejecución normal del programa, con esto podemos concluir que dentro del CALL ESI si esta el punto que debemos anular para que el error se genere.

Volvemos a reiniciar, paramos en el punto, pero esta vez vamos a ingresar al call. Ingresando al CALL vemos la siguiente imagen.

```
0078D9E7   6A 00          PUSH 0
0078D9E9   6A 00          PUSH 0
0078D9EB   6A 00          PUSH 0
0078D9ED   33C0           XOR EAX,EAX
0078D9EF   55             PUSH EBP
0078D9F0   68 74DA7800    PUSH 78DA74
0078D9F5   64:FF30        PUSH DWORD PTR FS:[EAX]
0078D9F8   64:8920        MOV DWORD PTR FS:[EAX],ESP
0078D9FB   832D 20A18A00  SUB DWORD PTR DS:[8AA120],1
0078DA02   73 55          JNB SHORT 0078DA59          RegexBud.0078DA59
0078DA04   E8 A35CF8FF    CALL 007136AC               RegexBud.007136AC
0078DA09   84C0           TEST AL,AL
0078DA0B   75 3A          JNZ SHORT 0078DA47          RegexBud.0078DA47
0078DA0D   6A 10          PUSH 10
0078DA0F   8D55 FC        LEA EDX,DWORD PTR SS:[EBP-4]
0078DA12   B8 04387100    MOV EAX,713804
0078DA17   E8 ECA2C7FF    CALL 00407D08               RegexBud.00407D08
0078DA1C   8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
0078DA1F   E8 D881C7FF    CALL 00405BFC               RegexBud.00405BFC
0078DA24   50             PUSH EAX
0078DA25   8D55 F8        LEA EDX,DWORD PTR SS:[EBP-8]
0078DA28   B8 FC377100    MOV EAX,7137FC
0078DA2D   E8 D6A2C7FF    CALL 00407D08               RegexBud.00407D08
0078DA32   8B45 F8        MOV EAX,DWORD PTR SS:[EBP-8]
0078DA35   E8 C281C7FF    CALL 00405BFC               RegexBud.00405BFC
0078DA3A   50             PUSH EAX
0078DA3B   6A 00          PUSH 0
0078DA3D   E8 BAB4C7FF    CALL 00408EFC               <JMP.&user32.MessageBoxA>
```

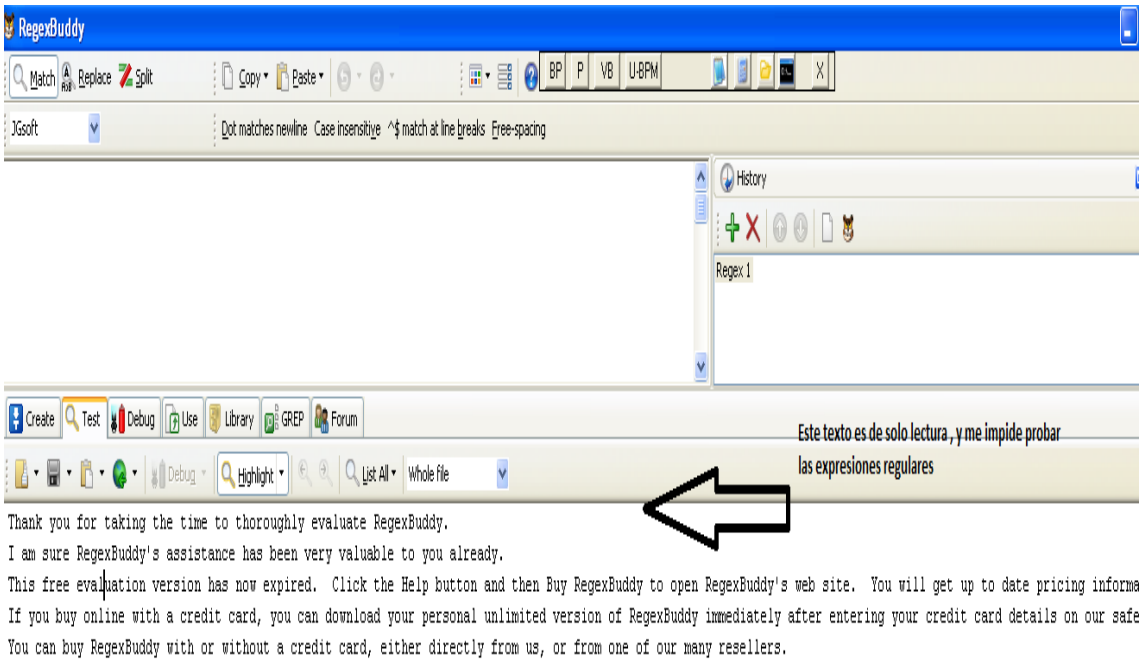
En la imagen vemos que hay un posible salto que podría ser el causante de nuestro error, probamos convirtiéndolo en un JMP a ver qué, pasa y damos F9.

WALA, nuestro programa carga sin problemas, tal como se muestra en la imagen



Guardamos los cambios y listo. Realizando esto, ya logramos superar el primer objetivo que era quitar la NAG de inicio.

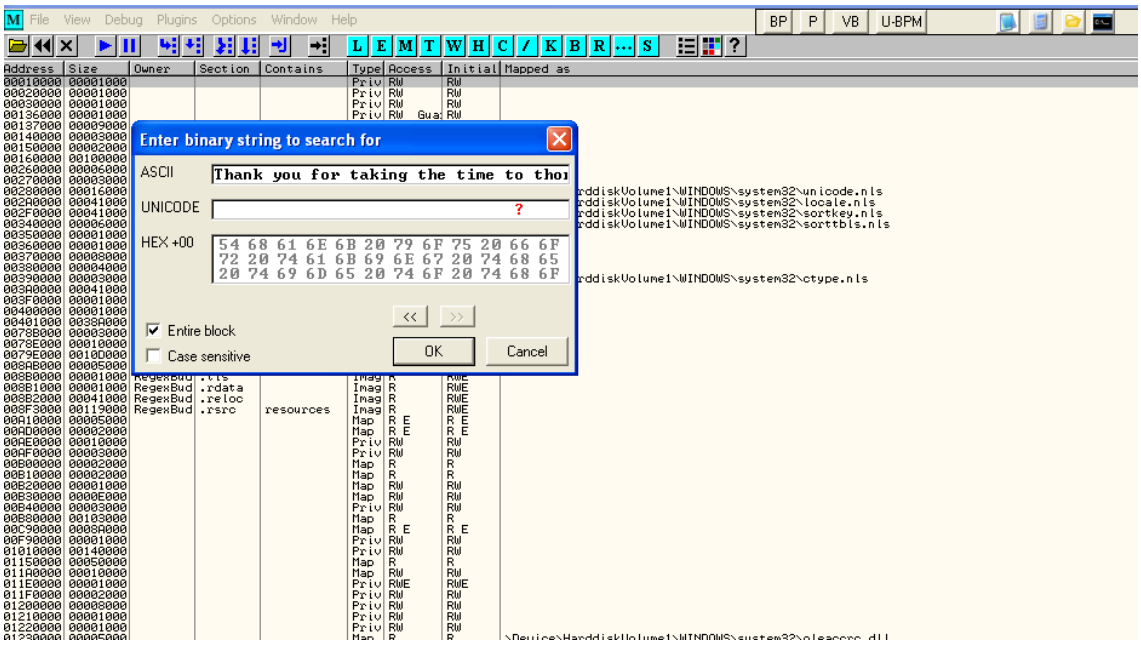
Pero aún nos falta quitar la restricción que tiene el textbox, para probar las expresiones regulares, tal como se muestra en la imagen abajo



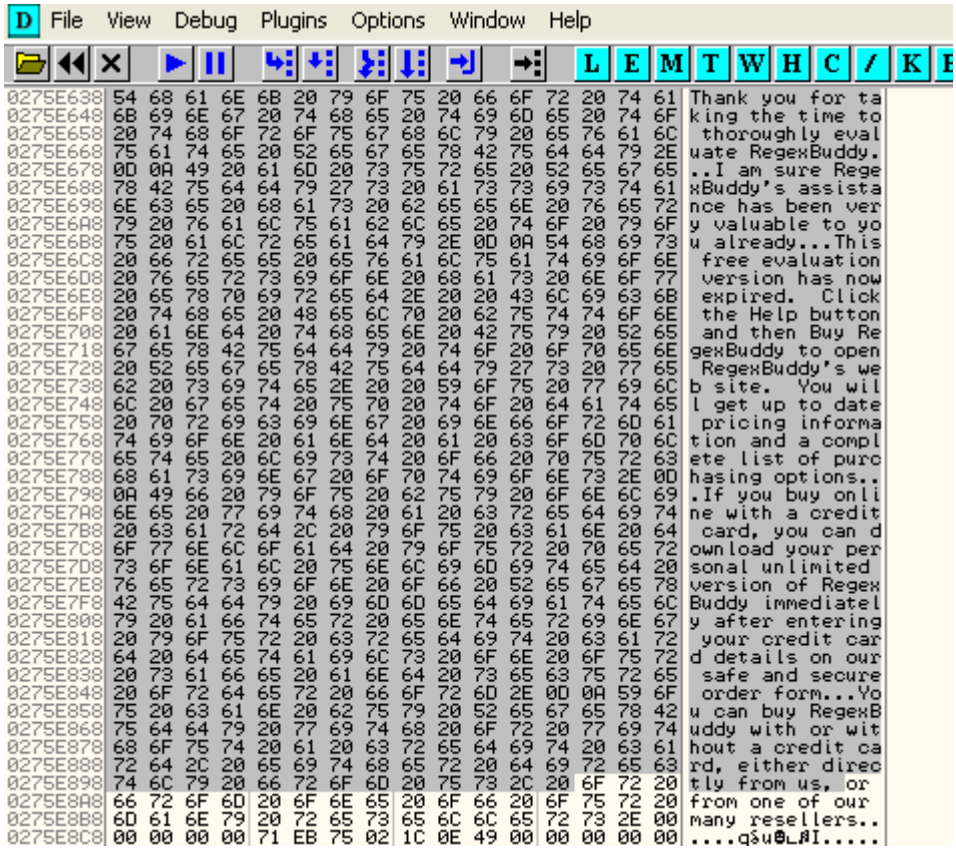
Revisando el comportamiento, analizamos que este texto se pone, a penas nosotros damos click inmediatamente, para solucionar esto primero voy a buscar en memoria donde se está copiando este texto, damos un search en la memoria.

Damos click en la opción M del toolbox de arriba, y copiamos el texto a buscar el cual es el siguiente:

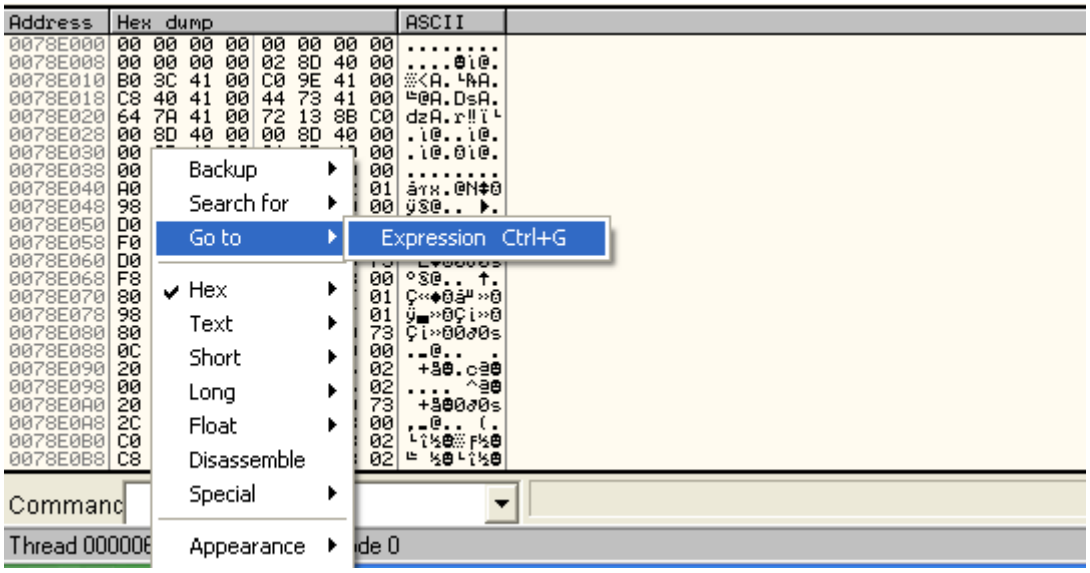
Thank you for taking the time to thoroughly evaluate RegexBuddy.
I am sure RegexBuddy's assistance has been very valuable to you already.
This free evaluation version has now expired. Click the Help button and then Buy RegexBuddy to open RegexBuddy's web site. You will get up to date pricing information and a complete list of purchasing options.
If you buy online with a credit card, you can download your personal unlimited version of RegexBuddy immediately after entering your credit card details on our safe and secure order form.
You can buy RegexBuddy with or without a credit card, either directly from us, or from one of our many resellers.

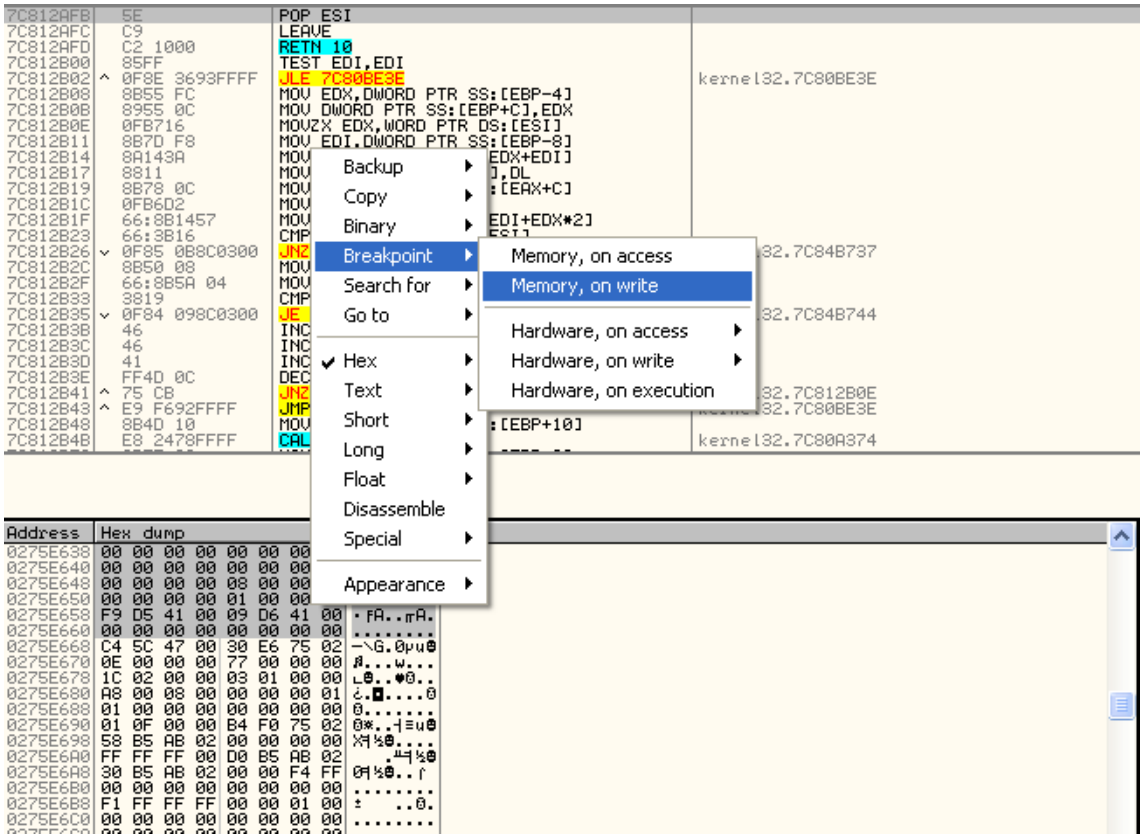


Damos ok, y nos lleva a la siguiente zona.

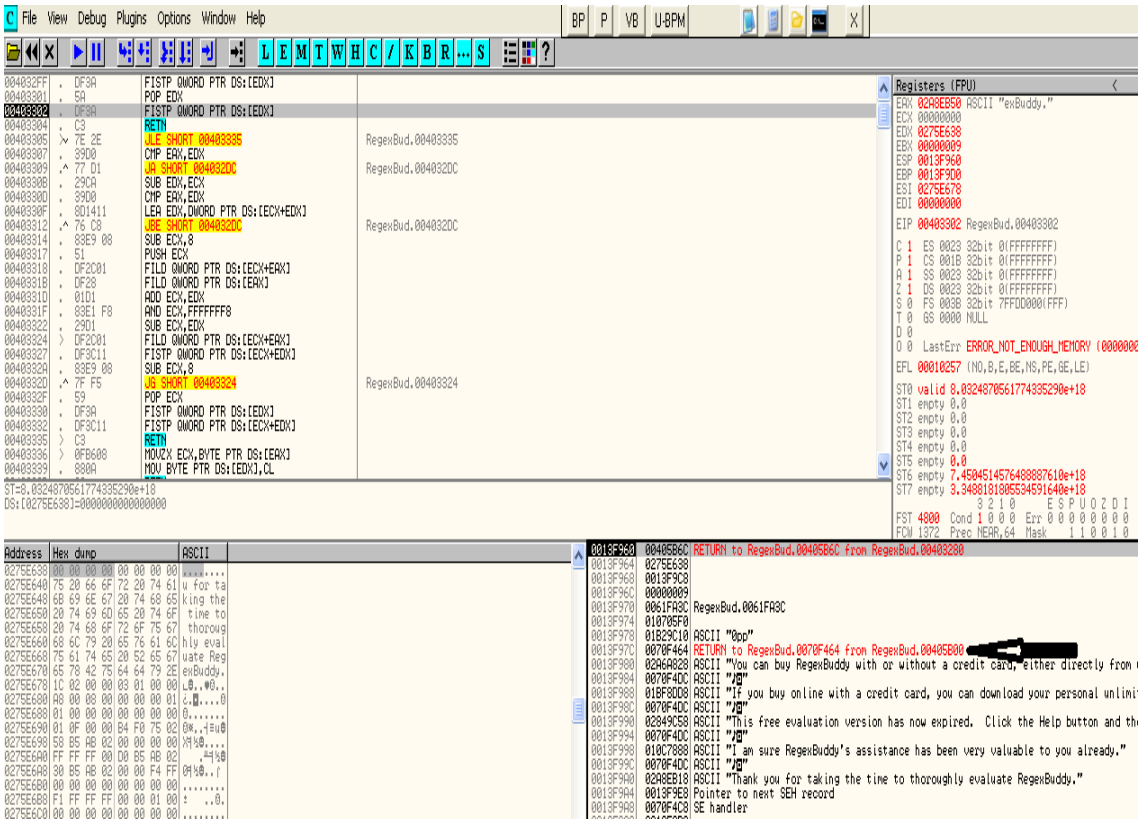


Con esto podemos concluir, que la dirección a partir de la cual se escribe el texto es **0275E638**, identificada la zona damos reiniciar, y después colocamos un breakpoint en la dirección **0275E638**





Al dar reiniciar, el olly parara en esta zona, y al volver a la pantalla y dar click en el textbox para en esta zona.

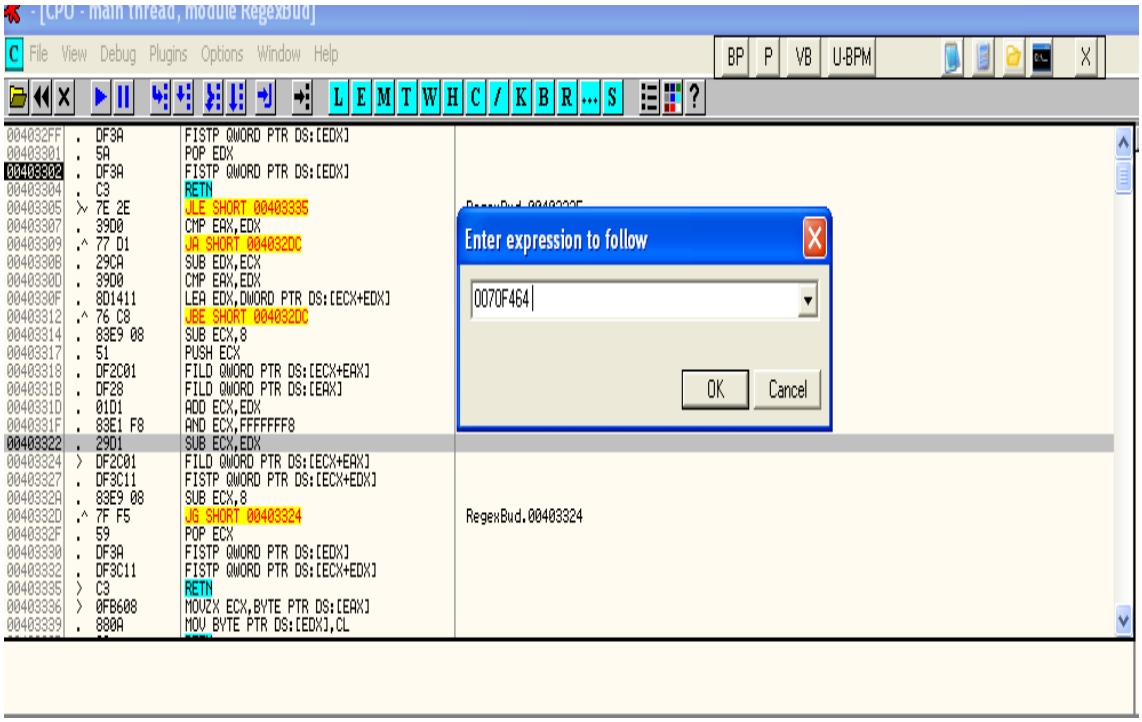


Esta zona, es el área caliente y observando la pila vemos unos Return, los cuales nos pueden llevar a la zona definitiva,

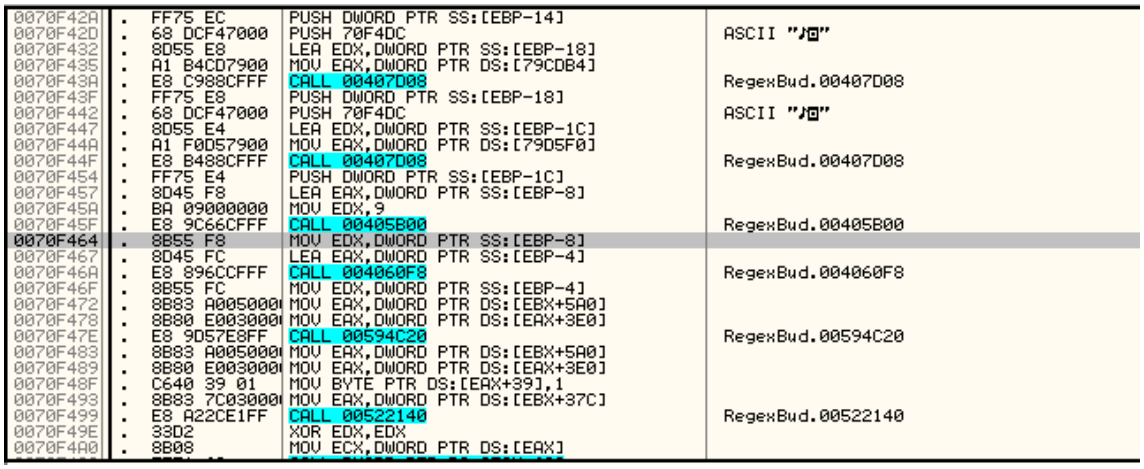
Yo probé los dos que vemos en memoria, pero el que nos interesa es el que vemos arriba seleccionado con la flecha negra.

Vamos a esta zona, y colocamos un breakpoint en la dirección de retorno especificada.

0013F97C 0070F464 RETURN to RegexBud.0070F464 from RegexBud.00405B00

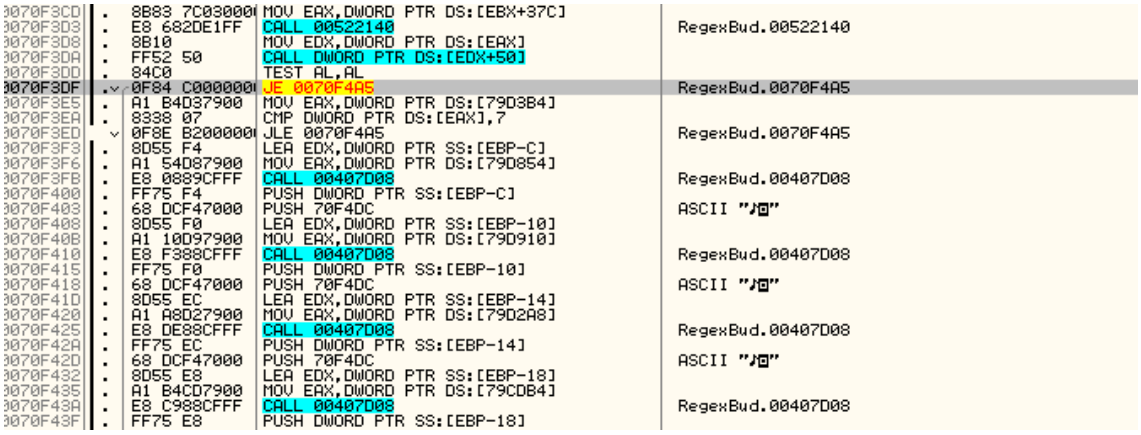


La cual nos lleva a la siguiente zona



El CALL que vemos aquí, es el que se llama antes de que active la protección.
Buscamos arriba si existe algún salto, el cual nos evite llegar a esta zona.

Haciendo nuestra búsqueda llegamos al área en cuestión.



Cambiamos el Salto JE por un salto JMP, guardamos los cambios y listo. Realizado la modificación nuestro Software queda totalmente operativo y usable.

Muchas gracias.