



.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

Software: Sys.exe

Objetivo: Desempacar.

Tools: OllyDG 1.10, ImpRec, LordPE.

Fecha: 06/05/2012.

Cracker: Ivinson

Tutorial N°: 12

Team: CracksLatinos.

Download: <http://www.mediafire.com/?we6bx2br38yz3sb>



.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

## Introducción

Tenemos otro programa al cual desempacar con fines educativos porque somos una lista que busca e intenta constantemente que los integrantes puedan compartir sus conocimientos y ¿qué mejor manera que dejarlos por escritos para la posteridad? Cuando desempagues, crackees o consigas algo interesante en un software sería muy importante que lo publiques a través de un tutorial para que todos podamos seguir investigando y ampliando ese descubrimiento o novedad.

Reconozco que este arte no es tan fácil como parece, por lo menos cuando empezamos, pero poco a poco vamos desentrañando sus secretos. Esto será una lucha constante entre Programador-Cracker. ¿Quién ganará? Pienso que siempre habrá un equilibrio porque ni ellos dejarán de programar ni nosotros dejaremos de crackear.

¿Quién dejaría de hacer algo que le guste y qué le ponga retos interesantes?

Quiero cerrar esta introducción con la frase: Más sabe el diablo por viejo que por diablo.



.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

## Buscando un packer

Nuestro amigo RDGMAX creó un detector de packer muy bueno.

¿Quién no ha utilizado RDG Packer Detector? Revisemos nuestro PE a ver con qué sorpresa nos saldrá.

ASProtect v1.2x (New Strain)

En este PE, RDG no nos dice con qué está compilado. Por ahora, carguémoslo en Olly y veamos su EIP.

```
00401000 68 01D PUSH Sys.0139D001
00401005 E8 010 CALL Sys.0040100B
0040100A C3     RETN
0040100B C3     RETN
```

Detenemos el análisis con la tecla Espacio. ☺



Podemos buscar en memoria “M” a ver si conseguimos algo que nos indique con qué está compilado.

L E M T W



Podemos observar la DLL de Visual Basic “msvbvm60”

```

733A0000 00001000 msvbvm60 PE header
733A1000 000FC000 msvbvm60 .text code,import:
7349D000 0000D000 msvbvm60 ENGINE code,data
  
```

Volvamos al desensamblado de Olly.



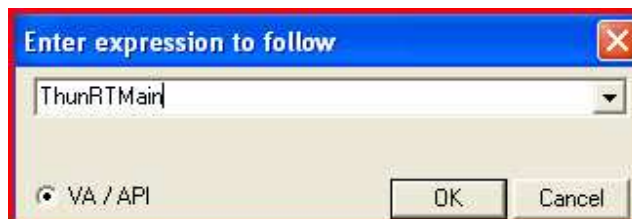
El OEP de un programa compilado en Visual Basic tiene esta apariencia:

```

00403162 FE25 58124000 JMP DWORD PTR DS:[401258] msvbvm60.ThunRTMain
00403163 68 70E44800 PUSH AudioCut.0048E470 ASCII "VB5!6&VB6FR.DLL"
0040316D E8 F0FFFFFF CALL AudioCut.00403162 JMP to msvbvm60.ThunRTMain
00403172 0000 ADD BYTE PTR DS:[EAX],AL
00403174 0000 ADD BYTE PTR DS:[EAX],AL
  
```

- 1) Un PUSH a la dirección de la DLL “VB5!6&VB6FR.DLL”
- 2) Un CALL al salto indirecto de la DLL “msvbv60.ThunRTMain”

Para encontrar el OEP en nuestra víctima buscaremos la String “ThunRTMain” presionando CTRL+G.



Caemos aquí:

```

733A35A4 55 PUSH EBP
733A35A5 8BEC MOV EBP,ESP
733A35A7 6AFF PUSH -1
  
```



Anotemos esta dirección 733A35A4. Reiniciemos Olly con CTRL+F2.  
Y vayamos a esa dirección con CTRL+G.



Luego, le ponemos un HBP en Execution. Presionamos F9 y caemos en:



En el Stack vemos algo interesante: “VB5!6&\*” el cual está incompleto.

Debería ser así: “VB5!6&VB6FR.DLL”. Por los momentos, solo anotemos la dirección interesante que vemos en el Stack 459004.



.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

## Buscando el OEP

Reiniciamos con CTRL+F2. Damos run con F9 y presionemos CTRL+B . Busquemos FF25 que son saltos indirectos.

Enter binary string to search for	
ASCII	٪
UNICODE	
HEX +00	FF 25

Caemos en:

```
0045845F  00FF  ADD BH,BH
00458461  25 BC1 AND EAX,4011BC
00458466 - FF25 9 JMP DWORD PTR DS:[401294]      msvbvm60.__vbaExceptionHandler
0045846C - FF25 E JMP DWORD PTR DS:[4012E4]      msvbvm60.__vbaFPException
```

Vayamos al final de esos JMP con CTRL+L o como ustedes quieran.

Y nos encontramos el OEP:

```
00458B5C - FF25 A JMP DWORD PTR DS:[4013A4]      msvbvm60.ThunRTMain
00458B62  0000  ADD BYTE PTR DS:[EAX],AL
00458B64 - E9 D37 JMP 01F9023C
00458B69  A4    MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00458B6A  65:40 INC EAX
00458B6C  7C 7B JL SHORT Sys.00458BE9
00458B6E  C9    LEAVE
00458B6F  0058 0 ADD BYTE PTR DS:[EAX],BL
00458B72  0000  ADD BYTE PTR DS:[EAX],AL
00458B74  3000  XOR BYTE PTR DS:[EAX],AL
00458B76  0000  ADD BYTE PTR DS:[EAX],AL
00458B78  50    PUSH EAX
```

Superfluous prefix



.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

Vemos que hay Stolen Bytes e instrucciones raras que no se parecen en nada a un OEP de un Visual Basic normal.

Entonces, nuestro OEP temporal será: 458B5C. Reinicimos Olly con CTRL+F2. Luego, CTRL+G y vamos a 458B5C.

```
00458B5C  90      NOP
00458B5D  D7      XLAT BYTE PTR DS:[EBX+AL]
00458B5E  BA 28E  MOV EDX,2EB7E828
00458B63  59      POP ECX
00458B64  1D F15  SBB EAX,4DD055F1
```

¡Que feo! ¿Verdad? Pongámosle un HBP en Execution y damos F9.

```
00458B5C - FF25 A JMP DWORD PTR DS:[4013A4]          msvbvm60.ThunRTMain
00458B62  0000   ADD BYTE PTR DS:[EAX],AL
00458B64 - E9 D37 JMP 01F9023C
00458B69  A4     MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00458B6A  65:40  INC EAX                               Superfluous prefix
00458B6C  7C 7B  JL SHORT Sys.00458BE9
00458B6E  C9     LEAVE
00458B6F  0058 0 ADD BYTE PTR DS:[EAX],BL
00458B72  0000   ADD BYTE PTR DS:[EAX],AL
00458B74  3000   XOR BYTE PTR DS:[EAX],AL
00458B76  0000   ADD BYTE PTR DS:[EAX],AL
00458B78  50     PUSH EAX
```

Ahora, sí. ☺ Arreglemos los Stolen Bytes de la siguiente manera:

PUSH 459004 ;Dirección de “VB5!6&VB6FR.DLL”.

CALL 458B5C ;Dirección del salto indirecto a “ThunRTMain”.

```
00458B5C - FF25 A JMP DWORD PTR DS:[4013A4]          msvbvm60.ThunRTMain
00458B62  68 049 PUSH Sys.00459004          ASCII "VB5!6&*"
00458B67  EB F0F CALL Sys.00458B5C         JMP to msvbvm60.ThunRTMain
```

Al arreglar esas instrucciones, debemos poner a cero las 2 siguientes porque están muy raras.



```

00458B6C 7C 7B JL SHORT Sys.00458BE9
00458B6E C9 LEAVE
00458B6F 0058 0 ADD BYTE PTR DS:[EAX],
00458B72 0000 ADD BYTE PTR DS:[EAX],
00458B74 3000 XOR BYTE PTR DS:[EAX],
  
```

Está quedando mejor. Hasta recuperamos un POP EAX:

```

00458B5C - FF25 A JMP DWORD PTR DS:[4013A4] msubvm60.ThunRTMain
00458B62 68 049 PUSH Sys.00459004 ASCII "VB5!6&*"
00458B67 E8 F0F CALL Sys.00458B5C JMP to msubvm60.ThunRTMain
00458B6C 0000 ADD BYTE PTR DS:[EAX],AL
00458B6E 0000 ADD BYTE PTR DS:[EAX],AL
00458B70 58 POP EAX
  
```

Arreglemos “VB5!6&\*” por esto “VB5!6&VB6FR.DLL”

```

00458B5C - FF25 A JMP DWORD PTR DS:[4013A4] msubvm60.ThunRTMain
00458B62 68 049 PUSH Sys.00459004 ASCII "VB5!6&*"
00458B67 E8 F0F CALL Sys.00458B5C JMP to
00458B6C 0000 ADD BYTE PTR DS:[EAX],AL
  
```

En el Dump, lo editamos así:

Address	Hex dump	ASCII
00459004	56 42 35 21 36 26 56 42	VB5!6&VB
0045900C	36 46 52 2E 44 40 40 00	6FR.DLL.

## Arreglando la IAT

La IAT de los Visual Basic comienzan en 401000. Por lo tanto, vayamos al Dump a esa dirección y elijamos la vista Long Address.

Address	Value	Comment
00401000	73469A19	msubvm60.EVENT_SINK_GetIDsOfNames





.-<lv!n\$on - lng. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

Vayamos al final:

```
004014A8 794730A7 msubvm60.rtcR8Ua1FromBstr
004014AC 00000000
004014B0 00140025
004014B4 00B0D868 Sys.00B0D868
```

Datos recopilados:

OEP: 458B62

Inicio de IAT: 401000

Fin de IAT: 4014AC

Datos para ImpRec:

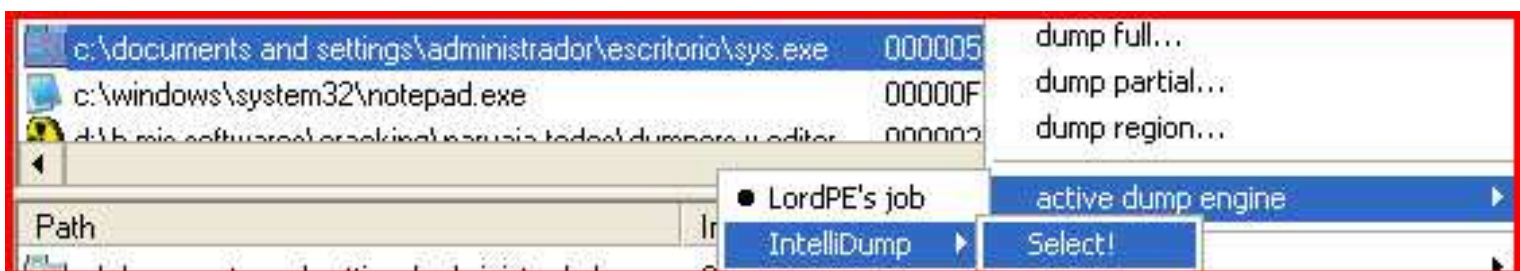
OEP: **58B62** (458B62-400000==58B62)

RVA: **1000** (401000-400000==1000)

Size: **4AC** (Fin - Inicio==Size. 4014AC-401000==4AC)

Dumpeando

Abramos LordPE y seleccionemos el proceso Sys.exe.





.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

Y luego:

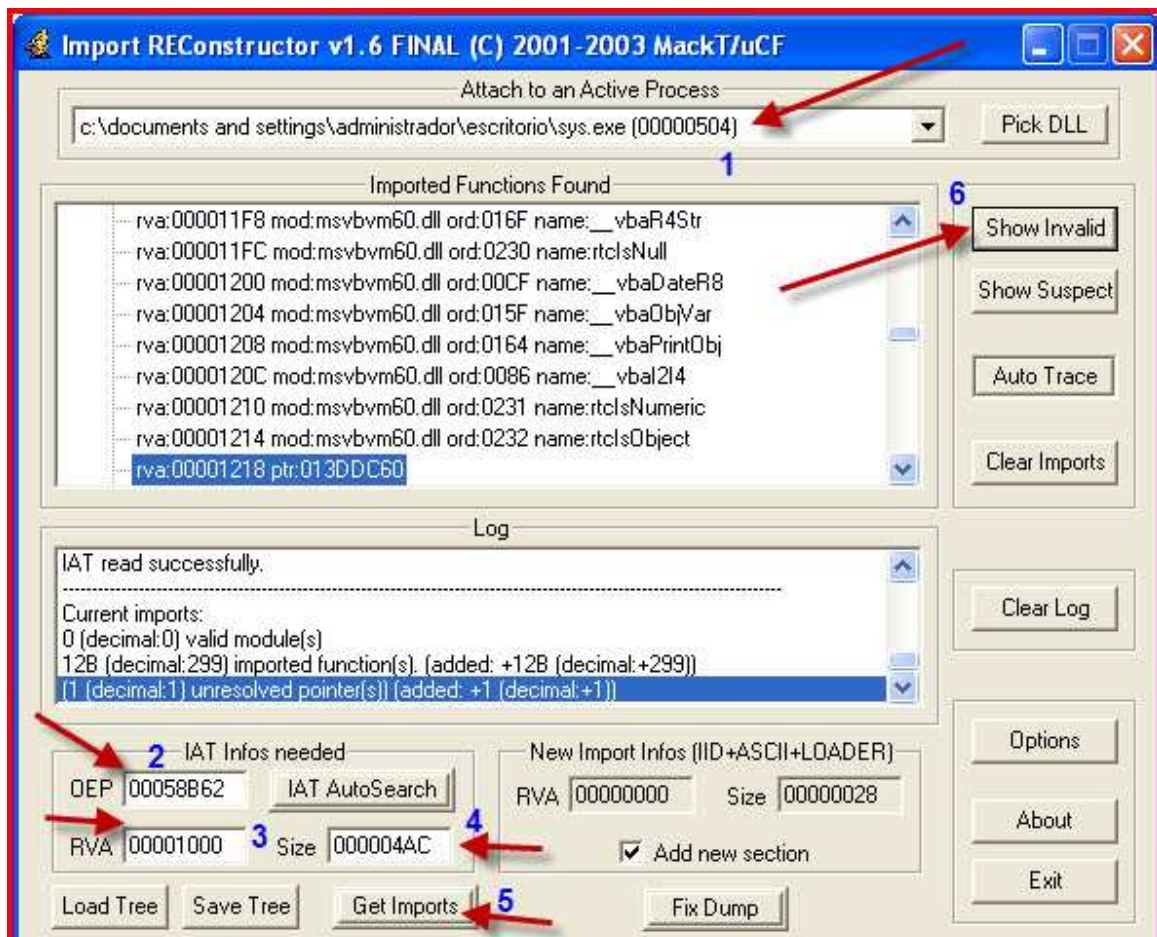


Guardamos el Dumpeado.

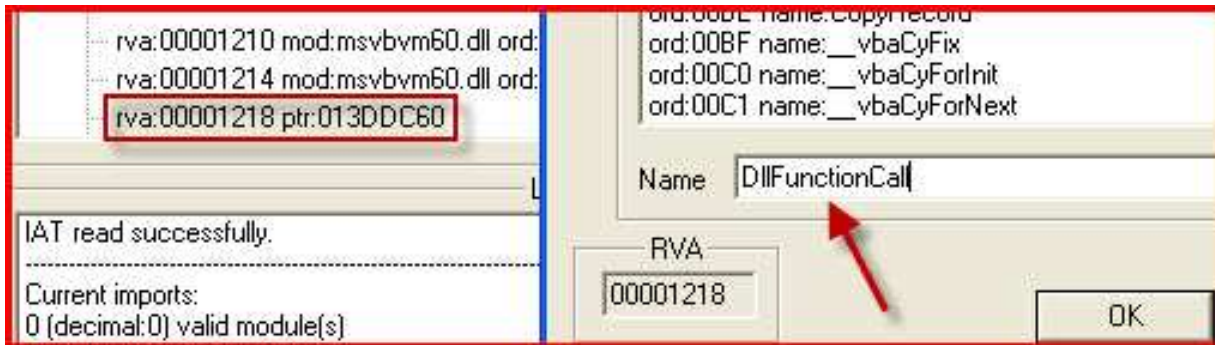


Arreglando el Dumpeado

Cargamos el original en ImpRec y colocamos los datos recopilados.



Vemos que solo tenemos una entrada mala. La de siempre. Le damos doble click y la escribimos a mano.



Ahora, le damos a Fix Dump y buscamos el Dumpeado.



Lo ejecutamos y:





.-<lv!n\$on - Ing. R3v3rS!v0>.-

CLS T3aM-No CoMp3t3Nc3

Gracias por leer.

Contacto: [ipadilla63@hotmail.com](mailto:ipadilla63@hotmail.com)