Programa	**Php generator Version 14.10.0.4**
Download	https://www.sqlmaestro.com/products/mysql/phpgenerator/download/
Descripción	Generador de codigo
Herramientas	Olly 1.10, ImporRecontructor, PeiD, PEditor
Dificultad	Newbie
Compresor/Compilador	Borland Delphi 6.0 – 7.0 – Asprotect 1.23 RC4
Protección	Empakado – Opciones deshabilitadas
Objetivos	Desempakarlo – Hacer que corra XDDD y parchear
Cracker	**xcdfgt** Fecha: 21/04/2015
Tutorial nº	1

Introducción

Este es primer tutorial, de los muchos que espero enviar a la comunidad, de la cual he aprendido mucho.
Este tuto está basado en el **277--ASProtect 1.23 RC4_by_+NCR.rar ,**el cual tome como base para desempacar este soft, aquí un repaso de lo que ahí ya se puede encontrar quiero dar especial agradecimiento al autor de este buen documento , así también como al maestro **RICARDO NARVAJA** que siempre ha sido un excelente mentor del cual hice todo su curso Cracking de cero , sin el cual no habría sido posible llegar hasta aca , espero seguir recorriendo el camino del conocimiento.

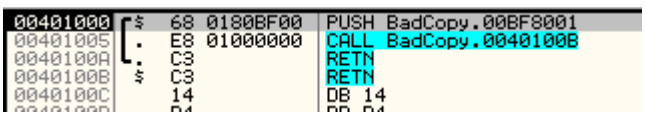
Muchas gracias

Comenzemos

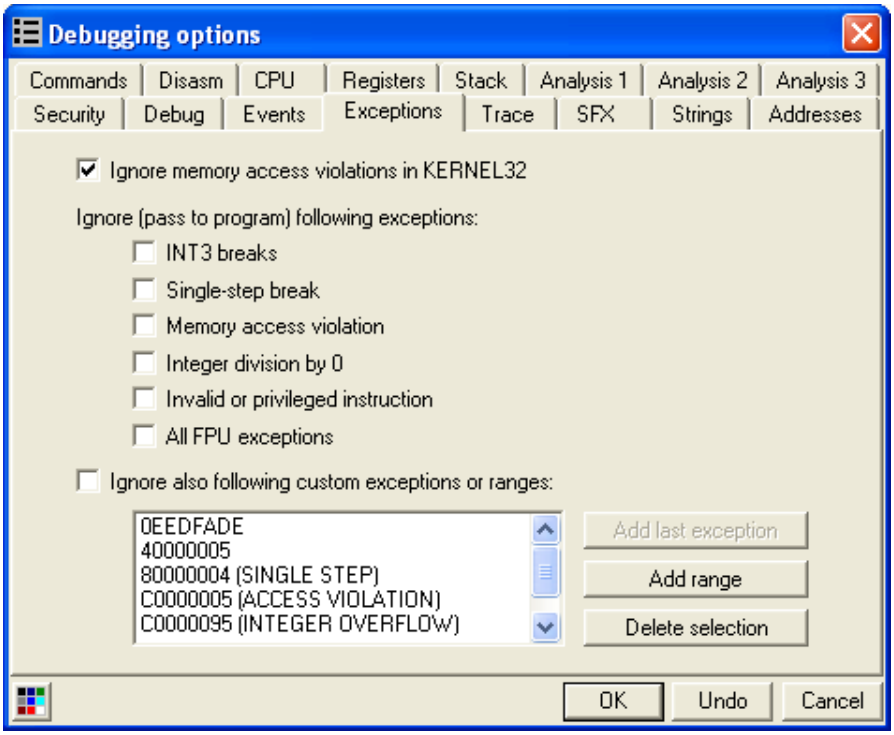
Como siempre, antes de comenzar a trabajar, le pasamos el PeiD a la víctima para ver que es lo que nos trae:

Veamos si es cierto esto, carguémoslo en Olly para tratar de llegar a este mismo OEP.

Ni bien cargamos el programa en Olly, podemos observar esto:



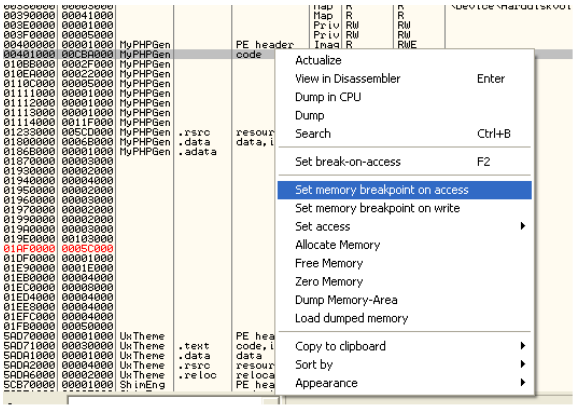
Bueno, ahora con el método de las excepciones llegaremos hasta la última antes de que arranque. Pero antes nos aseguramos de tener todas las tildes puestas en le pestaña de Options – Debugging Options – Exceptions, de esta manera:



Ahora ocultemos Olly para que el programa no lo detecte. Existen dos maneras fáciles de hacer esto, una es con el plugin de IsDebuggerPresent. Lo único que debe hacer es ir al menú Plugins – IsDebugPresent – Hide:

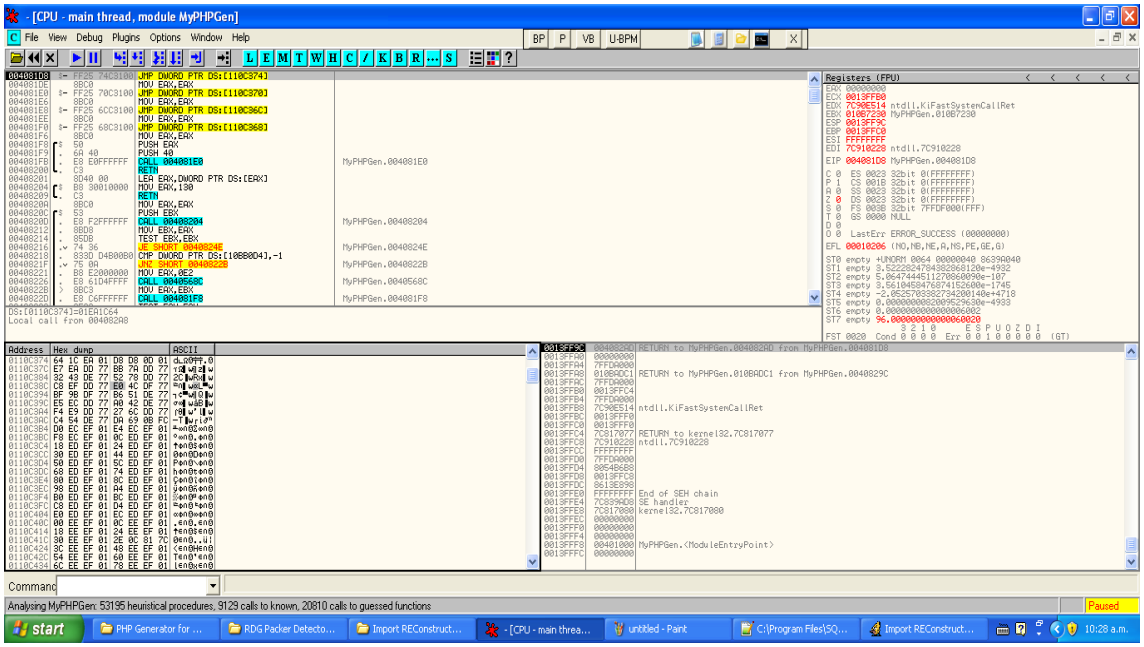
Después de configurar estas opciones, procedo a cargar el programa en el OllyDbg , como voy a usar el método de las excepciones para hallar el OEP, realizo lo siguiente.

Al intentar ejecutar el soft en el entorno del olly, este para en excepciones, las cuales puedo pasar con las teclas **SHIFT + F8** , al llegar a la excepción número 27 , voy a la opción M del olly , y en la sección code del programa coloco un **Set memory breakpoint on access** tal como se muestra en el a imagen abajo



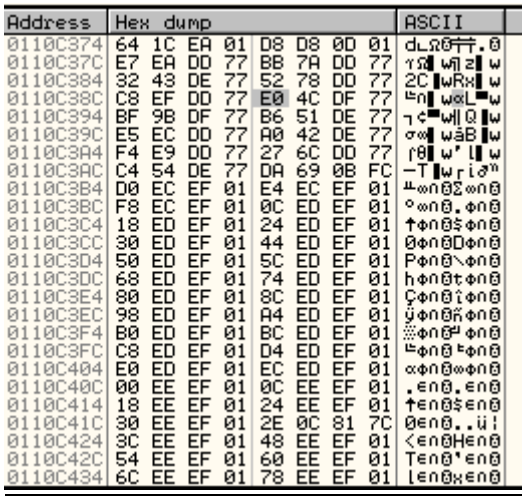
Después de realizar este proceso, debería llegar al OEP , dado esto deberías estar en una zona como esta:

OEP = 004081D8



Después de hacer este proceso , a buscar el inicio y fin de la IAT

Ahora, ya que tenemos el programa parado en el OEP, tratemos de encontrar los datos que hace falta para reparar la tabla de importaciones. Aprovechando también que el OEP es un salto a dicha tabla, hacemos un Follow in Dump – Memory Address y en el dump vemos esto:



Address	Hex dump	ASCII
0110C374	64 1C EA 01 D8 D8 0D 01	d∟Ω0╪╪.0
0110C37C	E7 EA DD 77 BB 7A DD 77	τΩ▌w╗z▌w
0110C384	32 43 DE 77 52 78 DD 77	2C▐wRx▌w
0110C38C	C8 EF DD 77 E0 4C DF 77	╚∩▌wαL▀w
0110C394	BF 9B DF 77 B6 51 DE 77	┐¢▀w╢Q▐w
0110C39C	E5 EC DD 77 A0 42 DE 77	σ∞▌wáB▐w
0110C3A4	F4 E9 DD 77 27 6C DD 77	⌠Θ▌w'l▌w
0110C3AC	C4 54 DE 77 DA 69 0B FC	−T▐wri♂ⁿ
0110C3B4	D0 EC EF 01 E4 EC EF 01	╨∞n0Σ∞n0
0110C3BC	F8 EC EF 01 0C ED EF 01	°∞n0.φn0
0110C3C4	18 ED EF 01 24 ED EF 01	↑φn0$φn0
0110C3CC	30 ED EF 01 44 ED EF 01	0φn0Dφn0
0110C3D4	50 ED EF 01 5C ED EF 01	Pφn0\φn0
0110C3DC	68 ED EF 01 74 ED EF 01	hφn0tφn0
0110C3E4	80 ED EF 01 8C ED EF 01	Çφn0îφn0
0110C3EC	98 ED EF 01 A4 ED EF 01	ÿφn0ñφn0
0110C3F4	B0 ED EF 01 BC ED EF 01	▒φn0╝φn0
0110C3FC	C8 ED EF 01 D4 ED EF 01	╚φn0╘φn0
0110C404	E0 ED EF 01 EC ED EF 01	αφn0∞φn0
0110C40C	00 EE EF 01 0C EE EF 01	.εn0.εn0
0110C414	18 EE EF 01 24 EE EF 01	↑εn0$εn0
0110C41C	30 EE EF 01 2E 0C 81 7C	0εn0..ü|
0110C424	3C EE EF 01 48 EE EF 01	<εn0Hεn0
0110C42C	54 EE EF 01 60 EE EF 01	Tεn0'εn0
0110C434	6C EE EF 01 78 EE EF 01	lεn0xεn0

Esos son los bytes que corresponden al salto. En caso de que no hubiéramos caído en el salto, podríamos haber hecho un Binary String con FF 25 para hallar los saltos.

Vemos que hay valores como 77xxxxxx y 01xxxxxx. Los primeros son direcciones reales de las funciones del Api de Window$. Los demás valores están re direccionados por el Asprotect para convertirlos en entradas malas.

Ahora subimos un poco para ver el inicio de la IAT:

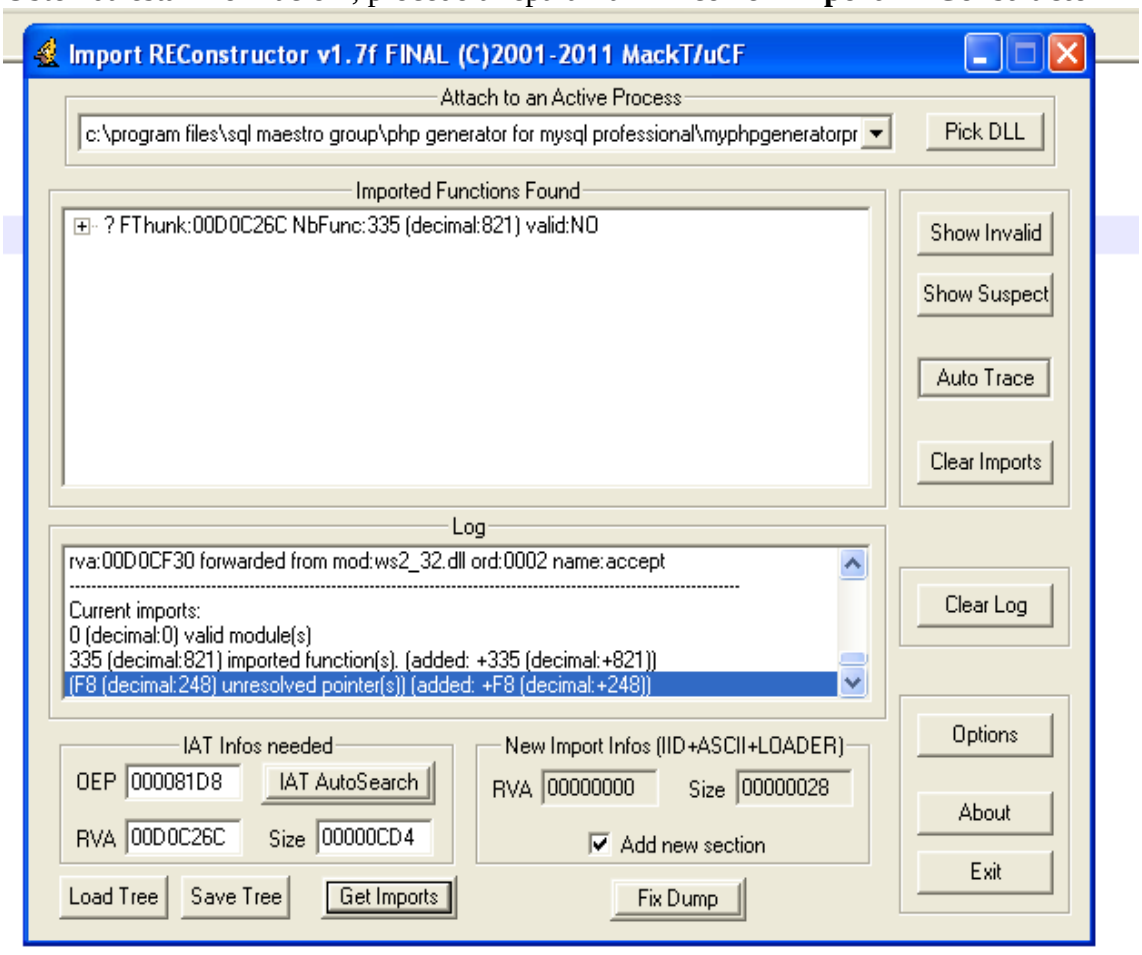
Para buscar el inicio el inicio y fin de la IAT me base en los manuales de Ricardo Después de la búsqueda del IAT, estos son los datos que se arrojan.

004081D8 (Real entry point)

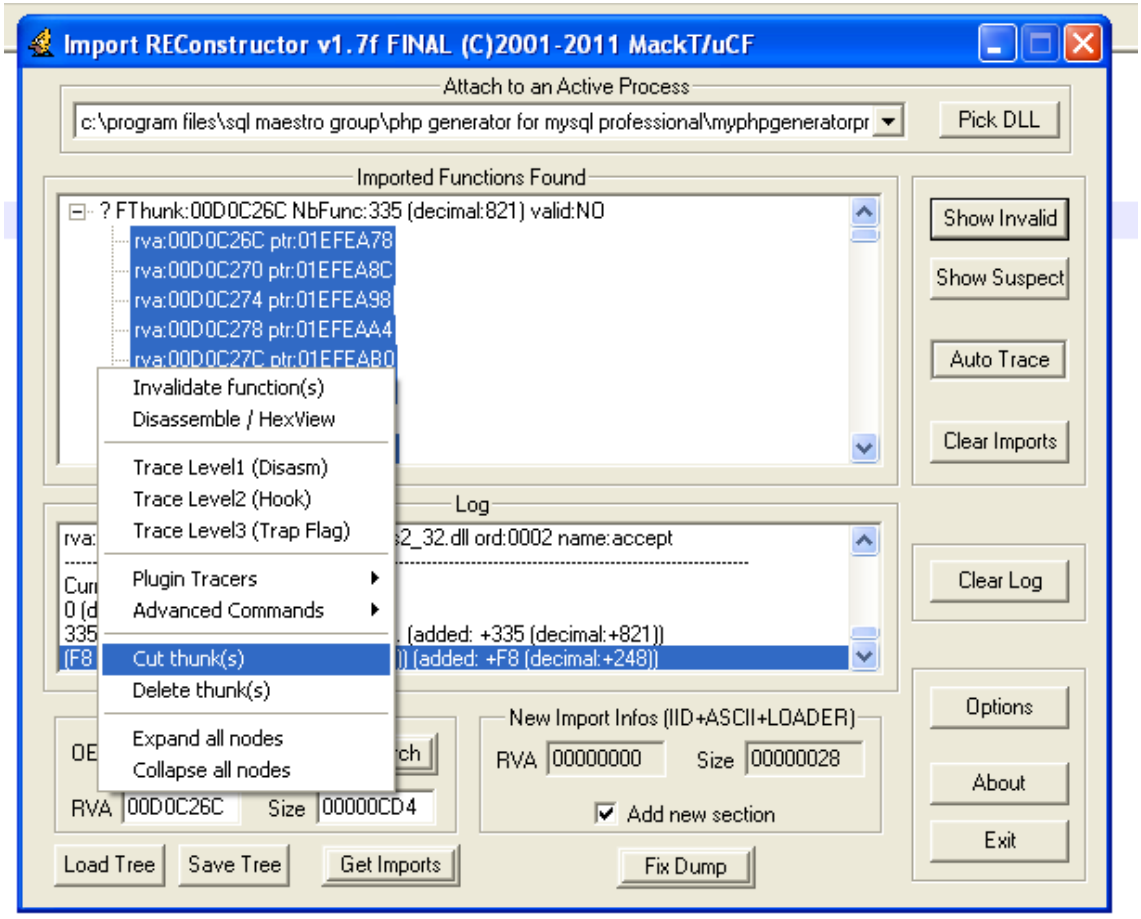
OEP = 004081D8 - 00400000 = 81D8
RVA = 110C26C - 00400000 = D0C26C
Size = 110CF40 - 110C26C = CD4

Inicio de la IAT = 110C26C
Final de la IAT = 110CF40

Obtenida esta información , procedo a reparar la IAT con el **Import REConstructor**

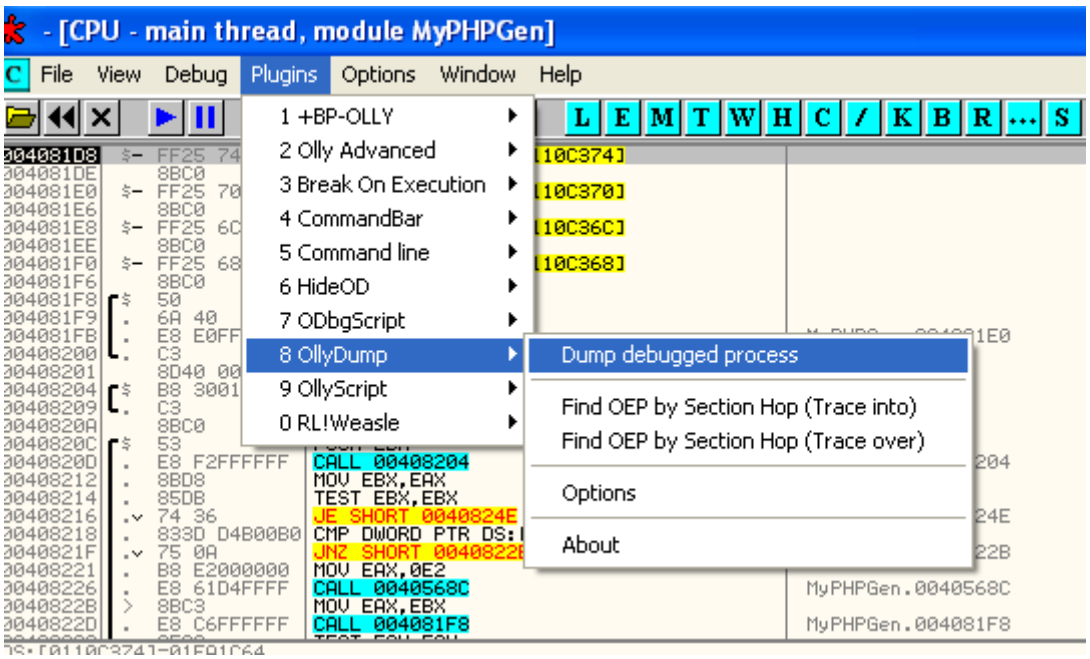


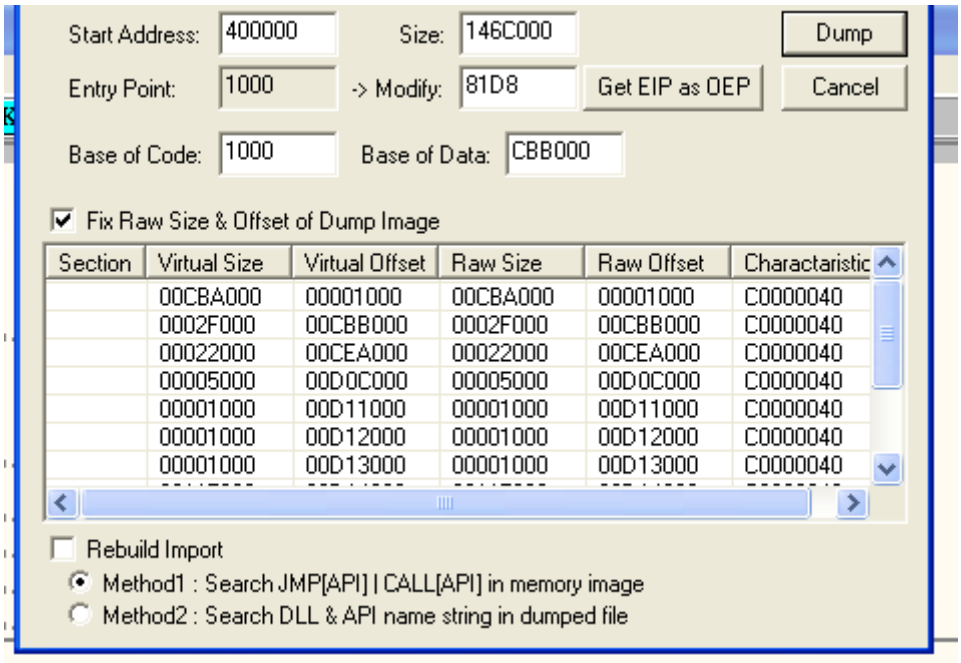
Después de realizar esto , doy a la opción **Show Invalid ,** la cual selecciona todas entradas invalidas , doy click derecho y presiono la opción Cut thunks



Después de realizar esto , doy click a la opción **Fix Dump** , para así reparar la tabla del dumpeado,

Pero antes de realizar este proceso ,debería ya tener mi Dumpeado , el cual obtengo de esta forma





Hay que tener cuidado con quitar la opción **Rebuild import** sino el plugin intentara repara la iat .

Este proceso me crea un archivo el cual le coloco el nombre dump.exe , recodar que a este ejecutable es al cual le debo aplicar el proceso de reparación de la IAT

Arrojado este resultado

Me debería quedar 3 ejecutables

1. MyPHPGeneratorPro.exe (EXE ORIGINAL)
2. dumped_.exe (Dumpeado)
3. dumped.exe (Dumpeado con la iat reparada)

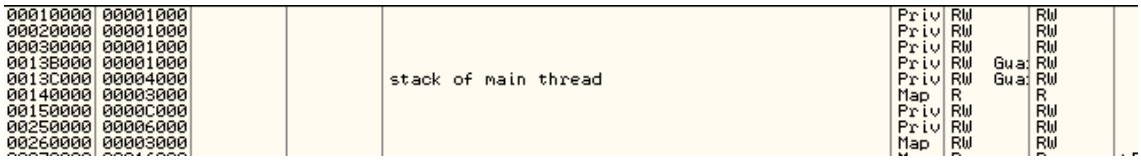
El ejecutable sobre el que vamos a realizar todos los cambios es el que se llama dumped_exe.

Después de realizar este proceso nuestro ejecutable debería quedar totalmente funcional ,listo para ser crackeado.

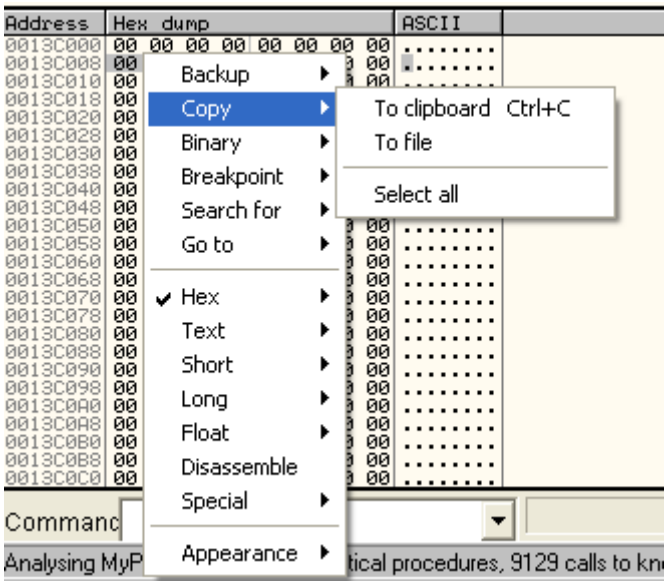
Pero las protecciones de software nos tienen otras sorpresas , y es que algunas secciones del ejecutable no se dumpearon de forma satisfactoria,
Para solucionar esto debemos, agregar las secciones faltantes a nuestro ejecutable dumped_.exe , así como también copiar la pila del ejecutable original a nuestro archivo dumpeado

Copiar la pila del ejecutable original a nuestro ejecutable reparado

Teniendo nuestro OllyDbg parado en OEP , vamos a la sección **M** y buscamos la sección que dice **STACK OF MAIN THREAD,** tal como se muestra en la imagen



Por la imagen vemos que la pila de ejecutable original empieza en 0013C000 y tiene de tamaño 00004000 , con esta información vamos a la sección dumpeado del olly
Y seleccionamos todos los bytes

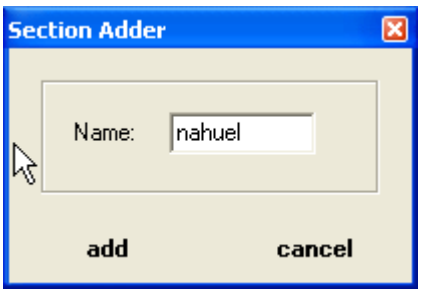


El proceso que debemos realizar consiste en crear una sección donde alojar este código y después crear un injerto en nuestro ejecutable dumpeado , con el cual vamos a cargar la pila y así hacer que tanto nuestro exe dumepado ,y nuestro exe original tengan la misma pila

Para hacer este proceso tomo como referencia el Manual **45-ASProtect 1.23 RC4_by_+NCR.doc** , el cual define el proceso de la siguiente forma

"Bien, retomemos el camino por el cual veníamos. Debíamos agregar una sección de 4000h bytes a nuestro dumpeado con la tabla reparada. Para esto, lo cargamos en Peditor y vamos a Sections. Una vez allí, nos vamos a la última sección del ejecutable (será la que haya agregado el Import para poner la tabla reparada) y le damos a "Add a section":

Le ponemos un nombre cualquiera, en este caso he elegido el mío, je:



Y le damos a “ADD”. Nos queda de esta manera:

	00001000	0077E000	00001000	0077E000	E0000020
	00001000	0077F000	00001000	0077F000	E0000020
	00011000	00780000	00011000	00780000	E0000020
.rsrc	00067000	00791000	00067000	00791000	E0000020
.data	00014000	007F8000	00014000	007F8000	E0000020
.adata	00001000	0080C000	00001000	0080C000	E0000020
.mackt	00003000	0080D000	00003000	0080D000	E0000060
nahuel	00000000	00810000	00000000	00810000	C0000040

Vemos que en “Virtual Size” y “Raw Size” no dice nada todavía, pero ya lo arreglaremos. Recuerden cambiar la características de esta nueva sección agregada a E0000020. Yo sin darme cuenta no lo he hecho y me ha funcionado igual XD (pues creo que sería lo más lógico, pues de esa sección solo le van a leer datos, dado que el injerto para copiar el stack lo haremos en otra parte. Si lo queremos hacer en esta sección, si deberíamos cambiar sus características).

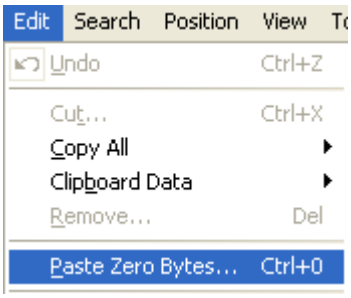
Bien, luego seleccionamos la sección recién agregada y le damos a “Edit Section” y en la ventana que no sale rellenamos los campos que nos interesan con los valores que necesitamos:

Y ahora si nos queda todo correcto:

nahuel	00004000	00810000	00004000	00810000	C0000040

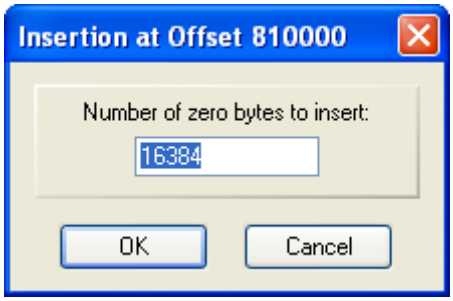
Ahora debemos agregar los bytes al archivo, dado que solo le hemos indicado al dumpeado el tamaño que debe reservar.

Así que abrimos nuestro editor hexadecimal preferido (yo utilizo WinHex;). Nos vamos al último bytes del archivo, lo seleccionamos y luego vamos al menú Edit – Paste Bytes Zero:”



Nos pregunta si queremos pegar los bytes al final del archivo, le decimos que sí y luego completamos con los bytes que nos hacen falta que son 4000h, pero WinHex los pide en decimal así que los pasamos:

Y luego completamos el trabajo:

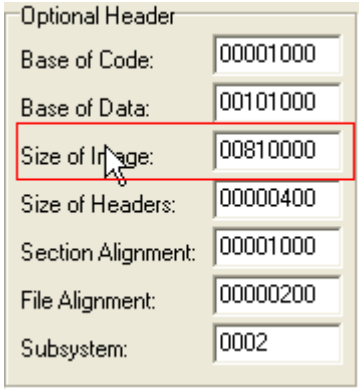


El numero 16384 es el equivalente de 4000 en decimal
Es decir Hex 4000 es igual a 16384 en decimal

Y para estar seguros, miramos la barra de estados de WinHex:

El trabajo está hecho. Luego guardamos los cambios y volvemos a Olly.

Solo falta corregir el Size of Image para compensar los 4000h bytes que agregamos, asi que lo abrimos en el editor PE nuevamente y nos fijamos cual es su Size of Image



A ese valor (810000h) le debemos sumar lo que hemos agregado, o sea 4000h. El resultado es 814000h:

Y lo cambiamos en el Editor PE. Si no hacemos esto, al querer cargar el programa en Olly, nos dará un error.”

Este proceso en resumen , es agregar una zona al ejecutable en la cual vamos a copiar los bytes de la pila , que después con nuestro injerto ejecutaremos

A continuación describo el injerto que hace posible esto. Debemos copiar estos bytes en una zona libre del ejecutable

Injerto de bytes

```
MOV ESI,1F00000
MOV EDI,1EB0000
MOV ECX,1000
REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
MOV DWORD PTR DS:[110C5D0],1F07340
MOV DWORD PTR DS:[110C480],1F06F3C
MOV DWORD PTR DS:[110C450],1F0AEB4
MOV DWORD PTR DS:[401000],401004
MOV DWORD PTR DS:[401004],6F420703
PUSHAD
MOV ESI,186F000
MOV EAX,DWORD PTR FS:[18]
MOV EAX,DWORD PTR DS:[EAX+8]
MOV EDI,EAX
MOV ECX,1000
REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
POPAD
```

XOR EAX,EAX
MOV ECX,13FFB0
MOV EDX,7C90E514
MOV EBX,10B7230
MOV ESP,DWORD PTR FS:[18]
MOV ESP,DWORD PTR DS:[ESP+8]
ADD ESP,3F9C
MOV EBP,DWORD PTR FS:[18]
MOV EBP,DWORD PTR DS:[EBP+8]
ADD EBP,3FC0
MOV ESI,-1
MOV EDI,7C910228
JMP 004081D8

Voy explicar el injerto para que se mas explicito

MOV ESI,1F00000 -> Inicio de la zona creada
MOV EDI,1EB0000 -> zona donde se van copiar los bytes
MOV ECX,1000 -> variable que controla el ciclo de repeticion
REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI] ->
Esta es la instrucción repetitiva que hace el copiado

Estas cuatro primeras líneas se refieren a una zona de memoria que el asprotect me dejo en ceros , para solucionar el problema cree la zona faltante en el ejecutable , y con las estas líneas garantizo que se copien a la zona correcta

MOV DWORD PTR DS:[110C5D0],1F07340
MOV DWORD PTR DS:[110C480],1F06F3C
MOV DWORD PTR DS:[110C450],1F0AEB4
MOV DWORD PTR DS:[401000],401004
MOV DWORD PTR DS:[401004],6F420703

Estas instrucciones las cree , para así direccionar el ejecutable a zonas con código valido , ya que en algunas redirecciones se apuntaba a **00 00 00 00** , tonces como solución al problema opte por agregar la zona faltante en el ejecutable, después de esto ir a la parte del código donde se hacia la redirección a ceros ,y cambiarla por una redirección valida.

Es importante considerar que las direcciones en azul, se calcularon del siguiente forma.

Ejemplo

00408780 $ FF25 D0C51001 JMP DWORD PTR DS:[110C5D0]

Zonas de memoria del exe original

01EFF340 -> Dirección de la instrucción valida
01EFC000 -> Dirección del inicio de la instrucción

3340

Zonas de memoria del exe dumpeado

01F04000 -> Dirección de inicio de la nueva sección
3340 + -> Suma el valor de la resta anterior, para que así la redirección funcione
---
01F07340

En resumen , lo que hace esto es garantizar que nuestra redirección caiga en la misma zona de memoria , del ejecutable original

PUSHAD
MOV ESI,186F000
MOV EAX,DWORD PTR FS:[18] -> **estructura PEB pointer al stack de pila**
MOV EAX,DWORD PTR DS:[EAX+8] ->**Se le suma 8 a la dirección**
MOV EDI,EAX -> **Se copia la dirección real**
MOV ECX,1000
REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]

Estas instrucciones son de copiado típico de pila , pero hare una explicación importante , cuando hice el injerto , queme las direcciones de pila en el código , o lo que se conoce comúnmente como harcodear la dirección , pero por explicación del maestro narvaja, me di cuenta que la dirección de la pila se puede calcular de forma dinámica usando la estructura PEB , que tiene un a puntador que me muestra el valor real de la pila en el momento de ejecución del programa

Esto es de vital importancia para que tu ejecutable funcione por fuera y dentro del olly , pues en mi caso antes de a hacer esto , cuando intentaba correr el soft fuera de un entorno de depuración ,fallaba es decir con doble click.

POPAD **->Recupera registros**
XOR EAX,EAX -> **Establece el registro EAX a cero**
MOV ECX,13FFB0 - >**El valor del registro ,se debe copiar del que tiene el exe original**
MOV EDX,7C90E514 –>**Lo mismo que el anterior**
MOV EBX,10B7230->>**El valor del registro ,se debe copiar del que tiene el exe original**

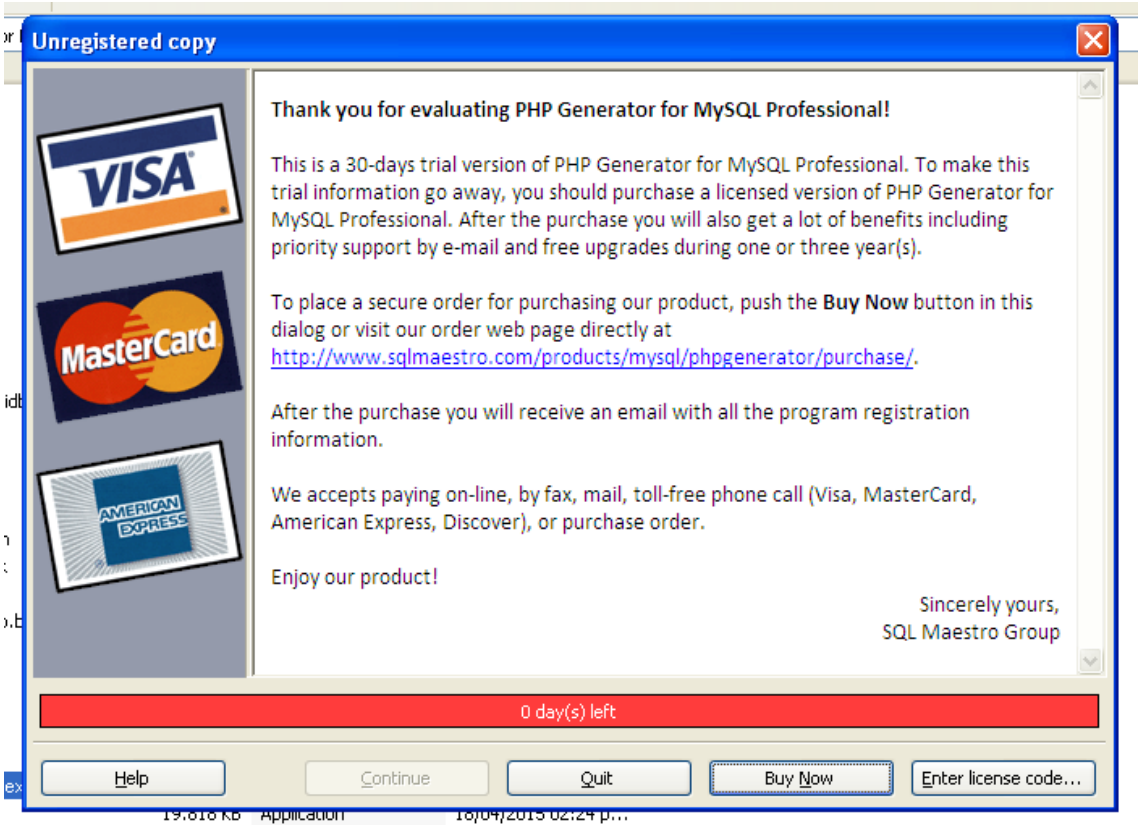
MOV ESP,DWORD PTR FS:[18]
MOV ESP,DWORD PTR DS:[ESP+8]
ADD ESP,3F9C
MOV EBP,DWORD PTR FS:[18]
MOV EBP,DWORD PTR DS:[EBP+8]
ADD EBP,3FC0
MOV ESI,-1
MOV EDI,7C910228

Además de la dirección de inicio de la pila, también hay dos registros de procesador que también se pueden obtener de forma dinámica y son el ESP y el EBP tal como se muestra arriba.

MOV ESI,-1-> El valor del registro, se debe copiar del que tiene el exe original
MOV EDI,7C910228 -> El valor del registro ,se debe copiar del que tiene el exe original
JMP 004081D8 -> Salto al OEP

Con esta explicación, fue la forma en la que logre que el ejecutable funcionara después de ser dumpeado.
Esto lo hice usando mucha información de aquí y de allá , con esto el ejecutable debería quedar listo para ser parcheado .

Como hallar el punto mágico donde el ejecutable decide si esta registrado o no

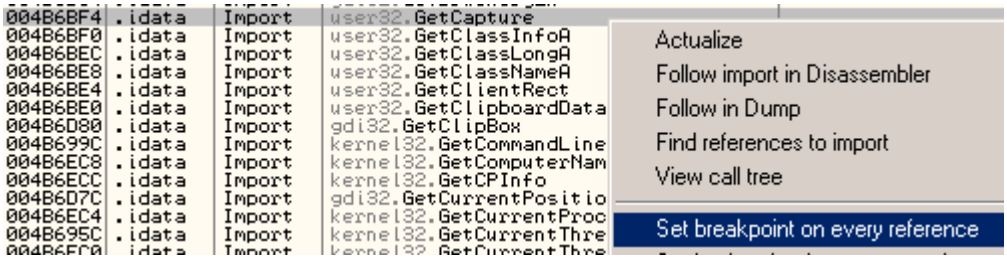


Superada la desempacada del software , solo nos queda vencer este última barrera y podremos usar el soft.

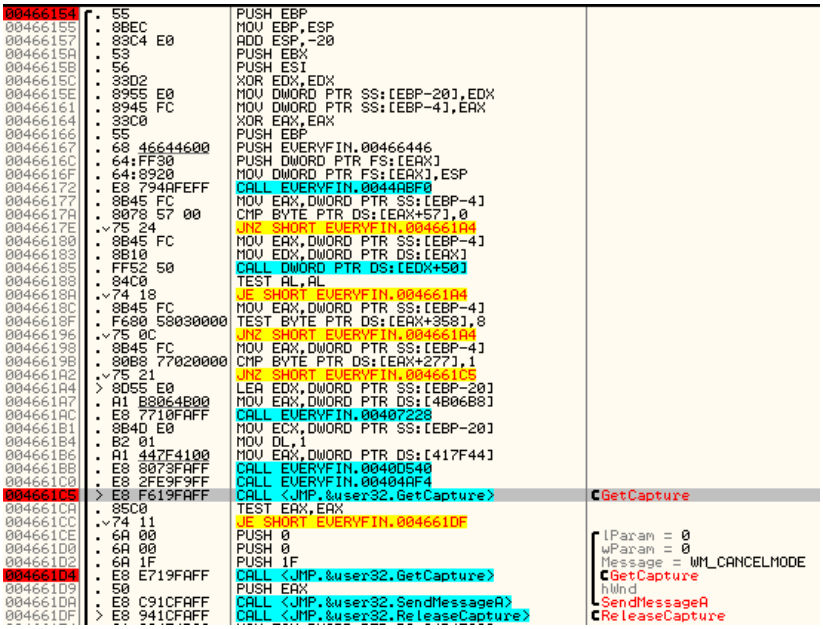
Para encontrar el punto donde el soft valida si esta registrado o no, use el tute llamado

838-COMO_ELIMINAR_NAGS_EN_PROGRAMAS_DELPHI .rar , el cual resumo de la siguiente forma

Después de tener cargado el soft en el olly, damos click en la opción ,Ctrl+N y buscamos a la api GetCapture, y colocamos la opción donde dice **Set breakpoint on every reference**



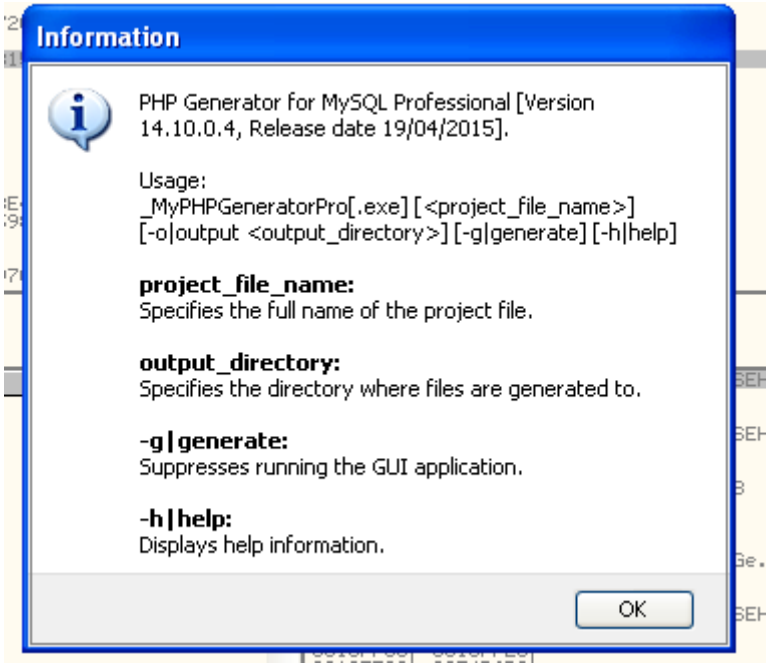
Le ponemos bps a todas las referencias a la api. Luego me fijo cuales son las que cumplen la condición descripta y dejamos esas solamente. Las que se vean como se muestra en la siguiente imagen



Después de encontrar esta zona, colocamos un bp en el inicio del api en **PUSH EBP** esto se hace con el fin se empezar analizar las diferentes direcciones y call que el soft hace , esto hasta encontrar un salto condicional que nos determine cuando esta registrado y cuando no , realizado esto llegamos a un punto como el que se muestra en la imagen abajo.

```
00F85A9D  .  53            PUSH EBX                                 Arg2
00F85A9E  .  8B5D 08       MOV EBX,DWORD PTR SS:[EBP+8]             Arg1
00F85AA1  .  53            PUSH EBX                                 _MyPHPGe.00F856E0
00F85AA2  .  E8 39FCFFFF   CALL 00F856E0
00F85AA7  .  A1 84580E01   MOV EAX,DWORD PTR DS:[10E5884]
00F85AAC  .  8B00          MOV EAX,DWORD PTR DS:[EAX]
00F85AAE  .  E8 657371FF   CALL 0069CE18                            _MyPHPGe.0069CE18
00F85AB3  .  33C0          XOR EAX,EAX
00F85AB5  .  55            PUSH EBP
00F85AB6  .  68 A55BF800   PUSH 0F85BA5
00F85ABB  .  64:FF30       PUSH DWORD PTR FS:[EAX]
00F85ABE  .  64:8920       MOV DWORD PTR FS:[EAX],ESP
00F85AC1  .  E8 5AFCFFFF   CALL 00F85720                            _MyPHPGe.00F85720
00F85AC6  .  84C0          TEST AL,AL
00F85AC8  .v 75 4B         JNZ SHORT 00F85B15                       _MyPHPGe.00F85B15
00F85ACA  .  33C0          XOR EAX,EAX
00F85ACC  .  55            PUSH EBP
00F85ACD  .  68 0E5BF800   PUSH 0F85B0E
00F85AD2  .  64:FF30       PUSH DWORD PTR FS:[EAX]
00F85AD5  .  64:8920       MOV DWORD PTR FS:[EAX],ESP
00F85AD8  .  A1 58820E01   MOV EAX,DWORD PTR DS:[10E8258]
00F85ADD  .  8B00          MOV EAX,DWORD PTR DS:[EAX]
00F85ADF  .  E8 006E51FF   CALL 0049C8E4                            _MyPHPGe.0049C8E4
00F85AE4  .  FF15 A4300E0  CALL DWORD PTR DS:[10E30A4]              _MyPHPGe.00F85598
00F85AEA  .  A1 58820E01   MOV EAX,DWORD PTR DS:[10E8258]
00F85AEF  .  8B00          MOV EAX,DWORD PTR DS:[EAX]
00F85AF1  .  E8 866E51FF   CALL 0049C97C                            _MyPHPGe.0049C97C
```

Allí podemos ver , que hay un salto tipo JNZ , para probar cambiamos el bit **z** del procesador a 0 y damos F9, hecho esto ,el programa nos lleva a la siguiente pantalla

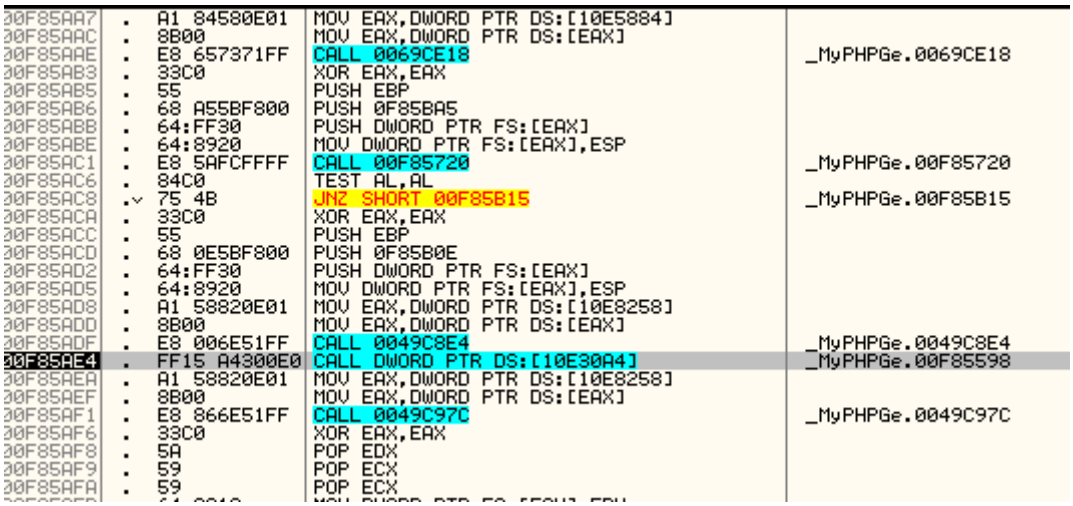


Pudo haber sido la solución jajja pero vemos que nos dimos cuenta, que esta zona puede estar la validación.

Por lo que he leído algunos soft son fáciles de crackear , solo consiste en cambiar algunos saltos, incluso uno y listo.

Sin embargo analizando y leyendo el código , pudimos enterarnos que algunos tipos de registro se basan en cambiar redirecciones de código a zonas donde aparece la NAG que nos impida usar el programa , lo que tenemos que hacer en estos casos es cambiar la redirección de memoria a la zona valida y llenar algunos datos de licencia y demás para que así , el programa piense que ha sido registrado , guardar los cambios y wala.

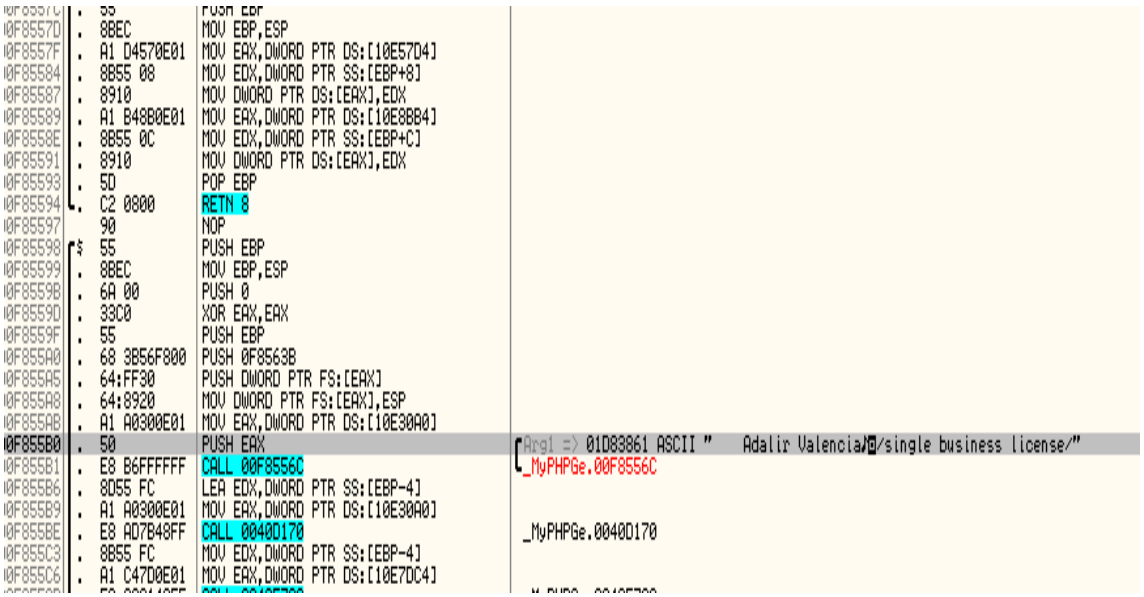
Lo que hicimos en este caso fue lo siguiente:



El salto que vemos aquí, es una redirección de memoria , fui al contenido de la dirección y la cambiamos a la dirección de la zona valida.

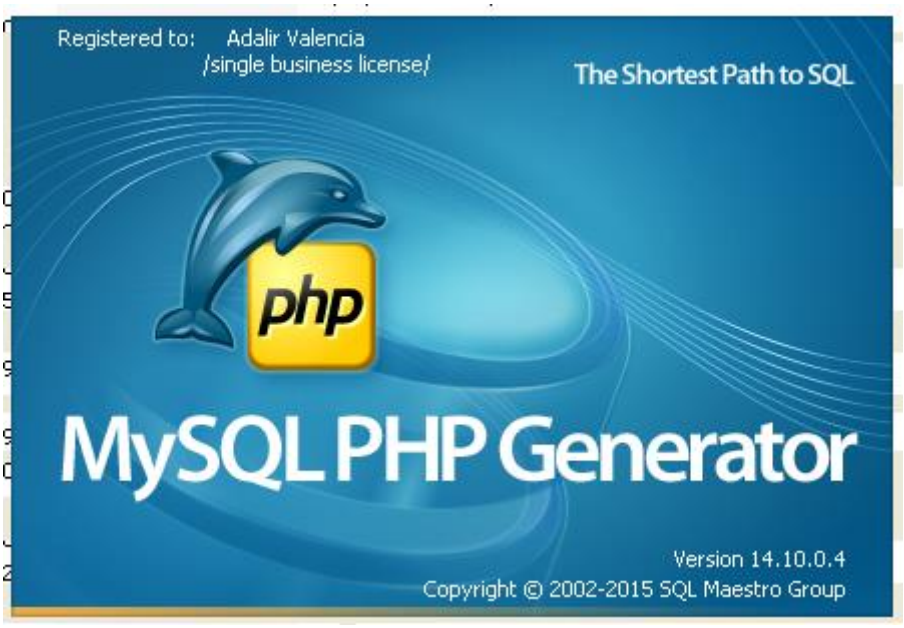
La dirección 10E30A4 apuntaba a la direccion **00F85548** .
Lo que hice fue , hacer que el call no llamara a esta dirección sino a la siguiente **00F85598**

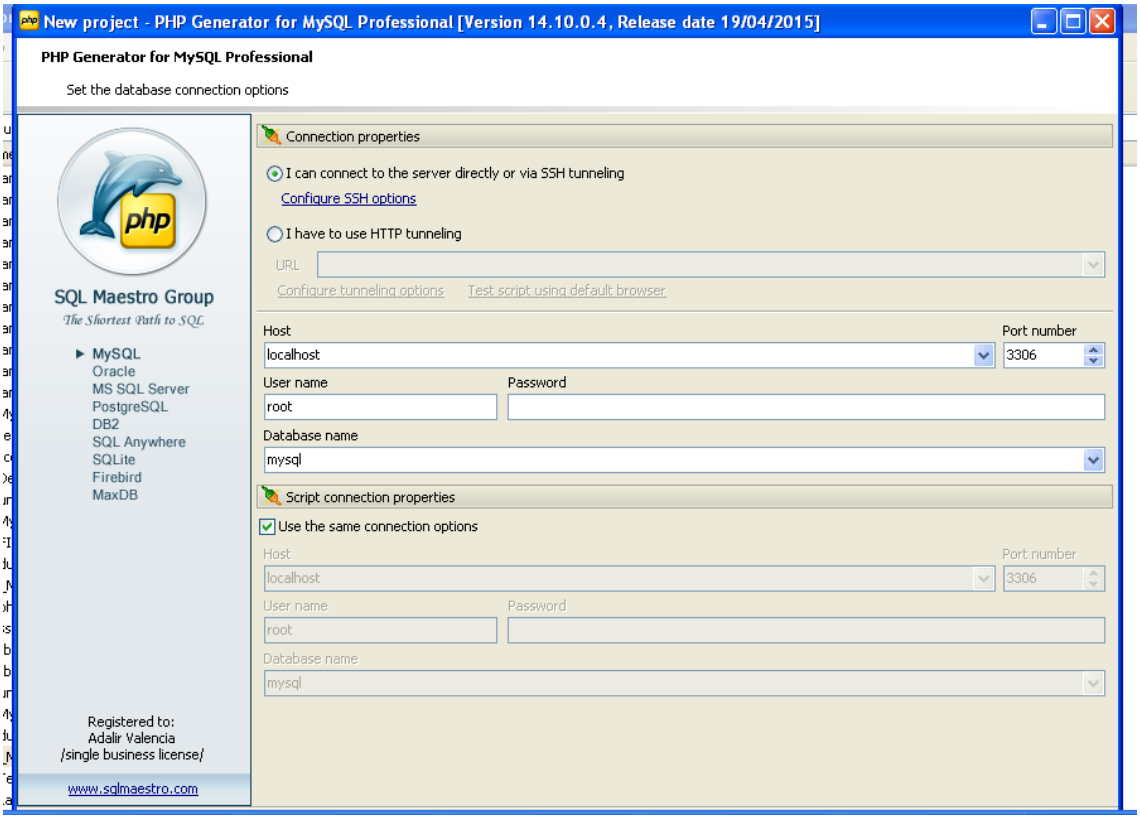
Hecho esto también analizamos que en la nueva dirección también se espera un dato que es donde se guardan los datos de registro de licencia



Es decir fuimos la dirección **01D83861** y la llenamos con los datos que arriba vemos , podría ser cualquier nombre , lo importante es que el código espera estos datos, pues sino los recibe, genera una excepción de acceso a memoria

Hecho esto nuestro programa nos dará la siguiente pantalla





Y listo nuestro software ha sido vencido

Hasta la próxima

xcdfgt

Abril de 2015