ADVERTENCIA:

Al abrir este documento, el lector acepta incondicionalmente su total y exclusiva responsabilidad legal, de

acuerdo a las leyes vigentes en su país, por el uso de las técnicas experimentales, educativas y/o de investigación

aquí vertidas en materia de programación especializada de computadoras. En caso de discrepar con alguno de

los puntos descriptos, deberá eliminar inmediatamente el presente documento y todo material asociado al

mismo.

Este tutorial es solo a los fines didáctico y como conocimiento de la informática en general, el autor de esta obra

NO se hace responsable del uso indebido e ilegal de los conocimientos aquí explicados.

PROGRAMA	FUNMORPH
PROTECCION	NINGUNA- 7 DIAS DE USO
HERRAMIENTAS	OLLYDBG
COMPILADOR	C++
DIFICULTAD	MINIMA
CRACKER	GUILLE3000

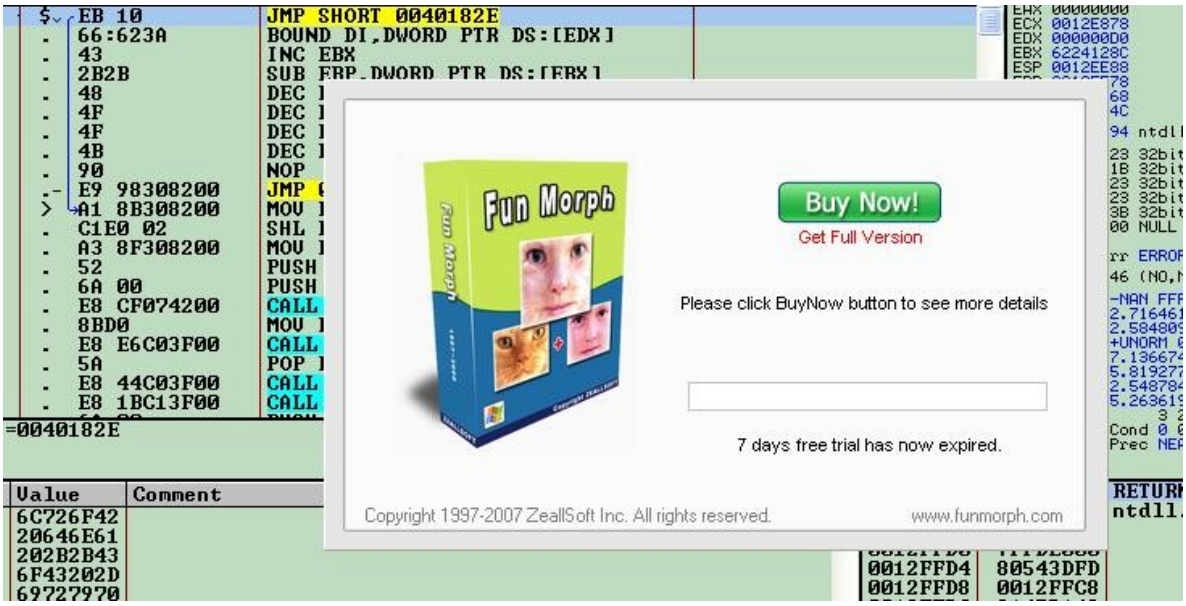
Introduccion

No hay mucho para decir.. este programa no tiene packer ni nada, es código puro y no podemos ingresar serial porque solo es una demo por 7 dias.. pero el programa esta full

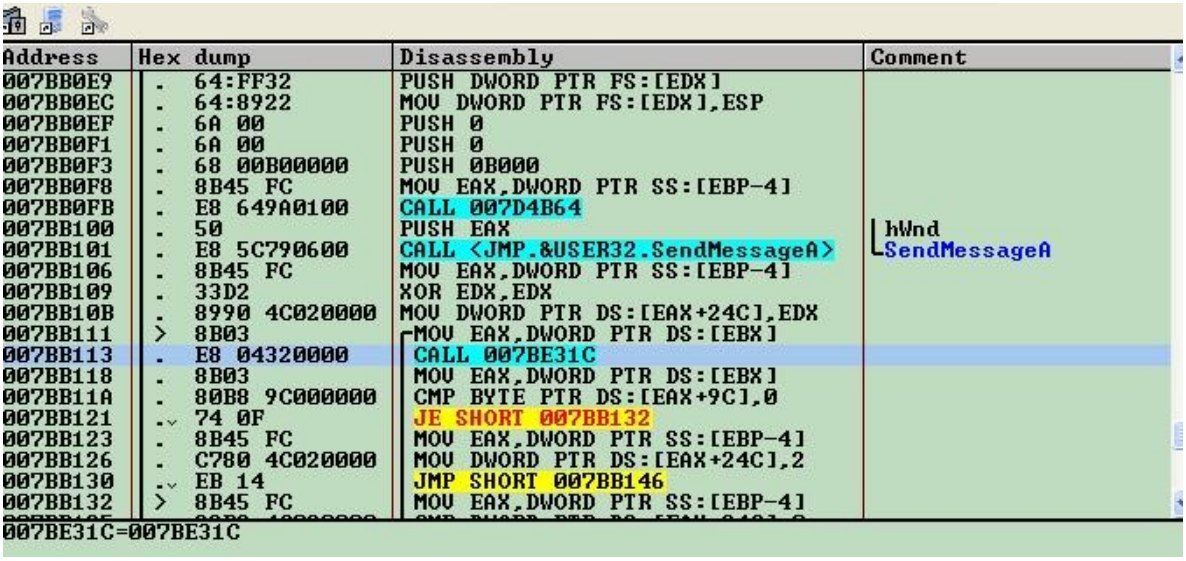
Asi que una vez que lo instalamos adelantamos el reloj para que venzan los 7 dias. Para hacer más rápido le adelanté el año del reloj de Windows a 2013

Empecemos:

Nos aparece una nag que nos avisa que el programa a expirado.

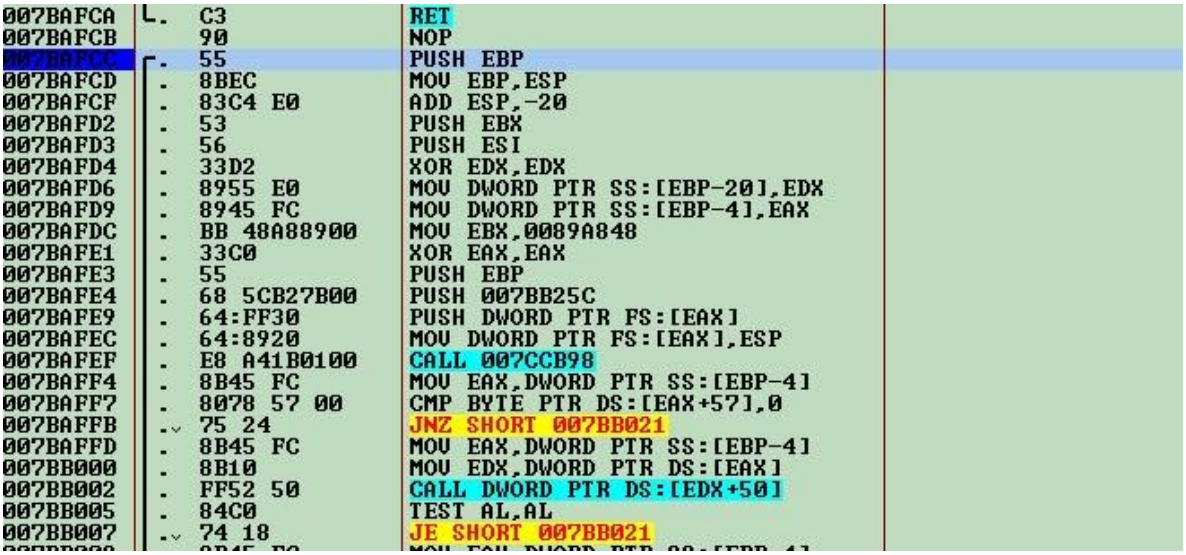


Asi que presionamos pausa en el olly y vamos a ver el árbol de llamadas (K) (call stack), buscamos la ultima línea y le hacemos doble clic, así que caemos aquí



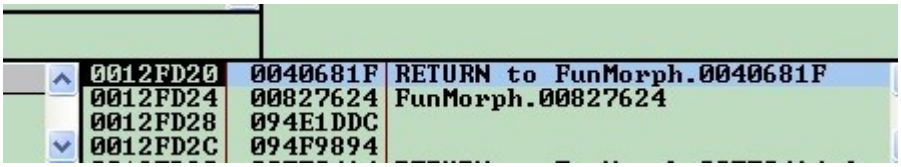
```
Address   Hex dump          Disassembly                          Comment
007BB0E9  .  64:FF32        PUSH DWORD PTR FS:[EDX]
007BB0EC  .  64:8922        MOV DWORD PTR FS:[EDX],ESP
007BB0EF  .  6A 00          PUSH 0
007BB0F1  .  6A 00          PUSH 0
007BB0F3  .  68 00B00000    PUSH 0B000
007BB0F8  .  8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
007BB0FB  .  E8 649A0100    CALL 007D4B64
007BB100  .  50             PUSH EAX                             hWnd
007BB101  .  E8 5C790600    CALL <JMP.&USER32.SendMessageA>      SendMessageA
007BB106  .  8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
007BB109  .  33D2           XOR EDX,EDX
007BB10B  .  8990 4C020000  MOV DWORD PTR DS:[EAX+24C],EDX
007BB111  >  8B03           MOV EAX,DWORD PTR DS:[EBX]
007BB113  .  E8 04320000    CALL 007BE31C
007BB118  .  8B03           MOV EAX,DWORD PTR DS:[EBX]
007BB11A  .  80B8 9C000000  CMP BYTE PTR DS:[EAX+9C],0
007BB121  .  74 0F          JE SHORT 007BB132
007BB123  .  8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
007BB126  .  C780 4C020000  MOV DWORD PTR DS:[EAX+24C],2
007BB130  .  EB 14          JMP SHORT 007BB146
007BB132  >  8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
007BE31C=007BE31C
```

. Como no vemos ningún salto condicional inmediato, subimos hasta el principio de la rutina y ponemos un bp para que pare



```
007BAFCA  L.  C3            RET
007BAFCB      90            NOP
007BAFCC  r.  55            PUSH EBP
007BAFCD  .   8BEC          MOV EBP,ESP
007BAFCF  .   83C4 E0       ADD ESP,-20
007BAFD2  .   53            PUSH EBX
007BAFD3  .   56            PUSH ESI
007BAFD4  .   33D2          XOR EDX,EDX
007BAFD6  .   8955 E0       MOV DWORD PTR SS:[EBP-20],EDX
007BAFD9  .   8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
007BAFDC  .   BB 48A88900   MOV EBX,0089A848
007BAFE1  .   33C0          XOR EAX,EAX
007BAFE3  .   55            PUSH EBP
007BAFE4  .   68 5CB27B00   PUSH 007BB25C
007BAFE9  .   64:FF30       PUSH DWORD PTR FS:[EAX]
007BAFEC  .   64:8920       MOV DWORD PTR FS:[EAX],ESP
007BAFEF  .   E8 A41B0100   CALL 007CCB98
007BAFF4  .   8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
007BAFF7  .   8078 57 00    CMP BYTE PTR DS:[EAX+57],0
007BAFFB  .   75 24         JNZ SHORT 007BB021
007BAFFD  .   8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
007BB000  .   8B10          MOV EDX,DWORD PTR DS:[EAX]
007BB002  .   FF52 50       CALL DWORD PTR DS:[EDX+50]
007BB005  .   84C0          TEST AL,AL
007BB007  .   74 18         JE SHORT 007BB021
```

Cuando para vemos en la pila de donde fue llamada esta rutina



```
0012FD20  0040681F  RETURN to FunMorph.0040681F
0012FD24  00827624  FunMorph.00827624
0012FD28  094E1DDC
0012FD2C  094F9894
```

Nos dirigimos allí, asi que hacemos clic derecho y seguir en el desensamblador

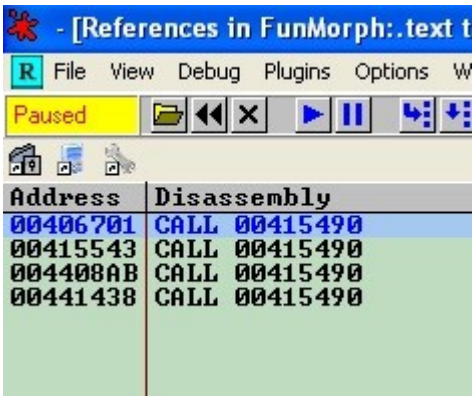
Subimos y vemos que después de un call realiza una comparación bastante interesante

```
004066E9  .  8D85 40FFFFFF  LEA EAX,DWORD PTR SS:[EBP-C0]
004066EF  .  BA 02000000    MOV EDX,2
004066F4  .  E8 D3A04100    CALL 008207CC
004066F9  .  8B96 98050000  MOV EDX,DWORD PTR DS:[ESI+598]
004066FF  .  8BC6           MOV EAX,ESI
00406701  .  E8 8AED0000    CALL 00415490
00406706  .  84C0           TEST AL,AL
00406708  .v 74 7B          JE SHORT 00406785
0040670A  .  66:C785 D8FEFF MOV WORD PTR SS:[EBP-128],104
00406713  .  BA D4378200    MOV EDX,008237D4          ASCII "Registered"
00406718  .  8D85 30FFFFFF  LEA EAX,DWORD PTR SS:[EBP-D0]
0040671E  .  E8 C99E4100    CALL 008205EC
00406723  .  FF85 E4FEFFFF  INC DWORD PTR SS:[EBP-11C]
00406729  .  8B10           MOV EDX,DWORD PTR DS:[EAX]
0040672B  .  8B86 BC040000  MOV EAX,DWORD PTR DS:[ESI+4BC]
00406731  .  E8 5A7C3C00    CALL 007CE390
00406736  .  FF8D E4FEFFFF  DEC DWORD PTR SS:[EBP-11C]
0040673C  .  8D85 30FFFFFF  LEA EAX,DWORD PTR SS:[EBP-D0]
00406742  .  BA 02000000    MOV EDX,2
00406747  .  E8 80A04100    CALL 008207CC
0040674C  .  BA 78000000    MOV EDX,78
00406751  .  8B86 34050000  MOV EAX,DWORD PTR DS:[ESI+534]
00406757  .  E8 70733C00    CALL 007CDACC
```

Je salta si son iguales y como AL va a valer 0 saltará .

O sea que analizando nos damos cuenta que necesitamos que AL a la salida del call valga 1 para que no salte y estemos registrados. Pero también tenemos que tener en cuenta que puede haber un doble chequeo (o quizás mas) y que nos lleve a no estar registrado y a mostrarnos la Nag del principio.

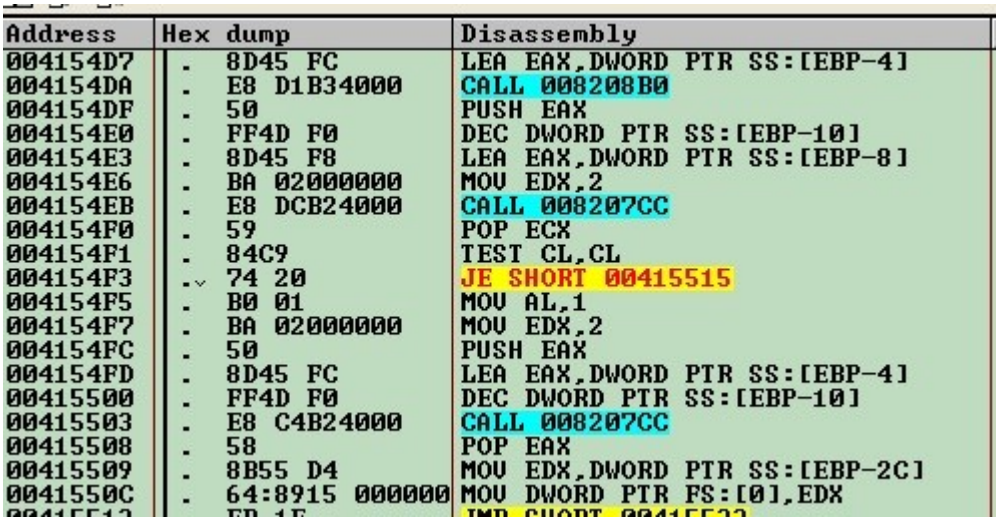
Frente a esta posibilidad y al ver que la salida del call 415490 es el que setea a AL, le hacemos clic botón derecho y find references call destination. Para ver si es llamado desde otro lugar



Como suponíamos, es llamado desde 3 lugares más.

Asi que pensemos.. si parcheamos desde dentro de este call, automáticamente quedarán parcheados los otros 3 resultados de AL.

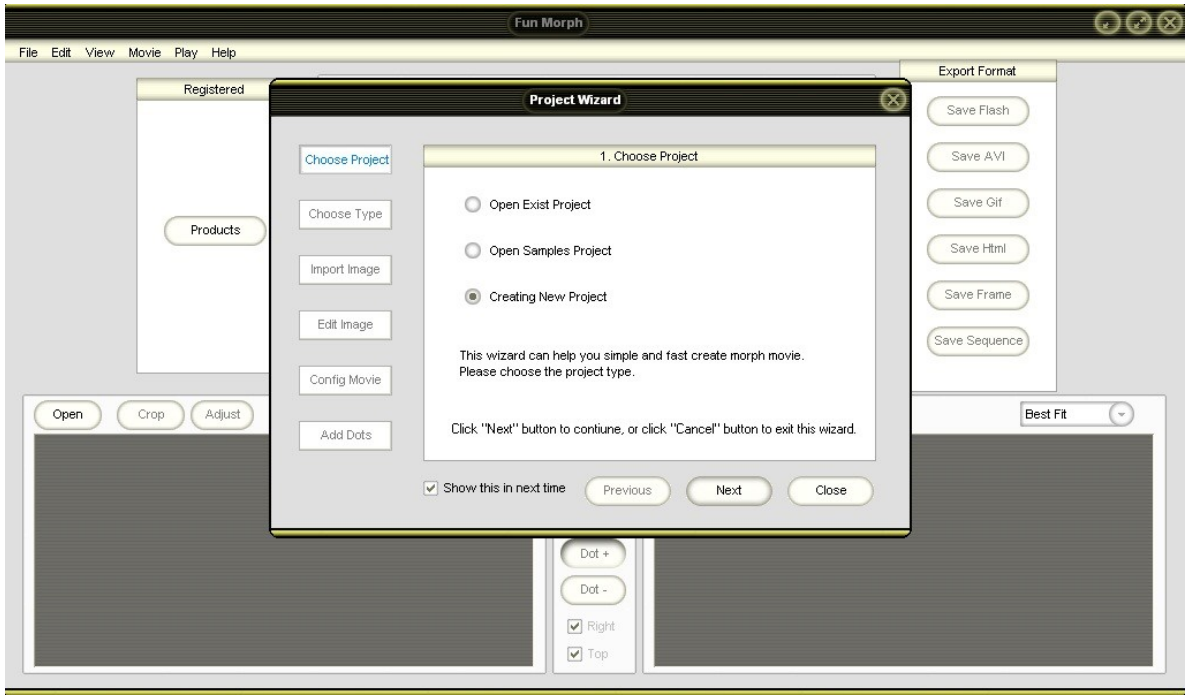
Asi que entremos dentro de este call con enter (o poniendo un bp y traceando) para estudiarlo por dentro.



Address	Hex dump	Disassembly
004154D7	. 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]
004154DA	. E8 D1B34000	CALL 008208B0
004154DF	. 50	PUSH EAX
004154E0	. FF4D F0	DEC DWORD PTR SS:[EBP-10]
004154E3	. 8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
004154E6	. BA 02000000	MOV EDX,2
004154EB	. E8 DCB24000	CALL 008207CC
004154F0	. 59	POP ECX
004154F1	. 84C9	TEST CL,CL
004154F3	. 74 20	JE SHORT 00415515
004154F5	. B0 01	MOV AL,1
004154F7	. BA 02000000	MOV EDX,2
004154FC	. 50	PUSH EAX
004154FD	. 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]
00415500	. FF4D F0	DEC DWORD PTR SS:[EBP-10]
00415503	. E8 C4B24000	CALL 008207CC
00415508	. 58	POP EAX
00415509	. 8B55 D4	MOV EDX,DWORD PTR SS:[EBP-2C]
0041550C	. 64:8915 000000	MOV DWORD PTR FS:[0],EDX

Vemos en la imagen un mov al,1 debajo de un salto condicional. Si traceamos vamos a ver que saltará sobre este mov porque no estamos registrados. Asi que lo nopeamos a Je para que el programa fluya sobre mov al,1

Guardamos el cambio que hemos hecho al ejecutable y lo reiniciamos, le damos a Run y el programa quedó registrado y funcionando a la perfeccion.



Agradecimientos:

A toda la Lista de CrackLatinos que siempre me ayudan cuando tengo una duda y a todos los que leen este tute.