



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 textbox
Cracker	xcdfgt Fecha: 26/05/2015
Tutorial nº	2

Introducción

Lo que se busca con este proceso de cracking es:

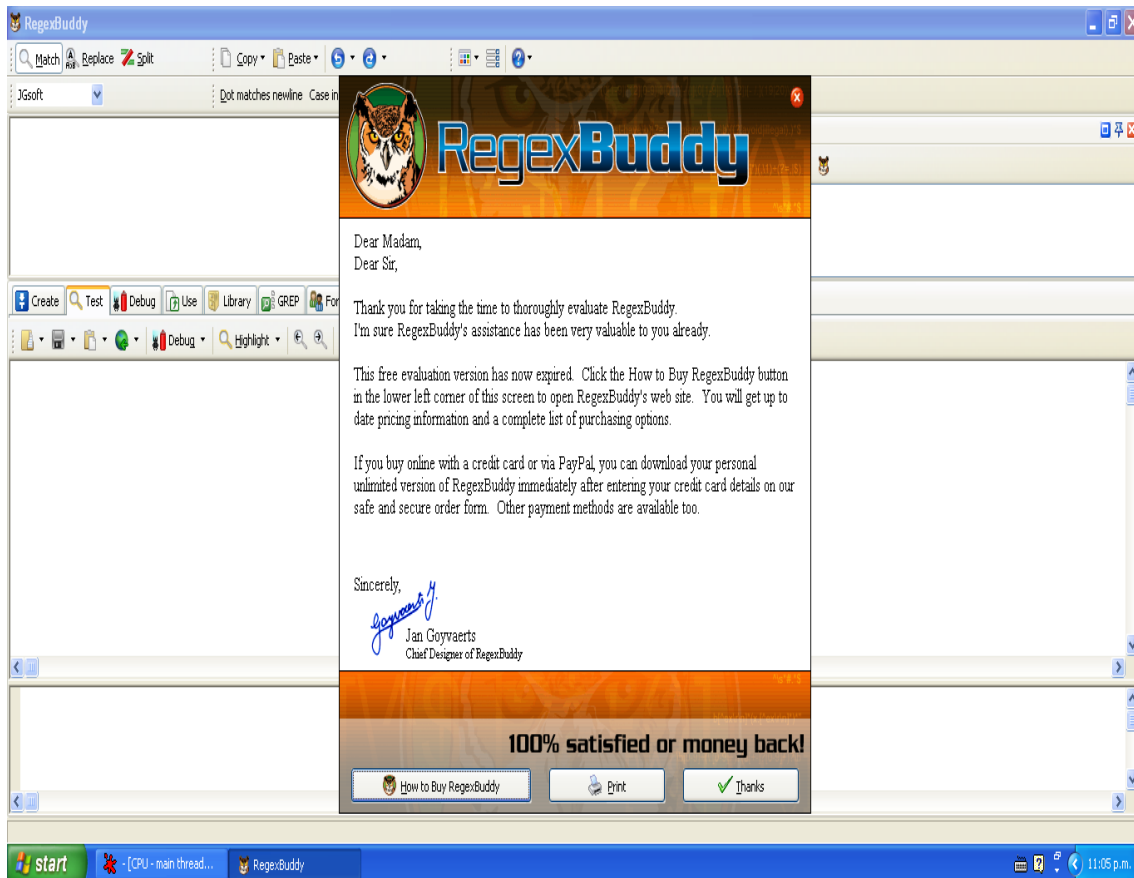
1. Quitar Nag de inicio
2. Suprimir Bloqueo de textbox, que impide probar nuestras expresiones regulares.

Comencemos

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

Borland Delphi 6.0 - 7.0 [Overlay]

Imagen de NAG



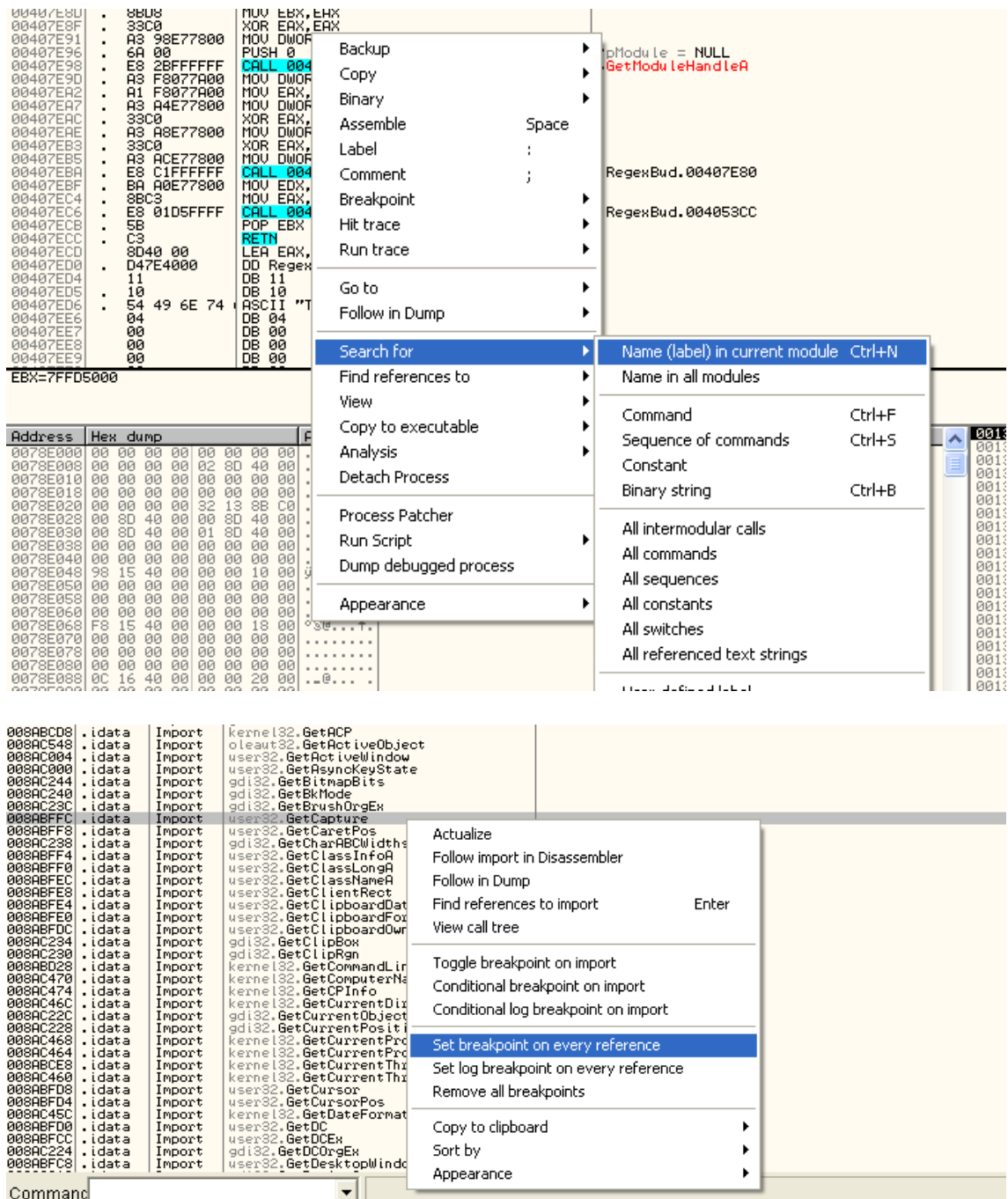
Para suprimir esta pantalla inicial, debemos buscar un call , el cual es responsable de esta fastidiosa ventana.

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.

RegexBuddy 3 – by +xcdfgt



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:

RegexBuddy 3 – by +xcdfgt

004647C8	84C0	TEST AL,AL	
004647CA	74 18	JE SHORT 004647E4	RegexBud.004647E4
004647CC	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004647CF	F680 58030000	TEST BYTE PTR DS:[EAX+358],8	
004647D6	75 0C	JNZ SHORT 004647E4	RegexBud.004647E4
004647D8	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004647DB	80B8 77020000	CMP BYTE PTR DS:[EAX+277],1	
004647E2	75 21	JNZ SHORT 00464805	RegexBud.00464805
004647E4	8D55 E0	LEA EDI,DWORD PTR SS:[EBP-20]	
004647E7	A1 B8CE7900	MOV EAX,DWORD PTR DS:[79CEB8]	
004647EC	E8 1735FAFF	CALL 00407D08	RegexBud.00407D08
004647F1	8B4D E0	MOV ECX,DWORD PTR SS:[EBP-20]	
004647F4	B2 01	MOV DL,1	
004647F6	A1 B0C34100	MOV EAX,DWORD PTR DS:[41C3B0]	
004647FB	E8 58B5FAFF	CALL 0040FD58	RegexBud.0040FD58
00464800	E8 AF08FAFF	CALL 004050B4	RegexBud.004050B4
00464805	E8 4244FAFF	CALL 00408C4C	GetCapture
0046480A	85C0	TEST EAX,EAX	
0046480C	74 11	JE SHORT 0046481F	RegexBud.0046481F
0046480E	6A 00	PUSH 0	IParan = 0
00464810	6A 00	PUSH 0	wParam = 0
00464812	6A 1F	PUSH 1F	Message = WM_CANCELMODE
00464814	E8 3344FAFF	CALL 00408C4C	GetCapture
00464819	50	PUSH EAX	hWnd
0046481A	E8 A547FAFF	CALL 00408FC4	SendMessageA
0046481F	E8 6047FAFF	CALL 00408F84	ReleaseCapture
00464824	A1 0C2C7A00	MOV EAX,DWORD PTR DS:[7A2C0C]	
00464829	E8 0A280000	CALL 00467038	RegexBud.00467038

Lo que tenemos que tener en cuenta, es que nuestra llamada al Api **GetCapture**, tenga abajo llamadas a las Apis **SendMessageA** 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**.

00464792	C3	RETN	
00464793	90	NOP	
00464794	55	PUSH EBP	
00464795	8BEC	MOV EBP,ESP	
00464797	83C4 E0	ADD ESP,-20	
0046479A	53	PUSH EBX	
0046479B	56	PUSH ESI	
0046479C	33D2	XOR EDI,EDI	
0046479E	8955 E0	MOV DWORD PTR SS:[EBP-20],EDI	
004647A1	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
004647A4	33C0	XOR EAX,EAX	
004647A6	55	PUSH EBP	
004647A7	68 864A4600	PUSH 464A86	
004647AC	64:FF30	PUSH DWORD PTR FS:[EAX]	
004647AF	64:8920	MOV DWORD PTR FS:[EAX],ESP	
004647B2	E8 0DA80000	CALL 0046EFC4	RegexBud.0046EFC4
004647B7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004647BA	8078 57 00	CMP BYTE PTR DS:[EAX+57],0	
004647BE	75 24	JNZ SHORT 004647E4	RegexBud.004647E4
004647C0	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
004647C3	8B10	MOV EDI,DWORD PTR DS:[EAX]	
004647C5	FF52 50	CALL DWORD PTR DS:[EDI+50]	
004647C8	84C0	TEST AL,AL	
004647CA	74 18	JE SHORT 004647E4	RegexBud.004647E4

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.

RegexBuddy 3 – by +xcdfgt

The screenshot shows a debugger window with the following components:

- Assembly List:** A list of assembly instructions with addresses, hex values, and mnemonics. Key instructions include:
 - 00464794: 55 PUSH EBP
 - 00464795: 8BEC MOV EBP,ESP
 - 00464797: 83C4 E8 ADD ESP,-20
 - 00464799: 53 PUSH EBX
 - 0046479B: 56 PUSH ESI
 - 0046479D: 33D2 XOR EDI,EDI
 - 0046479E: 8955 E8 MOV DWORD PTR SS:[EBP-20],EDI
 - 004647A1: 8945 FC MOV DWORD PTR SS:[EBP-4],EBX
 - 004647A4: 33D8 XOR EAX,EAX
 - 004647A6: 55 PUSH EBP
 - 004647A7: 68 86404600 PUSH 464086
 - 004647AC: 64:FF30 PUSH DWORD PTR FS:[EAX]
 - 004647AF: 64:8920 MOV DWORD PTR FS:[EAX],ESP
 - 004647B2: E8 00408000 CALL 00408000
 - 004647B7: 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
 - 004647B9: 8B78 57 00 CMP BYTE PTR DS:[EAX+57],0
 - 004647BE: 75 24 JNZ SHORT 004647E4
 - 004647C0: 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
 - 004647C3: 8B10 MOV EDI,DWORD PTR DS:[EAX]
 - 004647C5: FF52 50 CALL DWORD PTR DS:[EDI+50]
 - 004647C8: 84C0 TEST AL,AL
 - 004647CA: 74 18 JE SHORT 004647E4
 - 004647CC: 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
 - 004647CF: F680 50800000 TEST BYTE PTR DS:[EAX+50],0
 - 004647D6: 75 0C JNZ SHORT 004647E4
 - 004647D8: 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
 - 004647DB: 8B88 77820000 CMP BYTE PTR DS:[EAX+277],1
 - 004647E2: 75 21 JNZ SHORT 00464805
- Registers (FPU):** A list of registers and their values. Key registers include:
 - EAX: 01823760
 - ECX: 009A0E4
 - EDX: 006F1C38
 - EBX: 0013FE14
 - ESP: 0013FC38
 - EBP: 0013FC48
 - ESI: 01829C10
 - EDI: 0013FE8C
 - EIP: 00464794
- Disassembly:** A list of instructions with addresses, hex values, and mnemonics. Key instructions include:
 - 0013FC38: 0072AC78 RETURN to RegExBud.0072AC78
 - 0013FC3C: 0013FDE4 Pointer to next SEH record
 - 0013FC40: 0070AC30 SE handler
 - 0013FC44: 0013FC48
 - 0013FC48: 0013FD74
 - 0013FC4C: 0047218E RETURN to RegExBud.0047218E
 - 0013FC50: 0013FE8C
 - 0013FC54: 011E0C95
 - 0013FC58: 0013FE14
 - 0013FC5C: FFFFFFFF
 - 0013FC60: 001630F0
 - 0013FC64: 004A02C2
 - 0013FC68: 0013FC78
 - 0013FC6C: 74738219 RETURN to MSCVF.74738219 from MSCVF.7472F480
 - 0013FC70: 001630F0

A red arrow points to the instruction 'RETURN to RegExBud.0072AC78' in the disassembly pane.

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

RETURN to RegExBud 0072AC78.

Vamos a esta dirección

RegexBuddy 3 – by +xcdfgt

0070AC43	8B0D B0CC7900	MOV ECX,DWORD PTR DS:[79CCB0]	RegexBud.008AA0E4
0070AC49	A1 28D27900	MOV EAX,DWORD PTR DS:[79D228]	
0070AC4E	8B0D	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	8B0D	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.

0070AC43	8B0D B0CC7900	MOV ECX,DWORD PTR DS:[79CCB0]	RegexBud.008AA0E4
0070AC49	A1 28D27900	MOV EAX,DWORD PTR DS:[79D228]	
0070AC4E	8B0D	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	8B0D	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	
0070AC8F	C3	RETN	
0070AC90	^ E9 7BA2CFFF	JMP 00404F10	
0070AC95	^ EB EE	JMP SHORT 0070AC85	
0070AC97	5D	POP EBP	
0070AC98	C3	RETN	
0070AC99	8D40 00	LEA EAX,DWORD PTR DS:[EAX]	

Backup

Copy

Binary

Assemble Space

Label :

Comment ;

Breakpoint

Run trace

Edit Ctrl+E

Fill with 00's

Fill with N0Ps

Binary copy

Binary paste

New origin here Ctrl+Gray *

Go to

Thread

Follow in Dump

Search for

Find references to

View

Copy to executable

Analysis

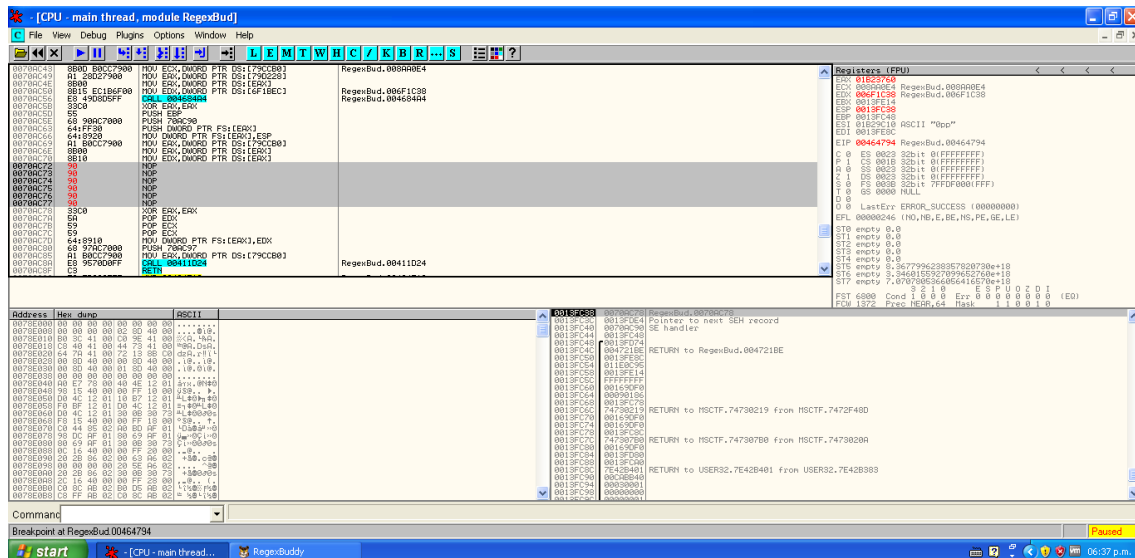
Detach Process

Process Patcher

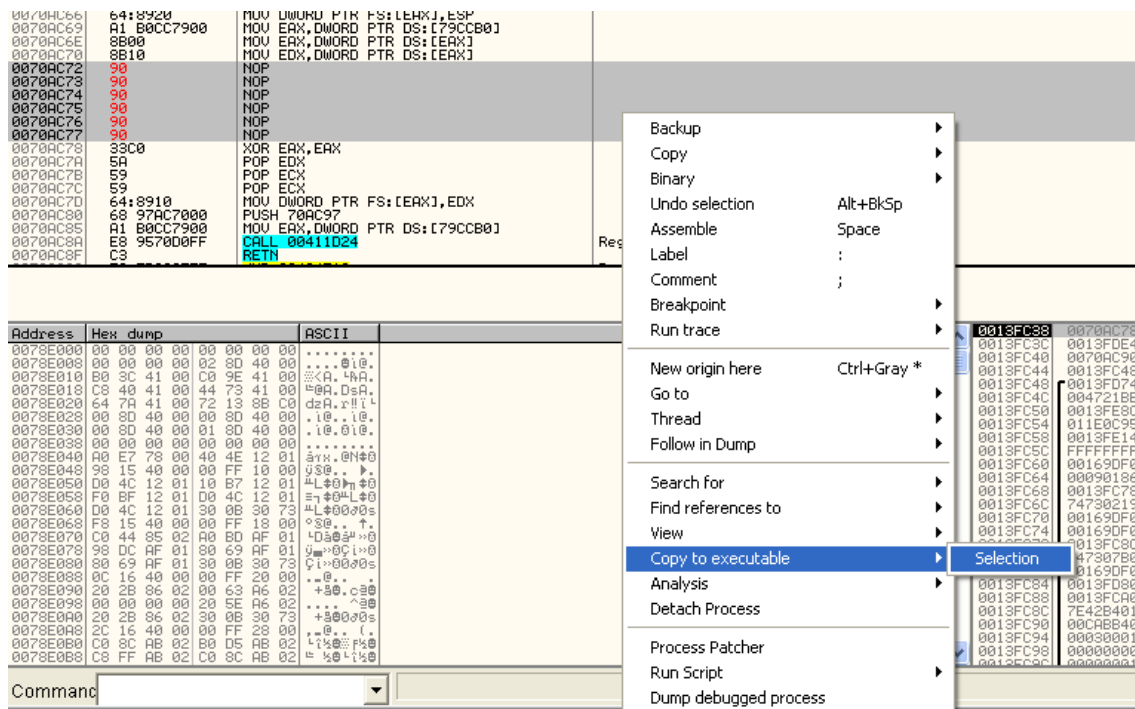
Address	Hex dump	ASCII
0078E000	00 00 00 00 00 00 00 00
0078E008	00 00 00 00 02 8D 40 00	...010
0078E010	00 3C 41 00 C0 9E 41 00	3CA.4A.
0078E018	C8 40 41 00 44 73 41 00	8QA.DsA.
0078E020	64 7A 41 00 72 13 8B C0	d2A.r11L
0078E028	00 30 40 00 00 8D 40 00	...010
0078E030	00 8D 40 00 01 8D 40 00	...010
0078E038	00 00 00 00 00 00 00 00
0078E040	A0 E7 78 00 40 4E 12 01	ax.0N40
0078E048	98 15 40 00 00 FF 10 00	...010
0078E050	D0 4C 12 01 10 B7 12 01	L01010
0078E058	F0 BF 12 01 D0 4C 12 01	=00L010
0078E060	D0 4C 12 01 30 0B 30 73	L00000s
0078E068	F8 15 40 00 00 FF 18 00	...010
0078E070	C0 44 85 02 A0 BD AF 01	0a0010
0078E078	98 DC AF 01 80 69 AF 01	000010
0078E080	80 69 AF 01 30 0B 30 73	010000s
0078E088	0C 16 40 00 00 FF 20 00	...010
0078E090	0A 00 00 00 00 00 00 00

Después del procedimiento quedaría así

RegexBuddy 3 – by +xcdfgt

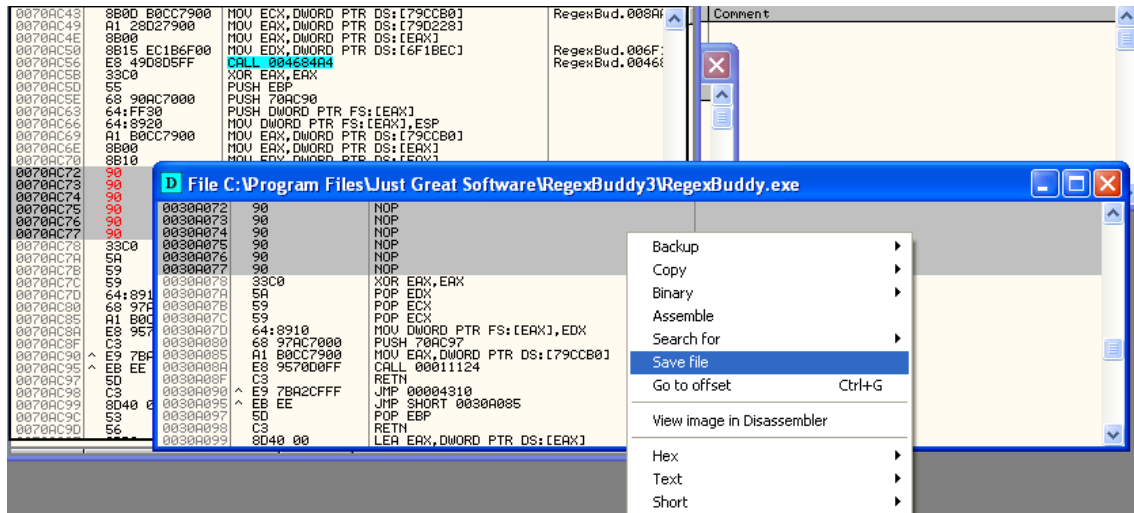


Guardamos los cambios realizados



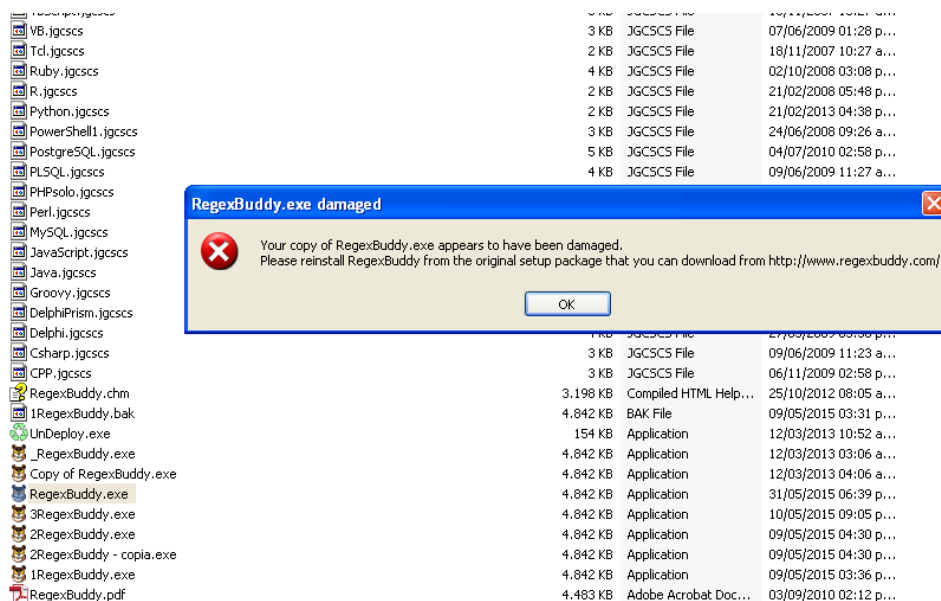
Seguido esto

RegexBuddy 3 – by +xcdfgt



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.

RegexBuddy 3 – by +xcdfgt

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	> 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	> 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	> 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	> 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

00405389	. 64:8920	MOV DWORD PTR FS:[EAX],ESP	
0040538C	. 3BFB	CMP EDI,EBX	
0040538E	> 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	> 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	
004053AF	. EB 14	MOV SHORT 004053C5	RegexBud.004053C5
004053B1	. E9 A68FFFFF	JMP 00404C5C	RegexBud.00404C5C
004053B6	. EB 45FFFFFF	CALL 00405300	RegexBud.00405300
004053B8	. EB 1CFDFFFF	CALL 0040580C	RegexBud.0040580C
004053C0	. E9 68FDFFFF	CALL 00405130	RegexBud.00405130
004053C5	. 5F	POP EDI	
004053C6	. 5E	POP ESI	
004053C7	. 5B	POP EBX	
004053C8	. 59	POP ECX	
004053C9	. 5D	POP EBP	
004053CA	. C3	RETN	

Registers (FPU)

EAX 00000000

ECX 00000000

EDX 00000001

EBX 00000000

ESP 0013F564

EBP 0013F458

ESI 00000000

EDI 00000000

EIP 7C90E514 ntddll.KiFastSys

C 0 ES 0023 32bit 0FFFFFFFF

P 1 CS 001B 32bit 0FFFFFFFF

A 0 SS 0023 32bit 0FFFFFFFF

Z 1 DS 0023 32bit 0FFFFFFFF

S 0 FS 003B 32bit 7FFD0000(F

T 0 GS 0000 NULL

D 0

O 0 LastErr ERROR_SUCCESS (0

EFL 00000246 (NO,NO,E,BE,NS,P

ST0 empty 0.0

ST1 empty 0.0

ST2 empty 0.0

empty 0.0

empty 0.0

empty 8.44944822782217720

empty 7.31170524531550098

empty 8.24368017969014674

1000 Cond 0 0 0 0 Err 0

1372 Prec NEAR,64 Nask

RegexBuddy.exe damaged

Your copy of RegexBuddy.exe appears to have been damaged.

Please reinstall RegexBuddy from the original setup package that you can download from <http://www.regexbuddy.com/>

OK

Jump from 0040538E

Address	Hex	dump	ASCII
0078E000	00 00 00 00	00 00 00 00
0078E008	00 00 00 00	02 00 40 00	...@..
0078E010	00 3C 41 00	00 3E 41 00	...A..
0078E018	03 40 41 00	44 73 41 00	...d..
0078E020	64 7A 41 00	72 13 8B C0	dZa..
0078E028	00 00 40 00	00 00 40 00	...@..
0078E030	00 00 40 00	00 00 40 00	...@..
0078E038	00 00 00 00	00 00 00 00
0078E040	A0 E7 78 00	40 4E 12 01	...x..
0078E048	58 15 40 00	FF 10 00 00	...y..
0078E050	4C E0 78 00	59 57 12 01	...L..
0078E058	F9 0F 12 01	00 4C 12 01	...f..
0078E060	4C E0 78 00	30 00 30 73	...L..
0078E068	F8 15 40 00	FF 18 00 00	...f..

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 cerca 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.

The screenshot shows a debugger window with assembly code on the left, registers on the right, and a central error message. The assembly code includes instructions like `XOR EAX, EAX`, `PUSH EBP`, `PUSH 4053B1`, `PUSH DWORD PTR FS:[EAX]`, `MOV DWORD PTR FS:[EAX], ESP`, `CMP EDI, EBK`, `JLE SHORT 004053A7`, `MOV EAX, DWORD PTR SS:[EBP-4]`, `MOV ESI, DWORD PTR DS:[EAX+EBX*8]`, `INC EBX`, `MOV DWORD PTR DS:[78D7D4], EBK`, `TEST ESI, ESI`, `JE SHORT 004053A3`, `CALL ESI`, `CMP EDI, EBK`, `JG SHORT 00405390`, `XOR EAX, EAX`, `POP EDI`, `POP ECX`, `POP ECX`, `MOV DWORD PTR FS:[EAX], EDI`, `JMP SHORT 004053C5`, `JMP 00404C5C`, `CALL 004053B0`, `CALL 004053C0`, `CALL 00405130`, `POP EDI`, `POP ESI`. The registers window shows `ESI: 0078D9E4` with a red arrow pointing to it. The error message says 'RegexBuddy.exe damaged' and 'Your copy of RegexBuddy.exe appears to have been damaged. Please reinstall RegexBuddy from the original setup package that you can download from http://www.regexbuddy.com/'. The bottom of the assembly window shows a hex dump and ASCII view.

RegexBuddy 3 – by +xcdfgt

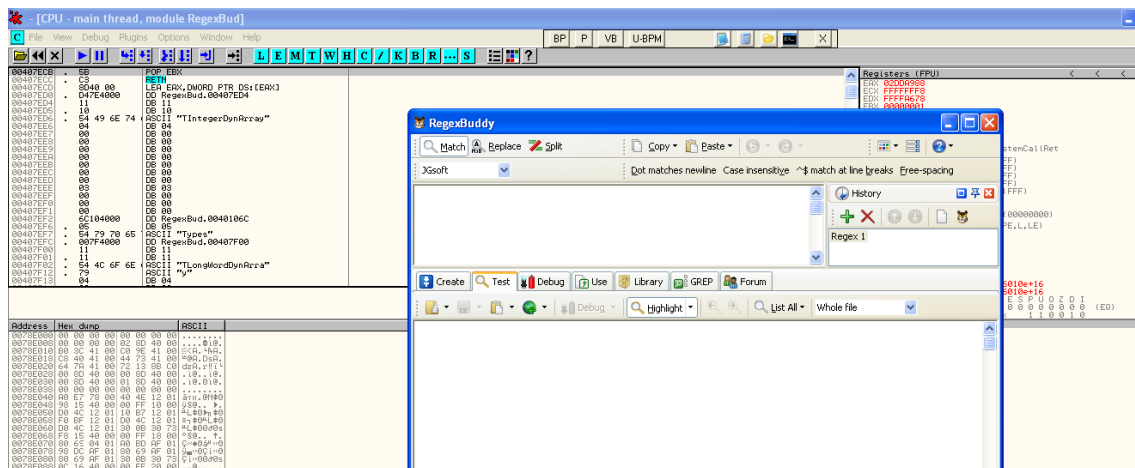
Después de volver a ejecutarlo, y parar en el breakpoint, cuando **ESI** valga 0078D9E4 vemos que el CALL ESI ejecuta la validación que nos saca de la ejecución normal del programa, con esto podemos concluir que dentro del CALL ESI si esta el punto que debemos anular para que el error se genere.

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 740A7800	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:[8A120],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.

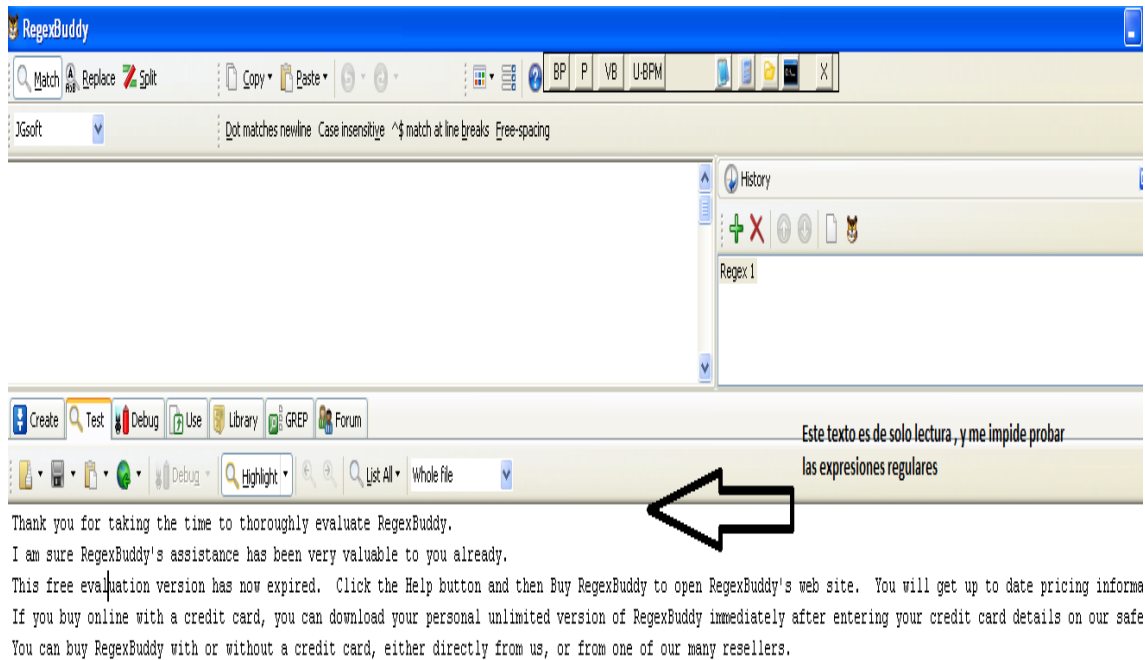
WALA, nuestro programa carga sin problemas, tal como se muestra en la imagen



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

RegexBuddy 3 – by +xcdfgt

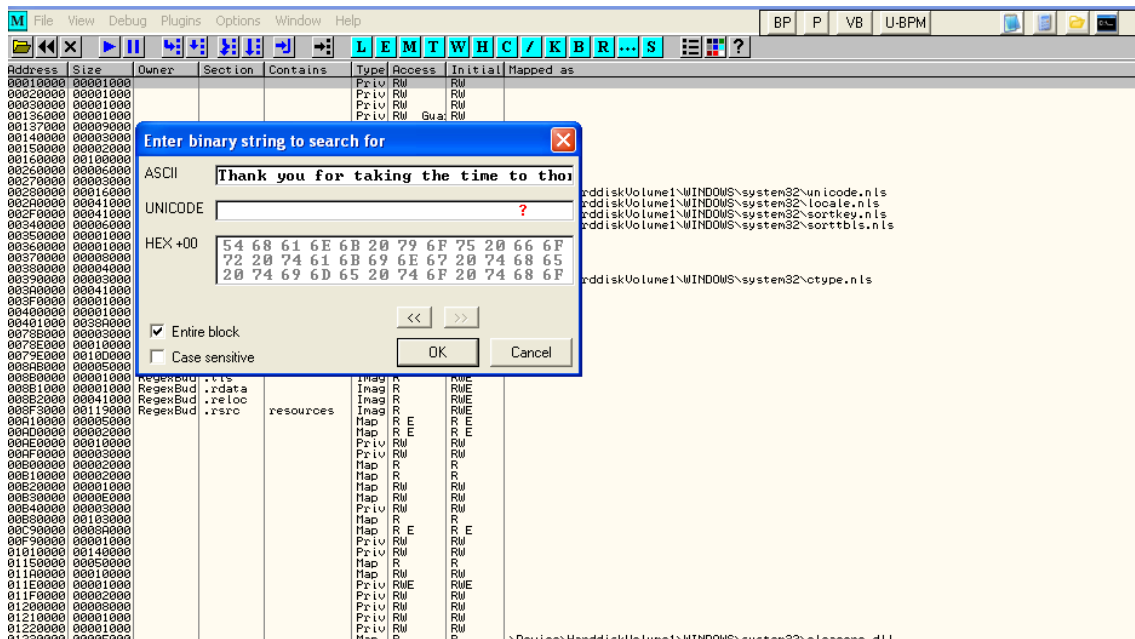


Revisando el comportamiento, analizamos que este texto se pone, a penas nosotros damos click inmediatamente, para solucionar esto primero voy a buscar en memoria donde se está copiando este texto, damos un search en la memoria.

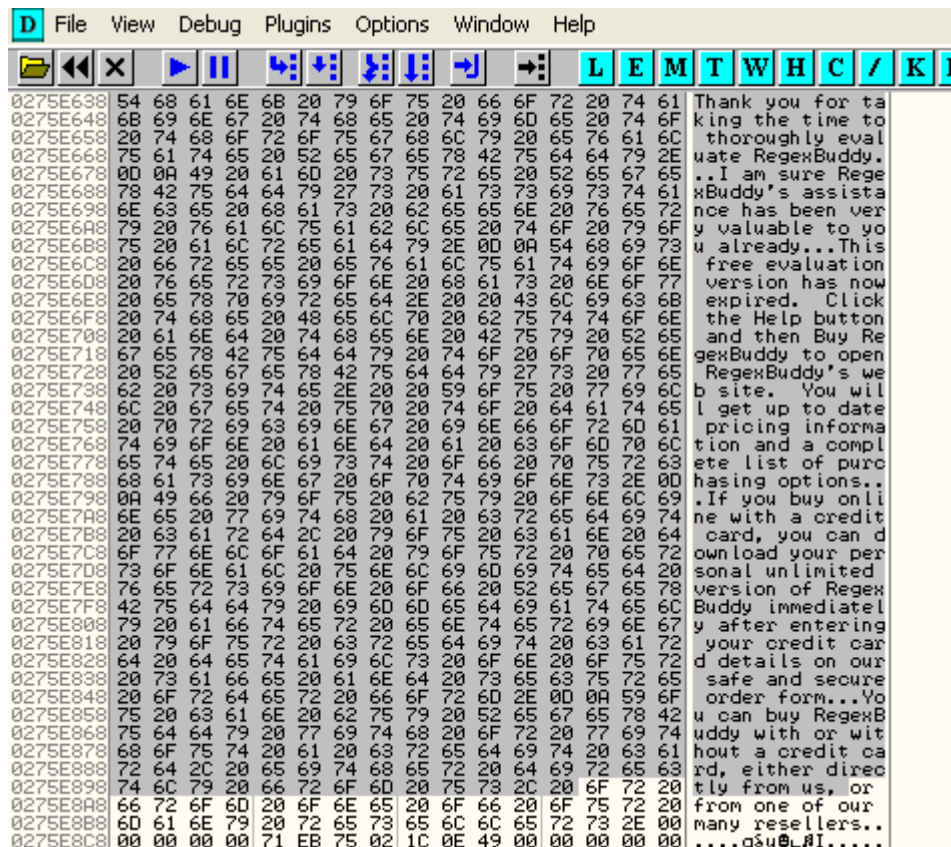
Damos click en la opción M del toolbox de arriba, y copiamos el texto a buscar el cual es el siguiente:

**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.**

RegexBuddy 3 – by +xcdfgt

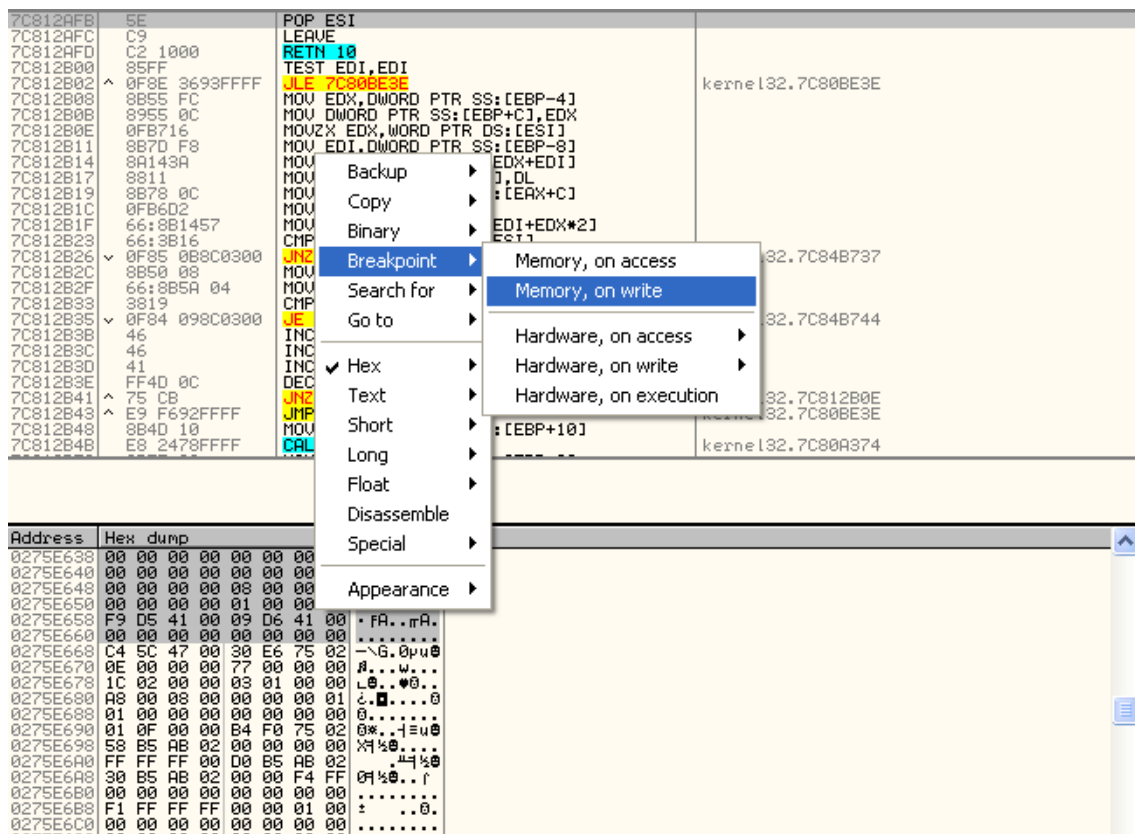
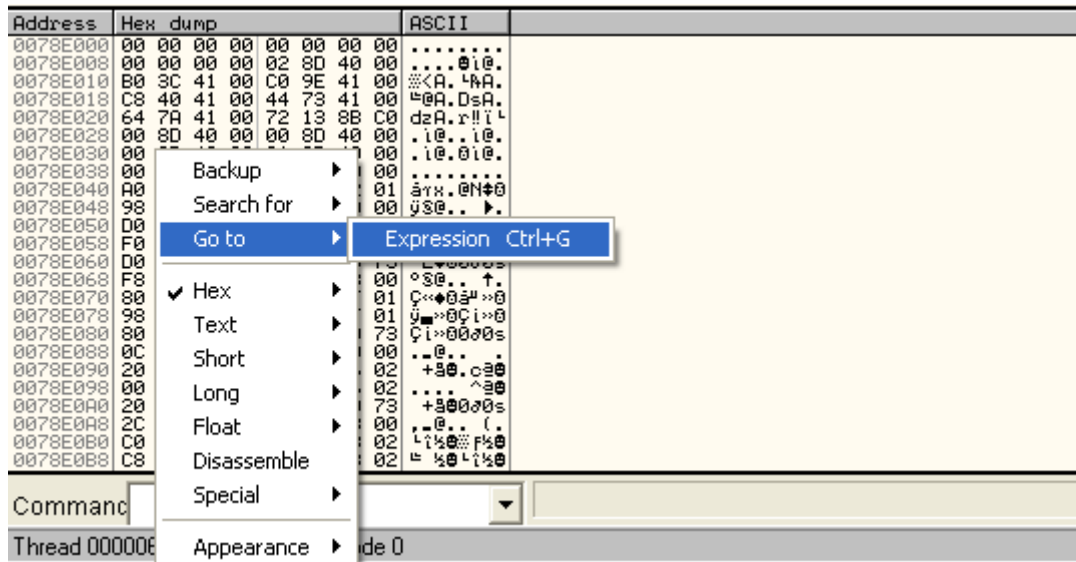


Damos ok, y nos lleva a la siguiente zona.



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**

RegexBuddy 3 – by +xcdfgt



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.

RegexBuddy 3 – by +xcdfgt

The screenshot shows a debugger window with three main panes:

- Assembly Window:** Displays assembly code. The instruction `RETN` at address `00403335` is highlighted. Other instructions include `FISTP QWORD PTR DS:[EDX]`, `POP EDX`, `LEA SHORT 00403335`, `CMP EAX, EDI`, `JG SHORT 004032D0`, `SUB EDI, ECX`, `CMP EAX, EDI`, `LEA EDI, QWORD PTR DS:[ECX+EDI]`, `JG SHORT 004032D0`, `SUB EDI, 8`, `PUSH EDI`, `FILD QWORD PTR DS:[ECX+EDI]`, `FISTP QWORD PTR DS:[EDI]`, `SUB EDI, 8`, `JG SHORT 00403324`, `POP EDI`, `FISTP QWORD PTR DS:[EDI]`, `FISTP QWORD PTR DS:[EDI]`, `MOVZX ECX, BYTE PTR DS:[EDI]`, and `MOV BYTE PTR DS:[EDI], CL`.
- Registers (FPU) Window:** Shows the state of registers. The `EAX` register contains `00405B00`, which is the address of the `RETN` instruction. Other registers like `ECX`, `EDX`, `EBX`, `ESP`, `EIP`, `EFL`, `ST0`, `ST1`, `ST2`, `ST3`, `ST4`, `ST5`, `ST6`, and `ST7` are also shown.
- Memory Dump Window:** Shows the contents of memory starting at address `00275E30`. The dump includes hex values and their corresponding ASCII representations.

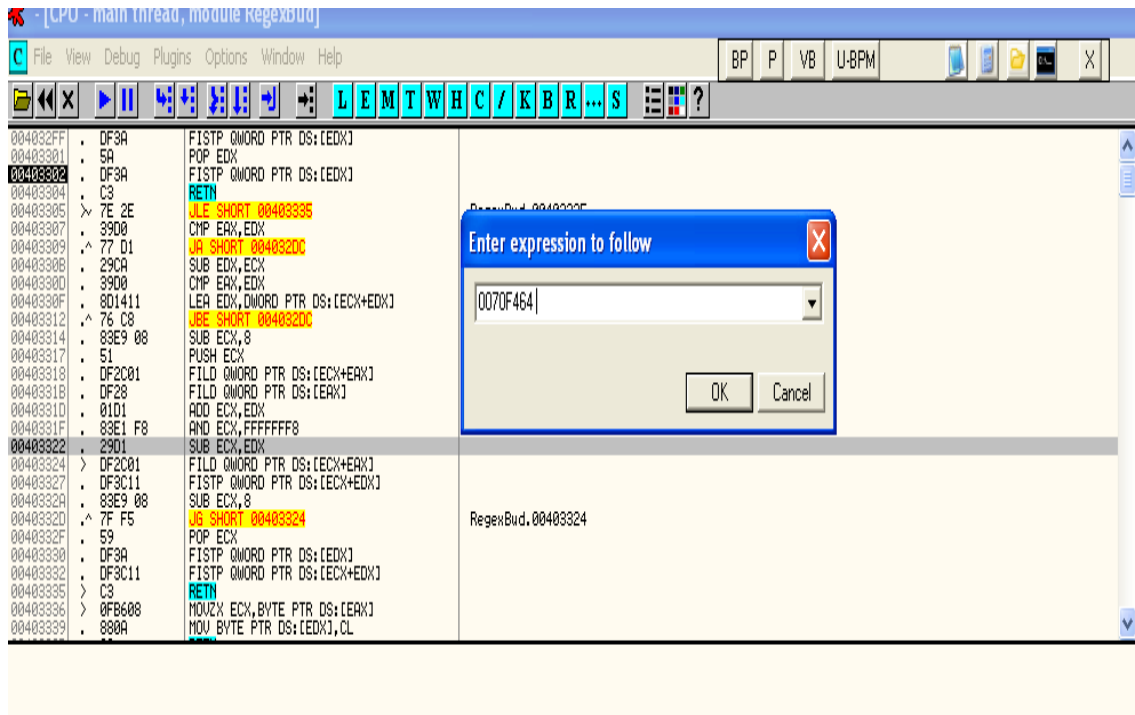
Esta zona, es el área caliente y observando la pila vemos unos Return, los cuales nos pueden llevar a la zona definitiva,

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

RegexBuddy 3 – by +xcdft



La cual nos lleva a la siguiente zona

0070F42A	FF75 EC	PUSH DWORD PTR SS:[EBP-14]	
0070F42D	68 DCF47000	PUSH 70F4DC	ASCII "٠"
0070F432	8D55 E8	LEA EDI, DWORD PTR SS:[EBP-18]	
0070F435	A1 B4CD7900	MOV EAX, DWORD PTR DS:[79CDB4]	RegexBud.00407D08
0070F43A	E8 C98CFFFF	CALL 00407D08	
0070F43F	FF75 E8	PUSH DWORD PTR SS:[EBP-18]	
0070F442	68 DCF47000	PUSH 70F4DC	ASCII "٠"
0070F447	8D55 E4	LEA EDI, DWORD PTR SS:[EBP-1C]	
0070F44A	A1 F0D57900	MOV EAX, DWORD PTR DS:[79D5F0]	RegexBud.00407D08
0070F44F	E8 B48CFFFF	CALL 00407D08	
0070F454	FF75 E4	PUSH DWORD PTR SS:[EBP-1C]	
0070F457	8D45 F8	LEA EDI, DWORD PTR SS:[EBP-8]	
0070F45A	BA 09000000	MOV EDI, 9	
0070F45F	E8 9C6CFFFF	CALL 00405B00	RegexBud.00405B00
0070F464	8B55 F8	MOV EDI, DWORD PTR SS:[EBP-8]	
0070F467	8D45 FC	LEA EDI, DWORD PTR SS:[EBP-4]	
0070F46A	E8 896CFFFF	CALL 004060F8	RegexBud.004060F8
0070F46F	8B55 FC	MOV EDI, DWORD PTR SS:[EBP-4]	
0070F472	8B83 A0050000	MOV EAX, DWORD PTR DS:[EBX+5A0]	
0070F475	8B80 E0030000	MOV EAX, DWORD PTR DS:[EAX+3E0]	
0070F47E	E8 9D57E8FF	CALL 00594C20	RegexBud.00594C20
0070F483	8B83 A0050000	MOV EAX, DWORD PTR DS:[EBX+5A0]	
0070F486	8B80 E0030000	MOV EAX, DWORD PTR DS:[EAX+3E0]	
0070F48F	C640 39 01	MOV BYTE PTR DS:[EAX+39], 1	
0070F493	8B83 7C030000	MOV EAX, DWORD PTR DS:[EBX+37C]	
0070F499	E8 A22CE1FF	CALL 00522140	RegexBud.00522140
0070F49E	33D2	XOR EDI, EDI	
0070F4A0	8B08	MOV ECX, DWORD PTR DS:[EAX]	

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.

RegexBuddy 3 – by +xcdfgt

3070F3CD	8B83 7C030000	MOV EAX,DWORD PTR DS:[EBX+37C]	
3070F3D3	E8 682DE1FF	CALL 00522140	RegexBud.00522140
3070F3D8	8B10	MOV EDX,DWORD PTR DS:[EAX]	
3070F3DA	FF52 50	CALL DWORD PTR DS:[EDX+50]	
3070F3DD	84C0	TEST AL,AL	
3070F3DF	0F84 C0000000	JE 0070F4A5	RegexBud.0070F4A5
3070F3E5	A1 B4D37900	MOV EAX,DWORD PTR DS:[79D3B4]	
3070F3EA	8338 07	CMPL DWORD PTR DS:[EAX],7	
3070F3ED	0F8E B2000000	JLE 0070F4A5	RegexBud.0070F4A5
3070F3F3	8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	
3070F3F6	A1 54D87900	MOV EAX,DWORD PTR DS:[79D854]	
3070F3FB	E8 0889CFFF	CALL 00407D08	RegexBud.00407D08
3070F400	FF75 F4	PUSH DWORD PTR SS:[EBP-C]	
3070F403	68 DCF47000	PUSH 70F4DC	ASCII "J"
3070F408	8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
3070F40B	A1 10D97900	MOV EAX,DWORD PTR DS:[79D910]	
3070F410	E8 F388CFFF	CALL 00407D08	RegexBud.00407D08
3070F415	FF75 F0	PUSH DWORD PTR SS:[EBP-10]	
3070F418	68 DCF47000	PUSH 70F4DC	ASCII "J"
3070F41D	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
3070F420	A1 A8D27900	MOV EAX,DWORD PTR DS:[79D2A8]	
3070F425	E8 DE88CFFF	CALL 00407D08	RegexBud.00407D08
3070F42A	FF75 EC	PUSH DWORD PTR SS:[EBP-14]	
3070F42D	68 DCF47000	PUSH 70F4DC	ASCII "J"
3070F432	8D55 E8	LEA EDX,DWORD PTR SS:[EBP-18]	
3070F435	A1 B4CD7900	MOV EAX,DWORD PTR DS:[79CDB4]	
3070F43A	E8 C988CFFF	CALL 00407D08	RegexBud.00407D08
3070F43F	FF75 E8	PUSH DWORD PTR SS:[EBP-18]	

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.