

# Ardamax Keylogger

<b>Versión :</b>	Ardamax Keylogger - 4.6
<b>URL :</b>	<a href="https://www.ardamax.com/">https://www.ardamax.com/</a>
<b>Protección :</b>	Ninguna
<b>Dificultad :</b>	Newbie
<b>Herramientas :</b>	Olllydbg v1.10 / RDG Packer Detector v0.7.6 2017
<b>Objetivo :</b>	Registrarlo
<b>Reverser :</b>	L1oR
<b>Lugar / Fecha :</b>	Perú – 26/02/17

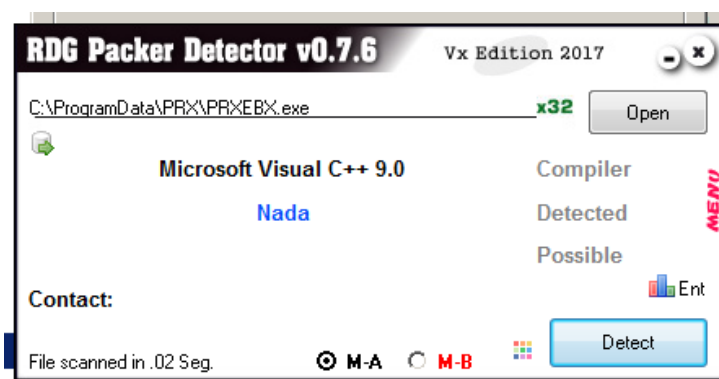
Hola amigos, hoy se explicare como reversear Ardamax Keylogger, con una ligero regla de registro, pero antes agradecer al Maestro Ricardo Narvaja por su nuevo aporte IDA PRO y sus exámenes propuestos, además al grupo Perú Crackeando, que le integra DavicoRm que aporta y nos enseña su método de punto mágico, a SoftDat por sus videos de cracking, y Abel por los retos propuestos y por motivarnos.

## INFORMACION DE LA VICTIMA

Es una herramienta que captura las actividades del usuario y guarda en un archivo de registro, El archivo de registro se puede ver en modo texto o vía web, también puede utilizar dicha herramienta para ver qué sucede con su computadora mientras estas ausente.

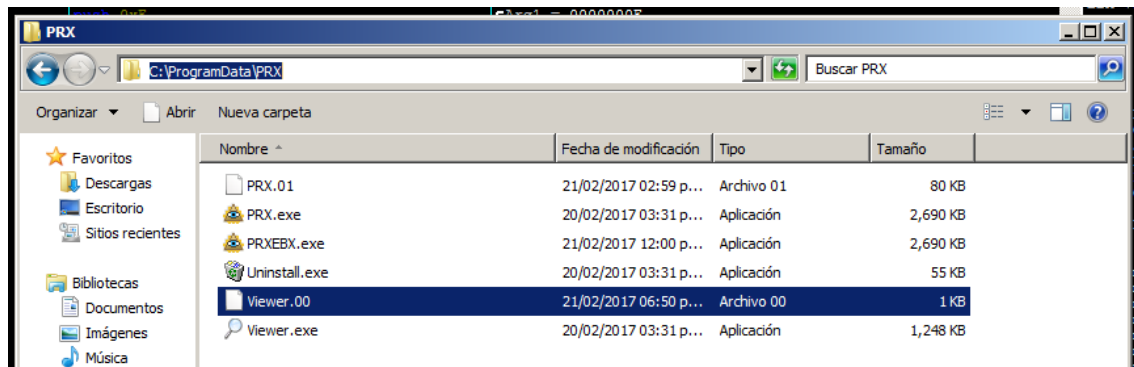
## AL ATAQUE

EL software tiene un tipo de seguridad que muchos conoces es la versión de prueba o TRIAL, de lo cual como siempre en casi todos mis futuros tutoriales, encontrar el “CHICO BUENO” es mi especialidad jaja, comenzaremos a sacar un BACKUP al .EXE, para futuras metidas de patas, bueno lo primero que se realizar es ver si Ardamax presenta protección o un unpack y el tipo de lenguaje que se compilo, en este caso vemos que no presenta protección y que está en lenguaje Visual C ++ 9.0.

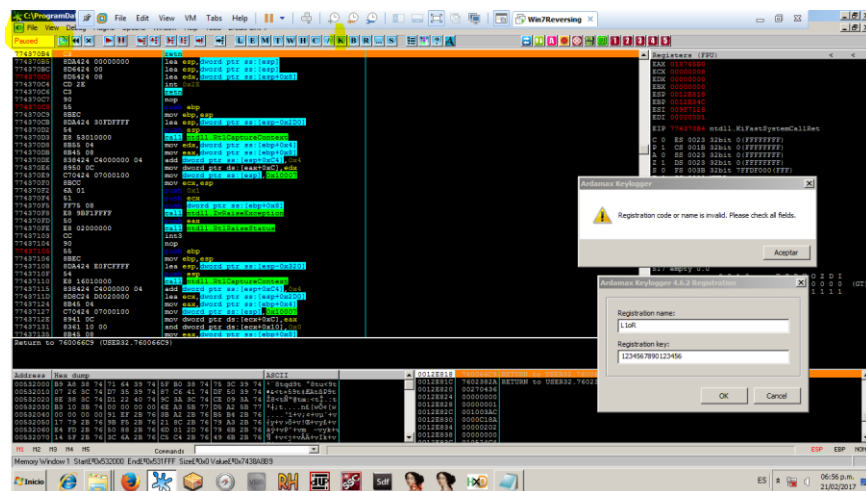


Nota:

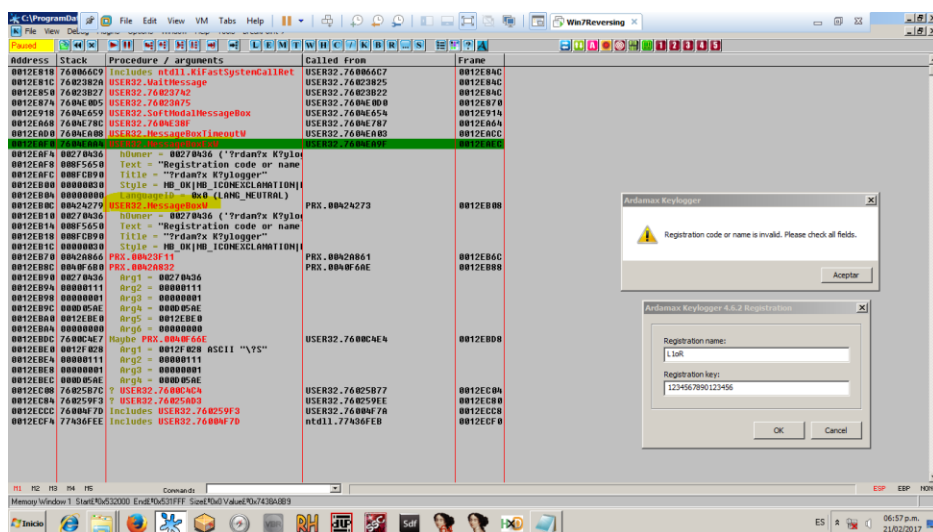
Tener en cuenta la ruta en donde se instala el software.



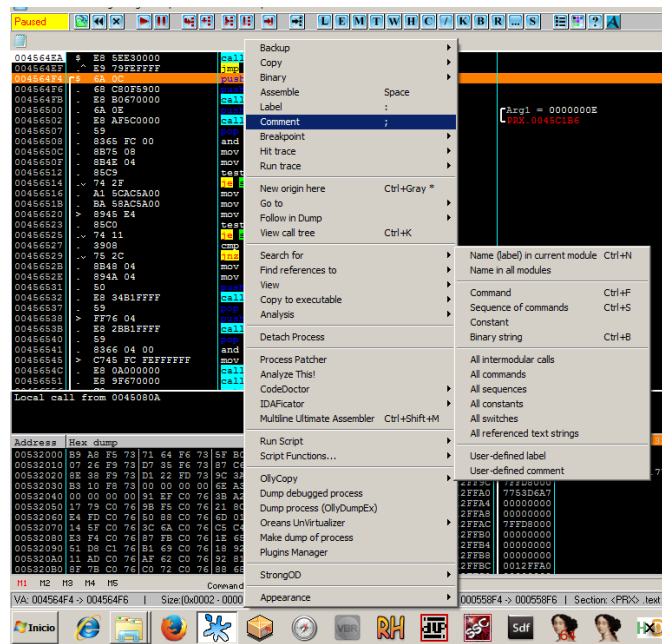
Para poder hallar el “BAD BOY”, tuve que ejecutar la aplicación “F9”, luego intente registrarme y bum me aparecio el “BAD BOY”, en donde pause. Y luego hice clic en la opcion “K”, como muestra en la imagen:



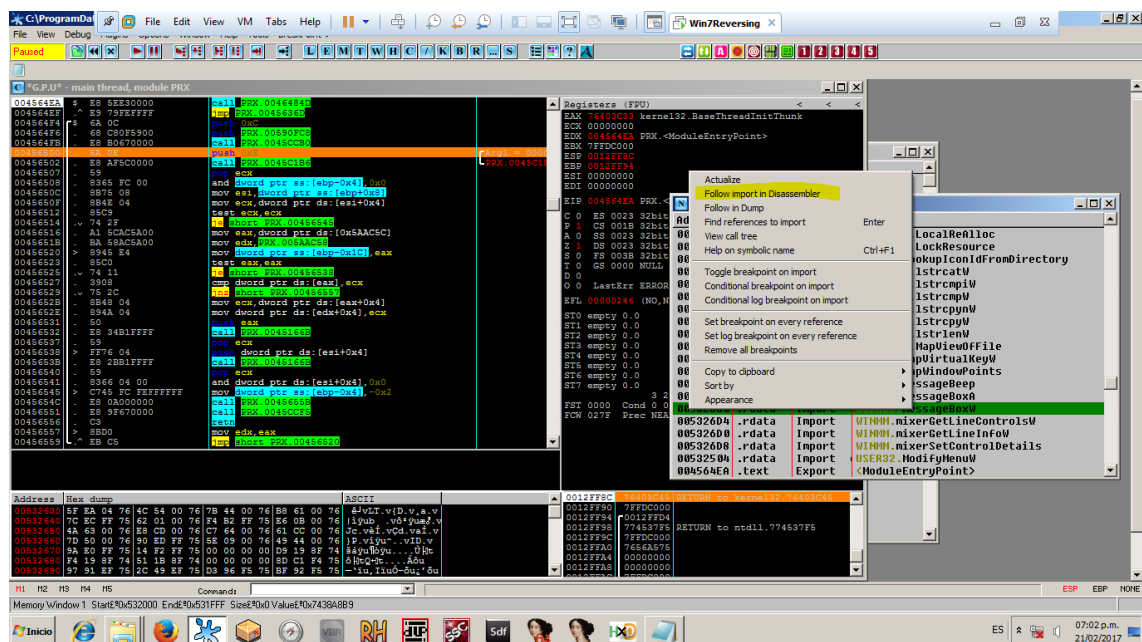
Como muestra la Opcion “K” o “VIEW-CALL STACK”, nos muestra el call que entramos al momento de poner PAUSA. En este caso vemos que MessageBoxW, tiene el proceso de valicacion.



Reiniciamos la aplicación y nos vamos a Search for -> Name (label) in current module.



Encontramos la clase MessageBoxW donde nos muestra el mensaje de el “BAD BOY”, de lo cual iremos al area de Disassembler de la clase.



En nuestro caso pondremos un BP en la direccion de memoria 7604EA5F (Inicio) y 7604EAA5 (FIN), de la clase MessageBoxW

```

7604EA5F 8BFF      mov     edi,edi
7604EA61 55        push    ebp
7604EA62 8BEC      mov     ebp,esp
7604EA64 833D 749A0576 00 cmp     dword ptr ds:[0x76059A74],0x0
7604EA6B 74 24      je      short USER32.7604EA91
7604EA6D 64:A1 18000000 mov     eax,dword ptr fs:[0x18]
7604EA73 6A 00      push    0x0
7604EA75 FF70 24    push    dword ptr ds:[eax+0x24]
7604EA78 68 A49E0576 push    USER32.7604EA91
7604EA7D FF15 3414FF75 call     near dword ptr ds:[<4KERNEL32.InterlockedCompareExchange
7604EA83 85C0      test     eax,eax
7604EA85 75 0A      jnz     short USER32.7604EA91
7604EA87 C705 A09E0576 01000000 mov     dword ptr ds:[0x76059EA0],0x1
7604EA91 6A 00      push    0x0
7604EA93 FF75 14    push    dword ptr ss:[ebp+0x14]
7604EA96 FF75 10    push    dword ptr ss:[ebp+0x10]
7604EA99 FF75 0C    push    dword ptr ss:[ebp+0xC]
7604EA9C FF75 08    push    dword ptr ss:[ebp+0x8]
7604EA9F E8 49FFFFFF call     USER32.MessageBoxW
7604EA44 5D        pop     ebp
7604EAA5 C2 1000    ret     0x10
7604EAA8 90        nop
7604EAA9 90        nop
7604EAAA 90        nop
7604EAB3 90        nop
7604EAB4 5D        pop     ebp
7604EAB6 8BFF      mov     edi,edi
7604EAB8 55        push    ebp
7604EAB9 8BEC      mov     ebp,esp
7604EABD 6A 07      push    0x7
7604EABF FF75 10    push    dword ptr ss:[ebp+0x10]
7604EAC1 FF75 0C    push    dword ptr ss:[ebp+0xC]
7604EAC3 FF75 08    push    dword ptr ss:[ebp+0x8]
7604EAC6 E8 DD630000 call     USER32.7604EA91
7604EAC8 5D        pop     ebp
7604EAC9 C2 0C00    ret     0xC
7604EACB 90        nop

```

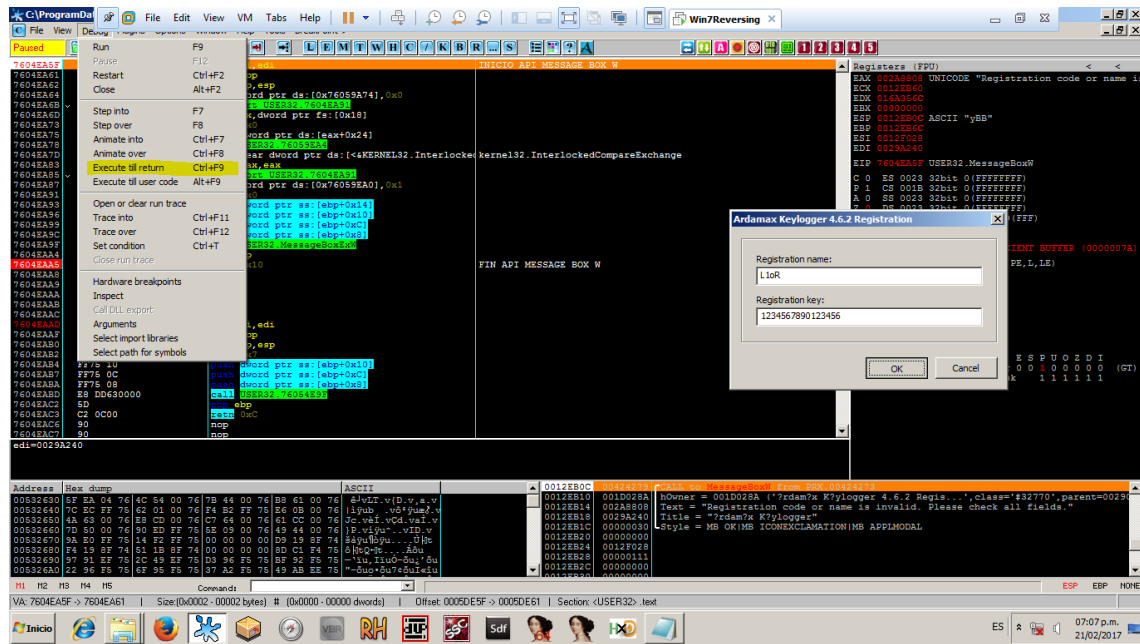
En este caso ejecute Ardamax con “F9”,e intente registrarme, donde se detuvo en el MessageBoxW, luego aprieto una vez mas “F9”.

```

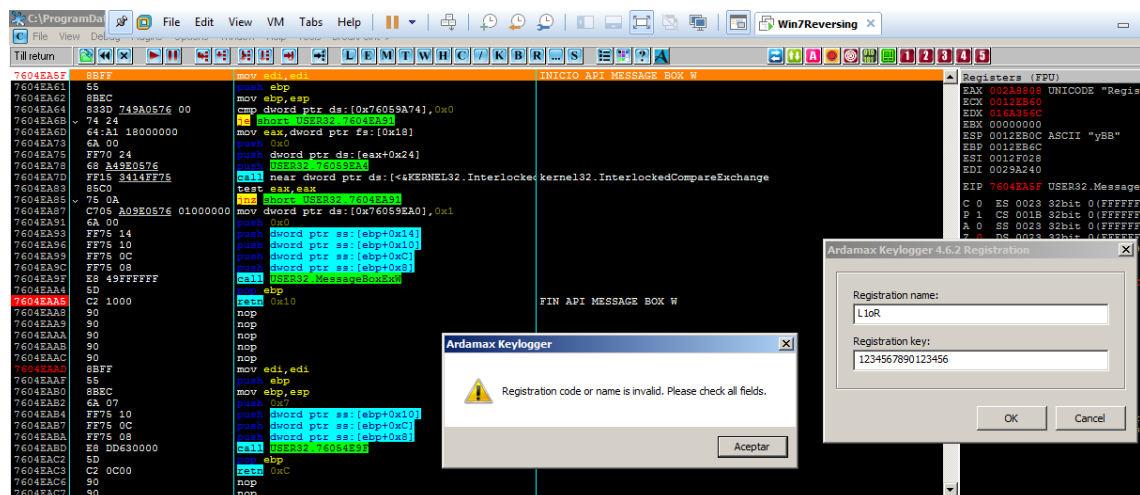
7604EA5F 8BFF      mov     edi,edi
7604EA61 55        push    ebp
7604EA62 8BEC      mov     ebp,esp
7604EA64 833D 749A0576 00 cmp     dword ptr ds:[0x76059A74],0x0
7604EA6B 74 24      je      short USER32.7604EA91
7604EA6D 64:A1 18000000 mov     eax,dword ptr fs:[0x18]
7604EA73 6A 00      push    0x0
7604EA75 FF70 24    push    dword ptr ds:[eax+0x24]
7604EA78 68 A49E0576 push    USER32.7604EA91
7604EA7D FF15 3414FF75 call     near dword ptr ds:[<4KERNEL32.InterlockedCompareExchange
7604EA83 85C0      test     eax,eax
7604EA85 75 0A      jnz     short USER32.7604EA91
7604EA87 C705 A09E0576 01000000 mov     dword ptr ds:[0x76059EA0],0x1
7604EA91 6A 00      push    0x0
7604EA93 FF75 14    push    dword ptr ss:[ebp+0x14]
7604EA96 FF75 10    push    dword ptr ss:[ebp+0x10]
7604EA99 FF75 0C    push    dword ptr ss:[ebp+0xC]
7604EA9C FF75 08    push    dword ptr ss:[ebp+0x8]
7604EA9F E8 49FFFFFF call     USER32.MessageBoxW
7604EA44 5D        pop     ebp
7604EAA5 C2 1000    ret     0x10
7604EAA8 90        nop
7604EAA9 90        nop
7604EAAA 90        nop
7604EAB3 90        nop
7604EAB4 5D        pop     ebp
7604EAB6 8BFF      mov     edi,edi
7604EAB8 55        push    ebp
7604EAB9 8BEC      mov     ebp,esp
7604EABD 6A 07      push    0x7
7604EABF FF75 10    push    dword ptr ss:[ebp+0x10]
7604EAC1 FF75 0C    push    dword ptr ss:[ebp+0xC]
7604EAC3 FF75 08    push    dword ptr ss:[ebp+0x8]
7604EAC6 E8 DD630000 call     USER32.7604EA91
7604EAC8 5D        pop     ebp
7604EAC9 C2 0C00    ret     0xC
7604EACB 90        nop

```

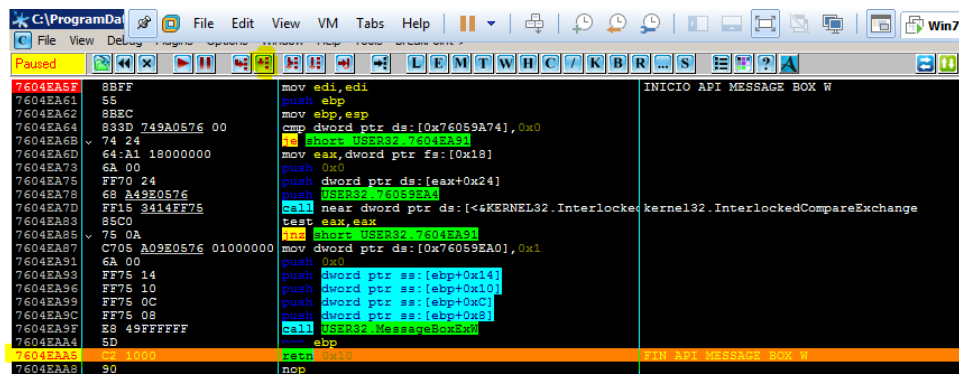
Otra manera de ir al final de la clase MessageBoxW, ir a Debug -> Execute till return.



Donde nos mostrara el MessageBox con el “BAD BOY”.

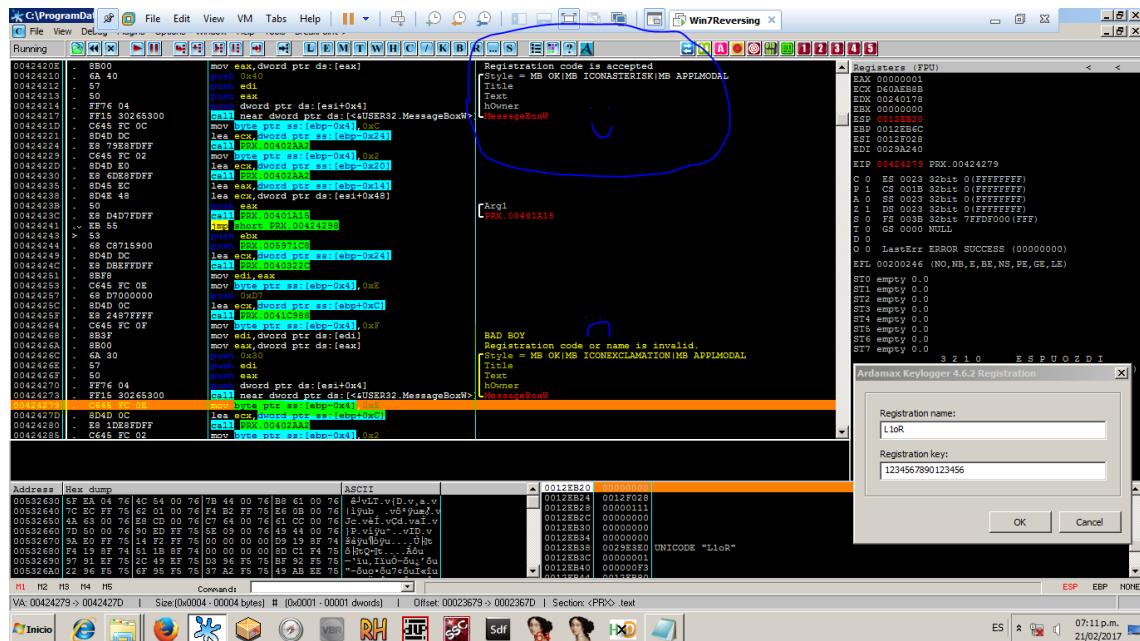


Cuando lleguemos a la direccion de memoria 7604EAA5, y finalmente apretamos “F8”, para retornar y ver que CALL realizo la invocacion de MessageBoxW en este caso se realizo en el 00424273.

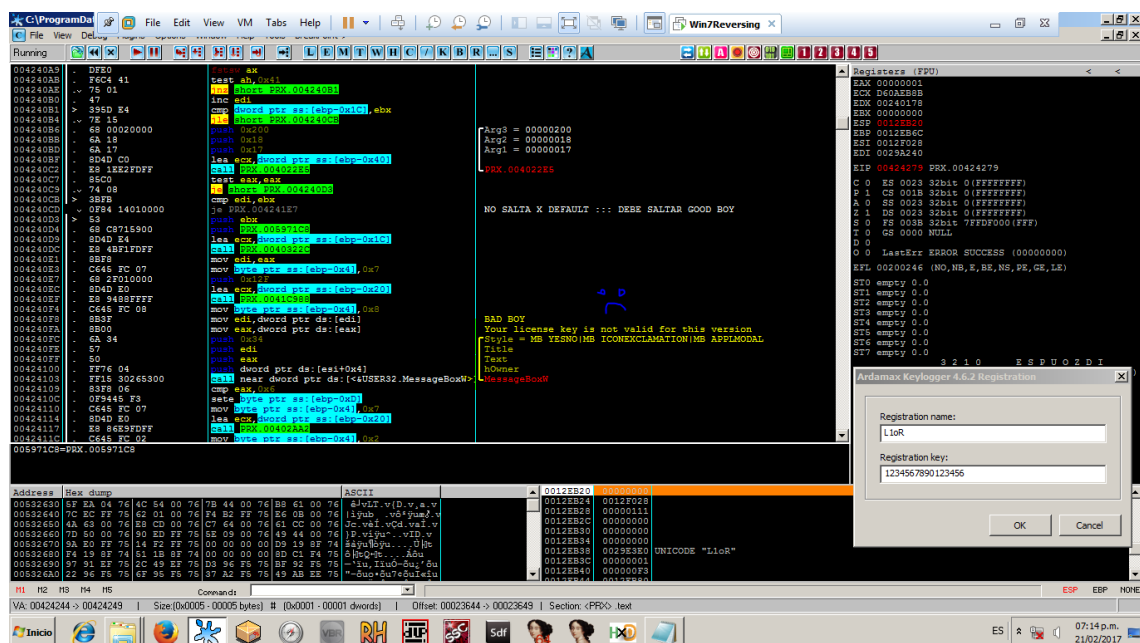




En el punto anterior se presiono “F8”, y llegamos a la direccion de memoria 00424279, en donde para no tardar tiempo en explicar, se analizo cada MessageBoxW, el de circulo azul con carita feliz, es el “GOOD BOY”, con la direccion de memoria 00424217 y el que esta sombreado con color amarillo con direccion de memoria 00424273, es el “BAD BOY”.

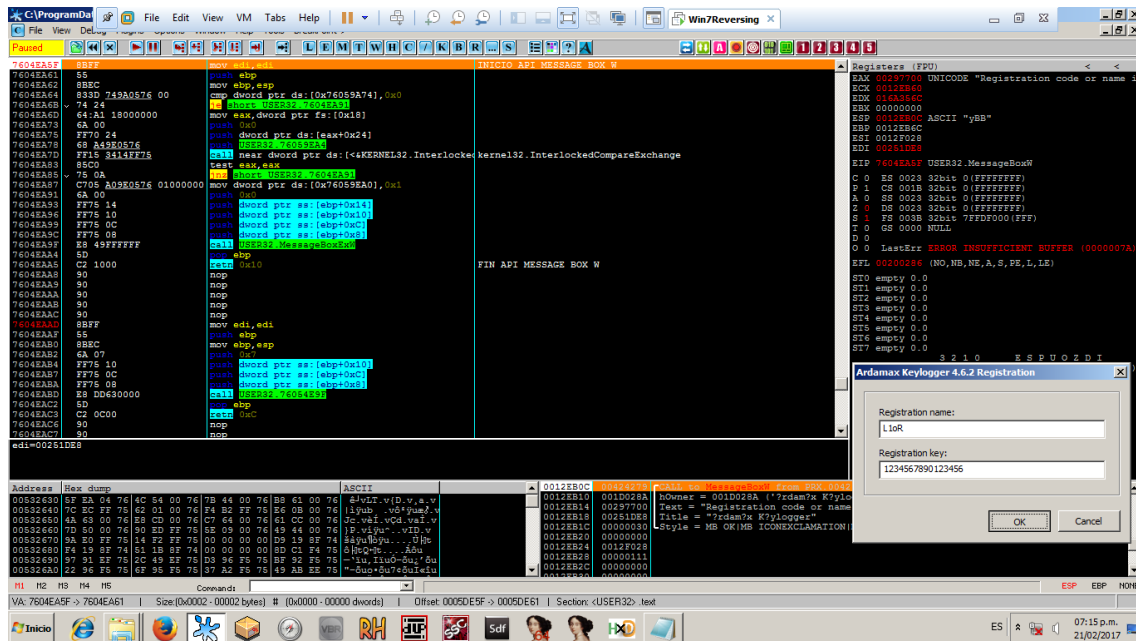


Este es otro MessageBoxW con direccion de memoria 00424103 y sombreado de amarillo es otro “BAD BOY”.

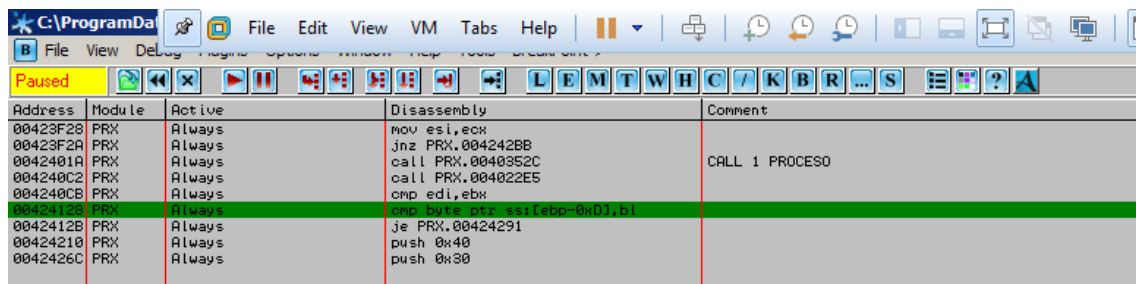


Nota:

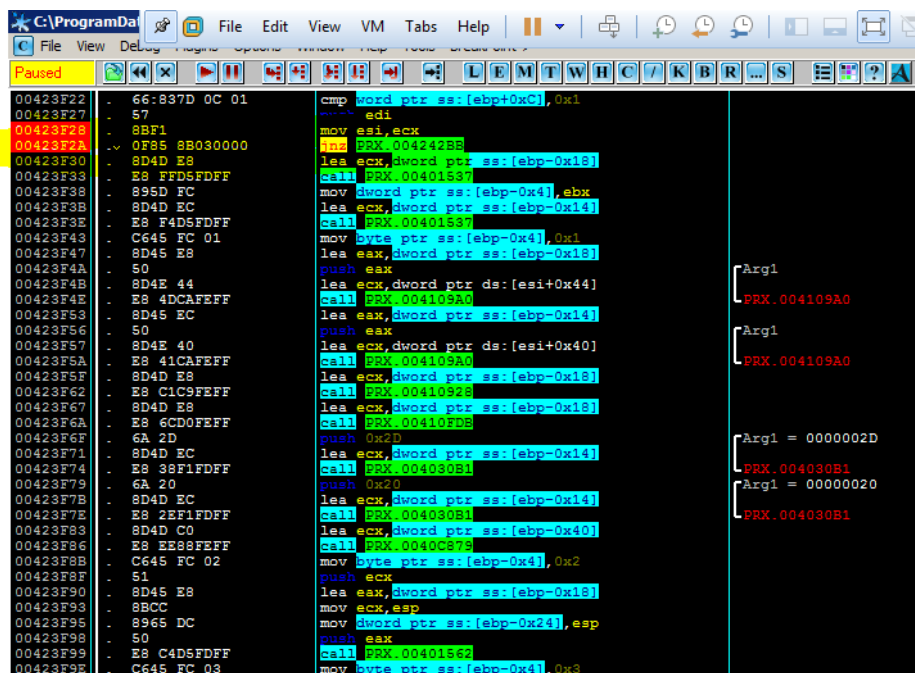
Antes de seguir con el analisis, quitar el BreakPoint de la clase MessageBoxW, sino estaríamos deteniendonos en cada momento.

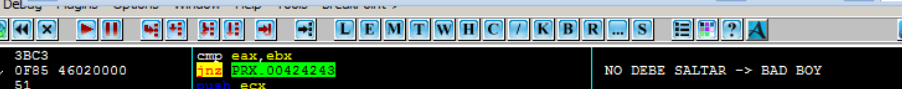


Sigamos con el analisis, hagamos clic en el boton “B”, para poder ver los BreakPoints que se asigno a cada direccion de memoria y poder realizar nuestro analisis.



En este caso presionamos “F9”, intentamos registrarnos y nos detenemos en el primer BreakPoint, si traceamos con “F8”, para poder analizar.





The screenshot shows the Win7Reversing application interface. At the top, there is a menu bar with 'File', 'View', 'Debug', 'Registers', 'Disassembly', 'Help', and 'Win7Reversing'. Below the menu bar is a toolbar with various icons for navigation and analysis. The main window displays assembly code with comments. The code is organized into columns: address, disassembly, and comment. The address column shows addresses from 00423FF5 to 00424035. The disassembly column shows instructions like 'cmp', 'ins', 'push', 'lea', 'mov', 'call', 'jz', 'leaq', and 'pushq'. The comment column contains text like 'NO DEBE SALTAR -> BAD BOY', 'CALL 1 PROCESO', 'EAX=EBX -> NO SALTE', and 'SALTA X DEFAULT ::: NO DEBE SALTAR -> BAD BOY'. The code is color-coded, with some lines highlighted in orange and others in red.

Address	Disassembly	Comment
00423FF5	cmp eax,ebx	
00423FF7	ins PRK.00424243	
00423FFD	push ecx	NO DEBE SALTAR -> BAD BOY
00423FFE	lea eax,dword ptr ss:[ebp-0x14]	
00424001	mov ecx,esp	
00424003	mov dword ptr ss:[ebp-0x24],esp	
00424006	push eax	
00424007	call PRK.00401562	
0042400C	mov byte ptr ss:[ebp-0x4],0x6	
00424010	lea eax,dword ptr ss:[ebp-0x40]	
00424013	push eax	
00424014	mov byte ptr ss:[ebp-0x4],0x2	
00424018	mov ecx,edi	
0042401A	call PRK.0040352C	CALL 1 PROCESO
0042401F	cmp eax,ebx	EAX=EBX -> NO SALTE
00424021	jz PRK.00424243	SALTA X DEFAULT ::: NO DEBE SALTAR -> BAD BOY
00424027	lea ecx,dword ptr ss:[ebp-0x40]	
0042402A	call PRK.004023A0	
0042402F	movsq qword ptr ds:[0x5344A8]	
00424035	push ecx	

The screenshot shows a Windows XP desktop with a taskbar containing icons for Internet Explorer, Firefox, and other applications. The main window is titled 'Win7Reversing' and displays assembly code, registers, and a hex dump. The assembly code is for a function named 'NO DEBE SALTAR -> BAD BOY'. The registers section shows various registers like EAX, ECX, EDI, etc. The hex dump shows memory addresses and their corresponding hex values. The application is running on a Windows XP desktop with various icons in the taskbar and a system clock showing 07:25 PM on 21/02/2017.



Sigamos traceando con **"F8"**, llegamos a la direccion de memoria 004240CB donde se encuentra un CMP (comparacion), donde:

EDI=1 y EBX=0, Z=0 -> No salta y nos muestra "BAD BOY"

Pero si :

EDI=1 y EBX=1 -> Z=1 -> Salta y nos muestra "GOOD BOY"

como les comente recordar el registro EBX=0 en la aterior comparacion.

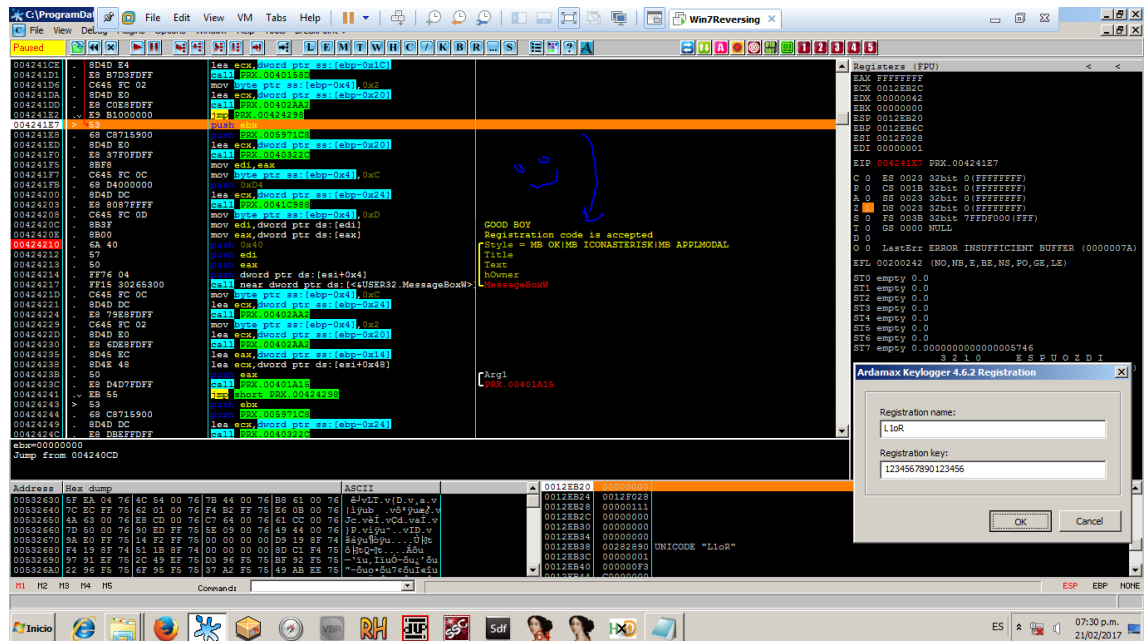
The screenshot shows the Win7Reversing application with the assembly view of a binary. The address 004240CB is highlighted, showing a `cmp edi, ebx` instruction. The registers `EDI` and `EBX` are shown in the right pane. A comment "BAD BOY" is visible in the assembly view. A registration dialog for "Ardamax Keylogger 4.6.2" is open in the foreground, showing the registration name "LtoR" and key "1234567890123456".

Sigamos con el analisis luego le explicare el cambio que se debe realizar, si :

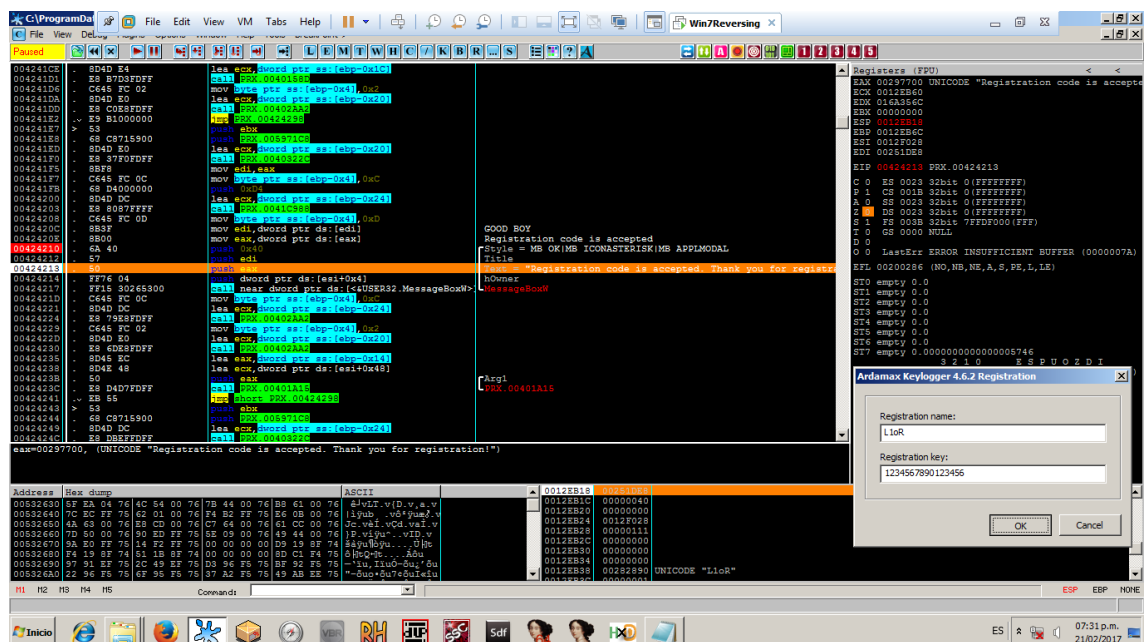
Activamos el Flag Z=1 -> saltaremos al "GOOD BOY", como muestra la carita feliz, realizara un salto a la direccion de memoria 004241E7.

The screenshot shows the Win7Reversing application with the assembly view of a binary. The address 004241E7 is highlighted, showing a `jmp if not taken` instruction. The registers `EDI` and `EBX` are shown in the right pane. A comment "GOOD BOY" is visible in the assembly view. A registration dialog for "Ardamax Keylogger 4.6.2" is open in the foreground, showing the registration name "LtoR" and key "1234567890123456".

Si observamos la flecha azul donde sigue el salto con la direccion de memoria 004241E7, y seguimos traceando con “F8”, llegaremos al “GOOD BOY”.

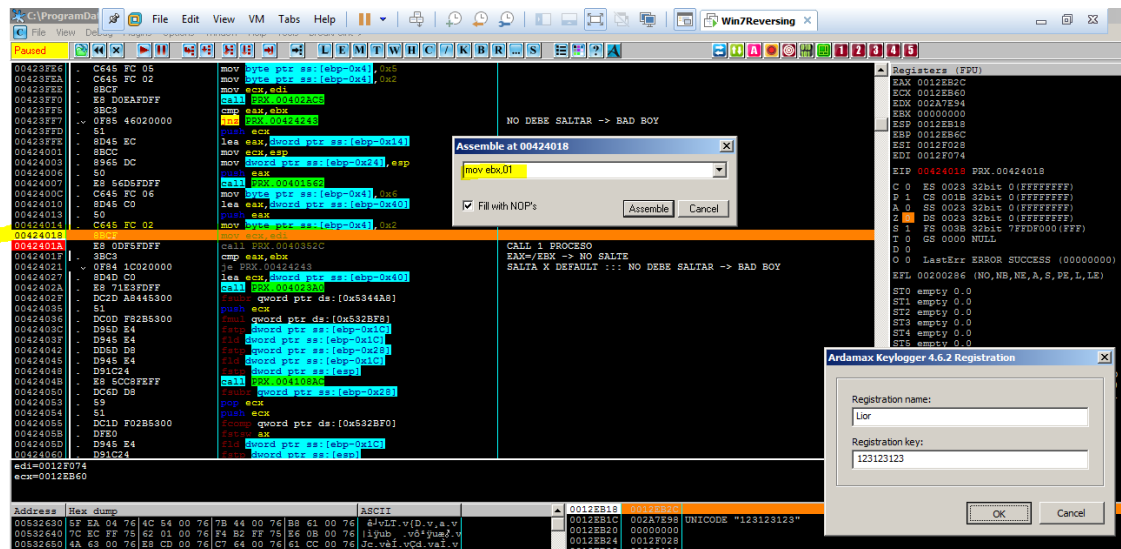


Al final observamos con tracear con “F8”, para poder ver el MessageBoxW de “GOOD BOY”.

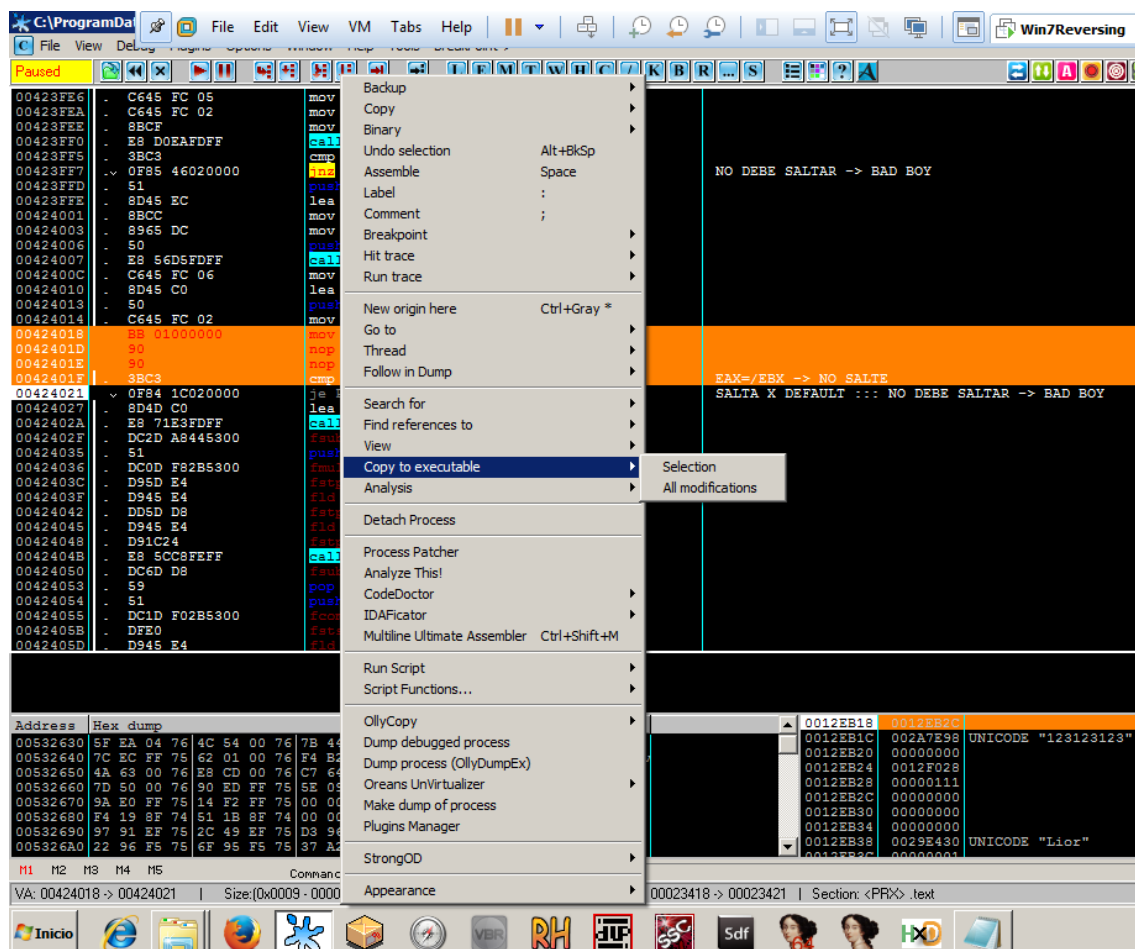


## Parchar ARDAMAX

Lo que debemos hacer es analizar la zona del primer salto, en la direccion de memoria 00424018, modificaremos el codigo por MOV EBX,01, quiere decir:  
EAX=0 - EBX=1 -> Z=0 -> NO SALTA y sigue analizando .



Vamos a guardar la modificacion realizado en la direccion de memoria 00424018.

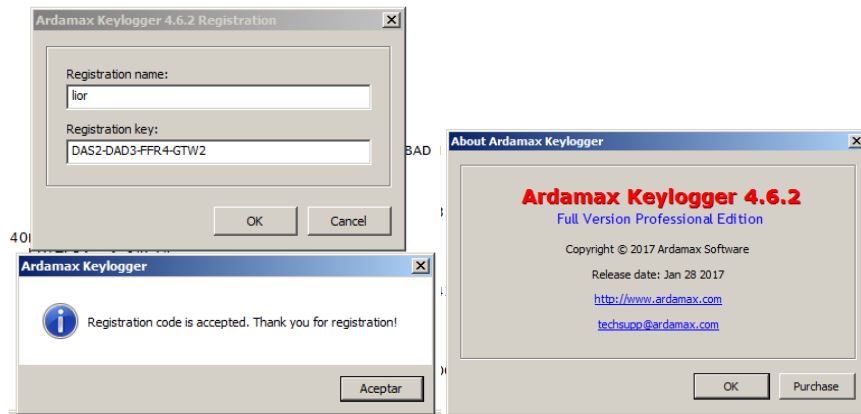


Luego la observacion que se realiza en este caso es que debe cumplir estas reglas para poder registralo, tener una cuenta de Usuario (Lo que desee) y la clave, cualquiera que se le ocurra pero debe tener en cuenta estas reglas : ZZZZ-XXXX-YYYY-XXXX, para poder registrar como version profesional, cada vez que ingreses una clave aleatorio debes tener en cuenta que debe ser alfanumerico y con sus guiones en grupo de 4 digitos.

Por ejemplo yo agrege :

Usuario : lior

Clave : das2-dad3-ffr4-gtw2



Eso seria todo amigos, gracias por su paciencia, espero que le guste, el manual, y si tiene critica, los espero.

Saludos

L1oR