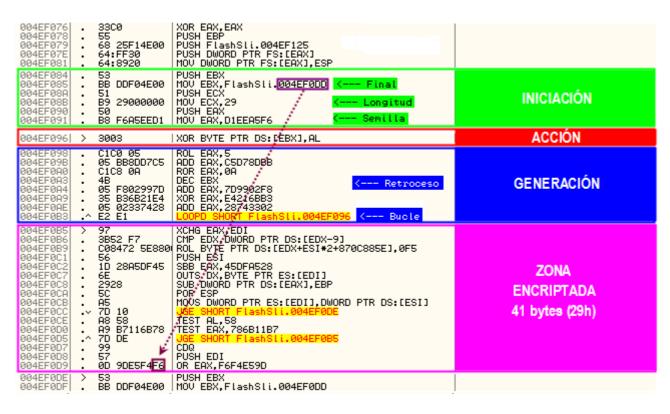
## Desencriptado en tiempo de ejecución

## Introducción

Hace algunos días me topé por causalidad con una versión antigua del programa FlashSlider (v 0.9) y tuve dificultades en llegar a la zona donde se analizaba el serial. Descubrí con sorpresa que las zonas nobles del programa eran desencriptadas en tiempo de ejecución. El mecanismo es sencillo y traté de aprovecharlo.

## El Análisis

Mediante un *memory breakpoint on write* en la sección CODE es fácil encontrar la zonas donde el programa reescribe sobre su código antes y después de ejecutarlo. Todas tienen un aspecto similar:



Se distinguen cuatro zonas (vamos a prescindir de otras cuestiones como la alteración del SEH):

- 1ª Zona INICIACIÓN: aquí se empujan a la pila los registros que van a ser utilizados en el proceso (preservación de registros). Se define (a) donde se va a realizar la (des)encriptación, EBX apunta al final de la zona encriptada que será recorrida hacia atrás; (b) la longitud en bytes de la zona a (des)encriptar, ECX hace su función contadora habitual; (c) una semilla inicial para el proceso en EAX.
- 2ª Zona ACCIÓN: aquí se realiza la alteración del código byte a byte. Contiene una sola instrucción que puede ser:
  - XOR BYTE PTR DS:[EBX],AL
  - ADD BYTE PTR DS:[EBX],AL
  - SUB BYTE PTR DS:[EBX],AL

Debemos tener en cuenta que debe ser reversible (tener inversa).

3ª Zona GENERACIÓN: Se realizan operaciones lógicas, y/o aritméticas para cambiar la semilla de una manera rápida, reproducible pero difícilmente reconocible. También se decremento el puntero EBX para

pasar al siguiente byte a procesar. Acaba inevitablemente con la instrucción LOOPD que cierra el bucle.

4ª Zona CÓDIGO ENCRIPTADO: Contiene los bytes encriptadas a procesar. Puede incluir una parte con basura inicial para despistar al desensamblador, aún después del desencriptado.

Después de salir del bucle se obtiene:

```
33C0
55
68 25F14E00
64:FF30
64:8920
53
BB DDF04E00
  004EF076
004EF078
004EF079
004EF07E
                                                                                                                                                                                                                               XOR EAX,EAX
PUSH EBP
PUSH FlashSli.004EF125
PUSH DWORD PTR FS:[EAX]
MOU DWORD PTR FS:[EAX],ESP
    004EF081
                                                                                                                                                                                                                                 MOV DWORD FIR FS:[EHX],ES
PUSH EEX
MOV EBX,FlashSli.004EF0DD
PUSH ECX
MOV ECX, 29
PUSH EAX
    004EF084
004EF085
                                                                                               51
B9 29000000
50
B8 F6ASEED1
3003
C1C0 05
     004EF08F
    004EF090
                                                                                                                                                                                                                               PUSH EAX
MOV EAX,D1EEA5F6
XOR BYTE PTR DS:[EBX],AL
ROL EAX,5
ADD EAX,C5D78DBB
ROR EAX,0A
DEC EBX
ADD EAX,7D9902F8
XOR EAX,E4216BB3
ADD EAX,28743302
LOOPD SHORT FlashSli,0048
    004EF098
004EF09E
                                                                                              C1C0 05
05 BB8DD7C5
C1C8 0A
4B
05 F802997D
35 B36B21E4
05 02337428
E2 E1
     004EF0A0
     004EF0A4
    004EF0A9
004EF0AE
                                                                                                                                                                                                                                 POP EAX
POP ECX
POP ECX
POP EBX
JMP SHORT FlashSli @@4EF@CA
TEST BYTE PTR DS: [EDI+EAX*4], CL.
CMC. # ***
004EF0B5
                                                                                               58
EB 10
8472 5E
880C87
F5
56
10 28A5DF45
                                                                                                                                                                                                                       MOU BYTH CMC. STATE OF THE STAT
    004EF06D
004EF0C0
004EF0C1
004EF0C2
004EF0C7
004FE0C8
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   BASURA
       aa4FFans
       004EF0D5
                                                                                                                                                                                                                                 MOV DWORD PTR FS:[EAX],EDX
PUSH FlashSli.004EF12C
PUSH EBX
```

Se observa cómo se oculta a los ojos indiscretos la subrutina 4E6E54. Después de su ejecución, el programa vuelve a encriptar esta parte del código.

## **Aplicación**

Crearemos una rutina en asm que llamada antes y después de un trozo de código de nuestro programa realice el truco anteriormente visto.

```
encriptador proc uses eax ebx ecx longitud:sdword
     mov ebx, dword ptr ds:[ebp+4]
                                                  ; direccion de retorno del call
     mov ecx, longitud
                                                  ; longitud
.if longitud < 0
          add ebx, ecx
         sub ebx, 11
         neg ecx
      .endif
     mov eax, Odleea5f6h
                                                  ; semilla
     otro:
          xor byte ptr ds:[ebx],al
         rol eax,5
         add eax,0c5d78dbbh
                                                  ; numero magico
         ror eax, Oah
          inc ebx
          add eax,028743302h
                                                  ; numero magico
      loopd otro
 encriptador endp
```

Esta rutina sólo necesitará el parámetro de la longitud del código a procesar, puesto que ya hemos dicho que la pondremos *inmediatamente antes* e *inmediatamente después* del código a modificar. La implementación es relativamente sencilla. Incluimos las llamadas en el código fuente. Las etiquetas inicio: y final: sirven para calcular el valor del parámetro a transferir a la subrutina. Si el parámetro transferido es positivo entiende que es la llamada primera y si es negativo entiende que es la llamada segunda.

```
start:
   mov eax, offset final - offset inicio
   invoke encriptador, eax
inicio:
   finit
   mov ecx, 2
    .repeat
       fild qword ptr ds:qFibo[8*ecx-8]
       fild qword ptr ds:qFibo[8*ecx-16]
       fadd
       fistp qword ptr ds:qFibo[8*ecx]
                                       PARTE DEL PROGRAMA
       inc ecx
    .until ecx == 32
   mov ebx, 1
                                        A PROTEGER
    .repeat
       lea eax, [2*ebx]
       fild qword ptr ds:qFibo[8*eax-8]
       fild qword ptr ds:qFibo[8*eax]
       fistp qword ptr ds:qSolu[8*ebx]
       invoke crt sprintf, addr szTexto, addr szFormato, ebx, qword ptr ds:qSolu[8*ebx]
       invoke MessageBox, NULL, addr szTexto, addr szTitulo, MB ICONINFORMATION+MB OK
       inc ebx
    .until ebx == 16
final:
   mov eax, offset inicio - offset final
   invoke encriptador, eax
   invoke ExitProcess,0
end start
```

Compilamos y hacemos link con la opción /section:.text,wer para que el programa pueda sobrescribir su propio código. Ponemos un breakpoint en el RET de la subrutina encriptadora.

```
PUSH EBP
MOV EBP,ESP
PUSH EAX
PUSH EBX
PUSH ECX
MOV EBX,DWORD PTR DS:[EBP+4]
MOV ECX,DWORD PTR SS:[EBP+8]
CMP DWORD PTR SS:[EBP+8]
00401001
00401003
                               8BEC
                               50
53
00401004
00401005
                               3E:8B5D 04
00401006
                              8B4D 08
837D 08 00
7D 07
0040100A
00401011
                                                              JGE SHORT Problema.004010

ADD EBX,ECX

SUB EBX,08

NEG ECX

MOV EAX,D1EEASF6

"XOR BYTE PTR DS:[EBX],AL

ROL EAX,5

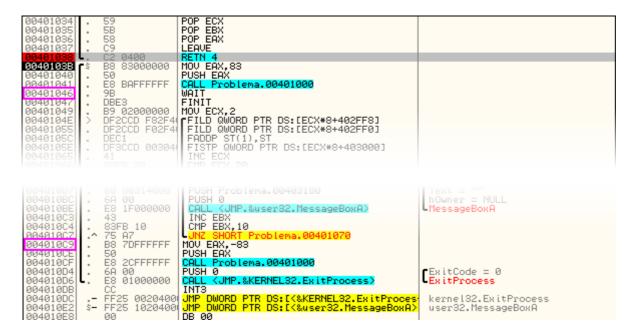
ADD EAX,CSD78DBB

ROR EAX,08

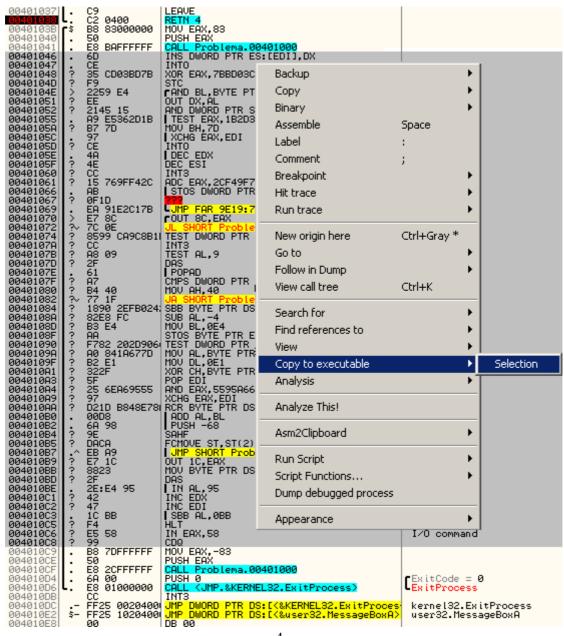
INC EBX

ADD EAX.28743302
                              03D9
83EB 0B
F7D9
B8 F6A5EED1
0040101
00401015
00401018
0040101A
0040101F
                               3003
C1C0 05
00401021
                              05 BB8DD7C5
C1C8 0A
00401024
00401029
                             43
05 02337428
E2 EB
59
5B
5B
                                                                ADD EAX,28743302
00401020
                                                              POP ECX
POP EBX
POP EAX
00401034
```

Tomamos nota del comienzo 401046 y final de la zona a encriptar 4010C8 (¡ojo!).



Damos RUN para en el BP. Seleccionamos la zona cambiada y grabamos los cambios mediante los "Copy to executable" y "Save file" habituales.



Ya tenemos nuestro programa listo para funcionar y protegido de vistazos indiscretos.

Agradecimientos a todos los CracksLatinos. Espero que os guste. Orniaco