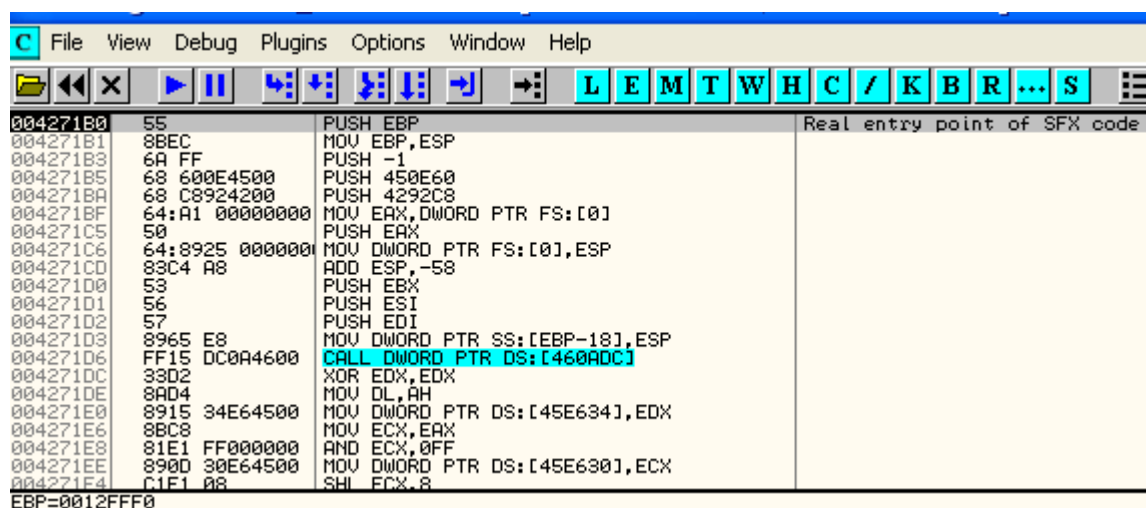


INTRODUCCION AL CRACKING CON OLLYDBG PARTE 37

En esta parte veremos algunos de los posibles metodos para **reparar entradas redireccionadas**, pero el tema no se agotara aquí, porque al igual que para llegar al OEP, hay tantos metodos como packers existen, y cuando veamos algunos otros packers saldrán metodos nuevos tanto para llegar al OEP como para reparar la IAT.

El tema es que estos metodos son ideas generales, y en otros packers pueden funcionar o no, o quizas necesitan cierta adaptacion según el caso, por lo cual es bueno siempre intentar y practicar mucho, e ir variando y probando.

De la misma forma que cuando vimos como llegar al OEP, daremos los metodos mas generales, y mas adelante cuando veamos distintos packers, adaptaremos estos metodos o usaremos alguno nuevo según el caso, que quizas no este aquí al no ser muy general.



```
File View Debug Plugins Options Window Help
[Icons] L E M T W H C / K B R ... S
004271B0 55 PUSH EBP
004271B1 8BEC MOV EBP,ESP
004271B3 6A FF PUSH -1
004271B5 68 600E4500 PUSH 450E60
004271BA 68 C8924200 PUSH 4292C8
004271BF 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
004271C5 50 PUSH EAX
004271C6 64:8925 00000000 MOV DWORD PTR FS:[0],ESP
004271CD 83C4 A8 ADD ESP,-58
004271D0 53 PUSH EBX
004271D1 56 PUSH ESI
004271D2 57 PUSH EDI
004271D3 8965 E8 MOV DWORD PTR SS:[EBP-18],ESP
004271D6 FF15 DC0A4600 CALL DWORD PTR DS:[460ADC]
004271DC 33D2 XOR EDX,EDX
004271DE 8AD4 MOV DL,AH
004271E0 8915 34E64500 MOV DWORD PTR DS:[45E634],EDX
004271E6 8BC8 MOV ECX,EAX
004271E8 81E1 FF000000 AND ECX,0FF
004271EE 890D 30E64500 MOV DWORD PTR DS:[45E630],ECX
004271F4 C1F1 08 SHL ECX,8
EBP=0012FFF0
Real entry point of SFX code
```

Bueno aquí estamos en el OEP del telock que como vimos la parte anterior tiene entradas redireccionadas a varias secciones que fueron creadas por el mismo packer en tiempo de desempacado.

| Address | Hex dump | ASCII |
|----------|--|--------------------|
| 0046080C | 00 00 00 00 00 00 00 00 00 00 00 00 00 F0 6B DA 77 |C.....-k rw |
| 0046081C | 1B 76 DA 77 F4 EA DA 77 E7 EB DA 77 83 78 DA 77 | +Vrw00rw00rw00rw |
| 0046082C | 00 00 00 00 DD 15 C5 58 2E BD C3 58 00 00 00 00 | ...!S+X.cFX... |
| 0046083C | 00 00 A1 00 11 00 A1 00 22 00 A1 00 33 00 A1 00 | ..l.4.l".l.3.l. |
| 0046084C | 41 00 A1 00 50 00 A1 00 5F 00 A1 00 7F 00 A1 00 | A.l.P.l..l.Δ.l. |
| 0046085C | 8D 00 A1 00 B0 00 A1 00 C1 00 A1 00 D2 00 A1 00 | i.l.Σ.l.I.l.É.l. |
| 0046086C | E4 00 A1 00 F4 00 A1 00 05 01 A1 00 24 01 A1 00 | \$.l.¶.l.Δ0i.\$0i. |
| 0046087C | 3E 01 A1 00 4F 01 A1 00 60 01 A1 00 71 01 A1 00 | >0i.00i.'0i.q0i. |
| 0046088C | 7F 01 A1 00 8E 01 A1 00 9D 01 A1 00 BD 01 A1 00 | Δ0i.Δ0i.00i.<0i. |
| 0046089C | CB 01 A1 00 EE 01 A1 00 FF 01 A1 00 10 02 A1 00 | τ0i.'0i.0i.Δ0i. |
| 004608AC | 22 02 A1 00 32 02 A1 00 43 02 A1 00 62 02 A1 00 | "0i.20i.C0i.b0i. |
| 004608BC | 7C 02 A1 00 8D 02 A1 00 9E 02 A1 00 AF 02 A1 00 | !0i.i0i.x0i.>0i. |
| 004608CC | BD 02 A1 00 CC 02 A1 00 DB 02 A1 00 FB 02 A1 00 | <0i.¶0i.00i.'0i. |
| 004608DC | 09 03 A1 00 2C 03 A1 00 3D 03 A1 00 4E 03 A1 00 | •0i.•0i.=0i.N0i. |
| 004608EC | 60 03 A1 00 70 03 A1 00 81 03 A1 00 A0 03 A1 00 | '0i.p0i.ü0i.Δ0i. |
| 004608FC | BA 03 A1 00 CB 03 A1 00 DC 03 A1 00 ED 03 A1 00 | 0i.τ0i.00i.Y0i. |
| 0046090C | FB 03 A1 00 0A 04 A1 00 19 04 A1 00 39 04 A1 00 | '0i.Δ0i.Δ0i.90i. |
| 0046091C | 47 04 A1 00 6A 04 A1 00 7B 04 A1 00 8C 04 A1 00 | G0i.j0i.†0i.i0i. |
| 0046092C | 9E 04 A1 00 AE 04 A1 00 BF 04 A1 00 DE 04 A1 00 | ×0i.<0i.70i.i0i. |
| 0046093C | F8 04 A1 00 09 05 A1 00 1A 05 A1 00 2B 05 A1 00 | ◊0i..0i.+0i.+0i. |
| 0046094C | 39 05 A1 00 48 05 A1 00 57 05 A1 00 77 05 A1 00 | 90i.H0i.W0i.w0i. |
| 0046095C | 85 05 A1 00 A8 05 A1 00 B9 05 A1 00 CA 05 A1 00 | Δ0i.Δ0i.¶0i.Δ0i. |
| 0046096C | DC 05 A1 00 EC 05 A1 00 00 00 9F 00 11 00 9F 00 | Δ0i.00i...f.Δ.f. |
| 0046097C | 22 00 9F 00 33 00 9F 00 41 00 9F 00 50 00 9F 00 | "f.3.f.A.f.P.f. |
| 0046098C | 5F 00 9F 00 7F 00 9F 00 8D 00 9F 00 B0 00 9F 00 | .f.Δ.f.l.f.Σ.f. |
| 0046099C | C1 00 9F 00 D2 00 9F 00 E4 00 9F 00 F4 00 9F 00 | I.f.É.f.Σ.f.¶.f. |

Ahi tenemos una vista de la IAT con sus entradas para reparar, realmente hay packers que redireccionan uno a dos entradas, por lo cual siempre es bueno conocer metodos para reparar una sola entrada, que es lo que veremos al inicio, obviamente si son muchas entradas a reparar esto no sirve, porque te vuelves viejo, pero hay que saber como detectar que api corresponde a cada entrada, a veces para verificar alguna entrada que es dudosa.

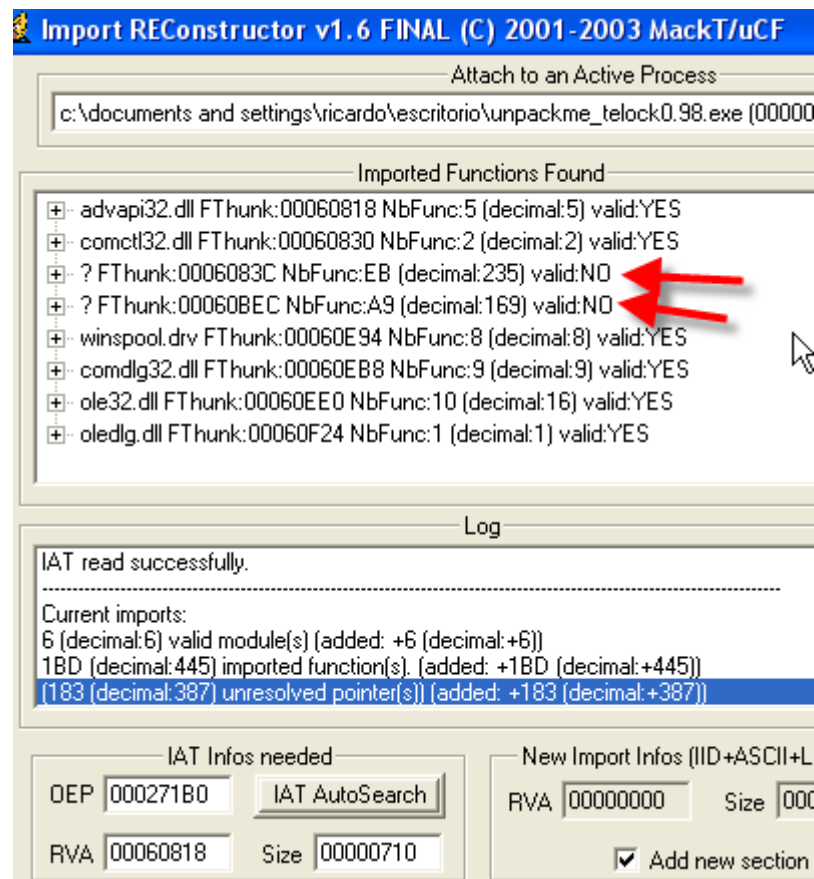
Tomemos la entrada que manualmente habíamos traceado y sabemos que finalmente va a GetVersion.

| | | | |
|----------|---------------|-------------------------------|--|
| 004271D1 | 56 | PUSH ESI | |
| 004271D2 | 57 | PUSH EDI | |
| 004271D3 | 8965 E8 | MOV DWORD PTR SS:[EBP-18],ESP | |
| 004271D6 | FF15 DC0A4600 | CALL DWORD PTR DS:[460ADC] | |
| 004271DC | 33D2 | XOR EDX,EDX | |
| 004271DE | 8AD4 | MOV DL,AH | |
| 004271E0 | 8915 34E64500 | MOV DWORD PTR DS:[45E634],EDX | |
| 004271E6 | 8BC8 | MOV ECX,EAX | |
| 004271E8 | 81E1 FF000000 | AND ECX,0FF | |
| 004271EE | 890D 30E64500 | MOV DWORD PTR DS:[45E630],ECX | |
| 004271F4 | C1E1 08 | SHL ECX,8 | |
| 004271F7 | 03CA | ADD ECX,EDX | |
| 004271F9 | 890D 2CE64500 | MOV DWORD PTR DS:[45E62C],ECX | |

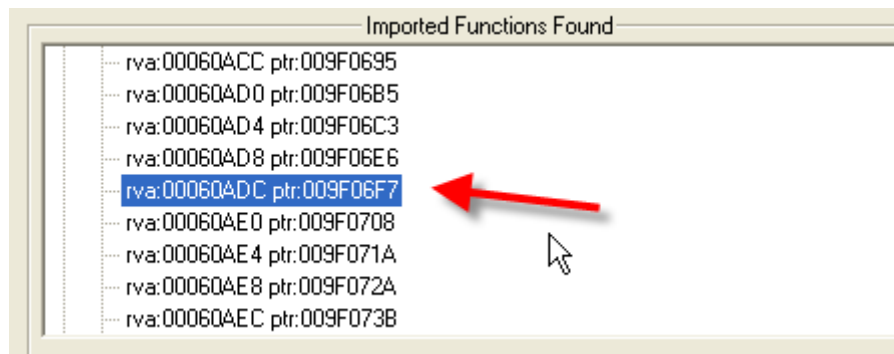
DS:[00460ADC]=009F06F7

| Address | Hex dump | ASCII |
|----------|---|------------------|
| 00460ADC | F7 06 9F 00 08 07 9F 00 1A 07 9F 00 2A 07 9F 00 | ~*f.~*f.~*f.~*f. |
| 00460AEC | 3B 07 9F 00 5A 07 9F 00 74 07 9F 00 85 07 9F 00 | ~*f.~*f.~*f.~*f. |
| 00460AFC | 96 07 9F 00 07 07 9F 00 65 07 9F 00 C4 07 9F 00 | ~*f.~*f.~*f.~*f. |
| 00460B0C | D3 07 9F 00 F3 07 9F 00 01 08 9F 00 24 08 9F 00 | ~*f.~*f.~*f.~*f. |
| 00460B1C | 35 08 9F 00 46 08 9F 00 58 08 9F 00 68 08 9F 00 | ~*f.~*f.~*f.~*f. |

Allí está el CALL y en el DUMP, la entrada de la IAT 460ADC con su contenido que apunta en mi máquina a 9F06F7, también tenemos el IMP REC abierto con todos los valores que hallamos a mano, de RVA, SIZE y OEP escritos, y a la vista las entradas malas.



Pongamos el IMP REC también para ver esa entrada

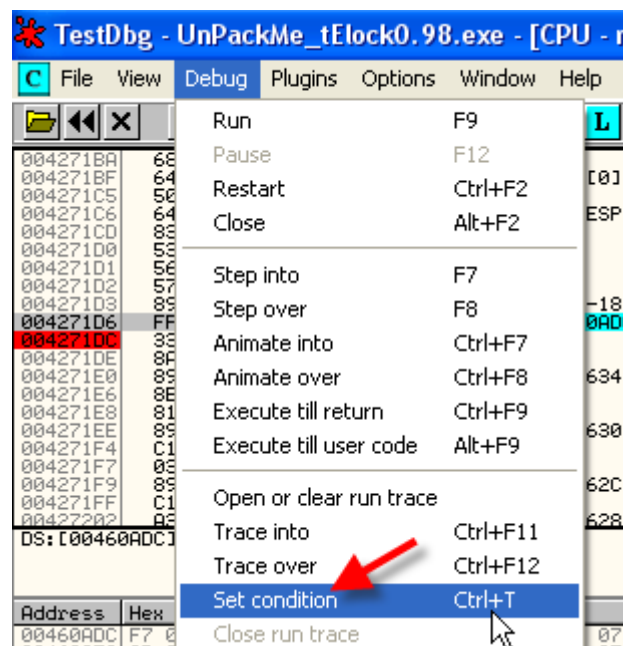


60ADC corresponde a la entrada de la IAT 460ADC.

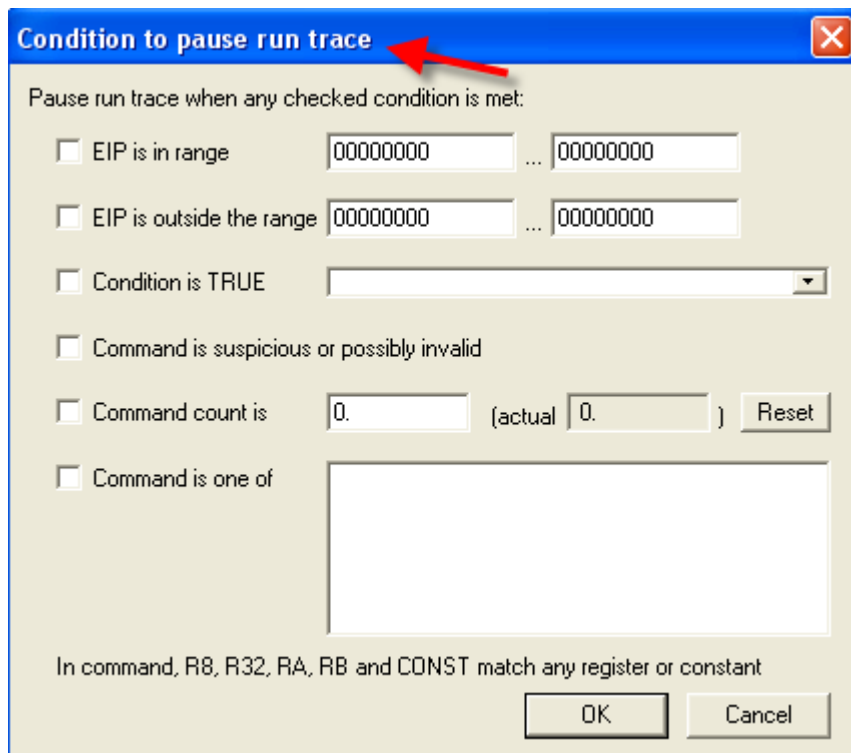
Volvamos al OLLYDBG y a nuestra entrada, ya vimos que traceando a mano llegamos a una api, sabemos que el OLLYDBG trae un traceador incorporado, vamos a usarlo para llegar mas rapido, ya que aquí fueron 5 o 6 líneas que traceamos y ya llegamos a la api, pero hay packers que dan vueltas y vueltas antes de llegar a la misma, por lo cual el traceo automatico suele ser util.

| | | |
|----------|---------------|-------------------------------|
| 004271D1 | 56 | PUSH ESI |
| 004271D2 | 57 | PUSH EDI |
| 004271D3 | 8965 E8 | MOV DWORD PTR SS:[EBP-18],ESP |
| 004271D6 | FF15 DC0A4600 | CALL DWORD PTR DS:[460ADC] |
| 004271D6 | 33D2 | XOR EDX,EDX |
| 004271DE | 8AD4 | MOV DL,AH |
| 004271F0 | 8915 34F64500 | MOV DWORD PTR DS:[45F634],EDX |

Ponemos antes de tracear un BP en el retorno de la api, no sea cosa que nos equivoquemos y quede traceando sin parar nunca.

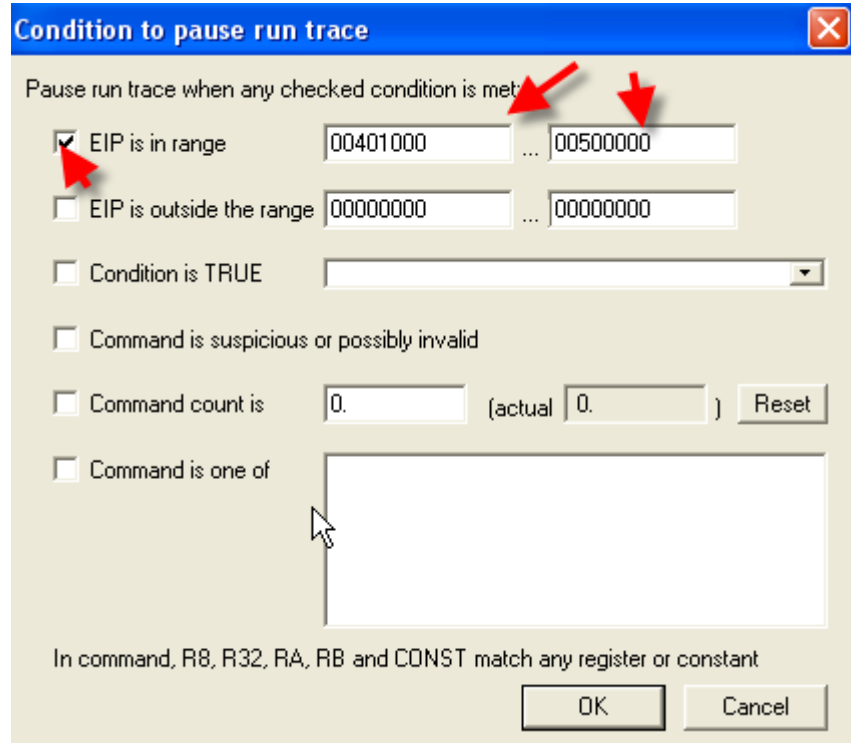


En DEBUG-SET CONDITION tenemos la posibilidad de elegir las opciones en las cuales el OLLYDBG detendra el traceo.



Allí vemos la ventana de CONDICIONES PARA PAUSAR EL TRACEO, veremos algunas que pueden funcionar y se pueden adaptar según el caso.

EIP IS IN RANGE significa que si ponemos la tilde en este renglon, OLLYDBG parara cuando el EIP se encuentre dentro del rango de valores que especifiquemos en la derecha, por ejemplo.



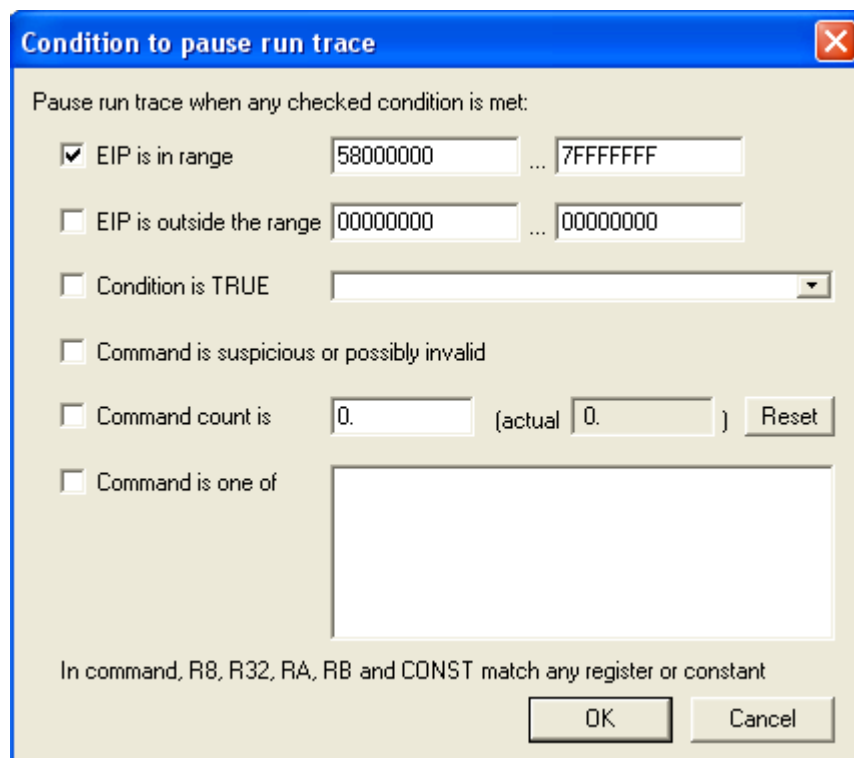
Este ejemplo significa que OLLYDBG traceara hasta que vea que EIP vale algun valor comprendido entre 401000 y 500000 por supuesto esto es un ejemplo, y no nos sirve en este caso, ya que queremos que se detenga en la api.

Para hallar los valores entre los cuales queremos que OLLYDBG se detenga en este caso miramos el mapa de memoria.

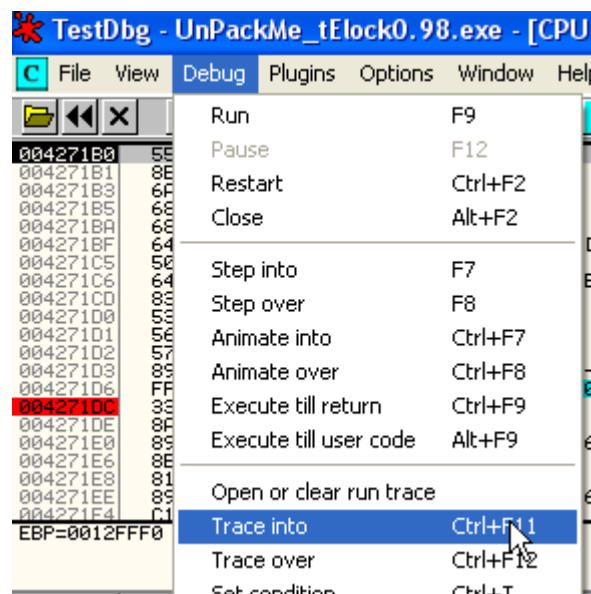
| | | | | | | | | |
|----------|----------|----------|--------|--------------|------|-----|-----|-----------------------------|
| 00320000 | 00006000 | | | | Map | R | R | \Device\HarddiskVolume1\Win |
| 00330000 | 00041000 | | | | Map | R | R | |
| 00380000 | 00001000 | | | | Priv | RWE | RWE | |
| 00390000 | 00001000 | | | | Priv | RWE | RWE | |
| 003A0000 | 00001000 | | | | Priv | RW | RW | |
| 003B0000 | 00001000 | | | | Priv | RW | RW | |
| 00400000 | 00001000 | UnPackMe | | PE header | Imag | RW | RWE | |
| 00401000 | 0004A000 | UnPackMe | .teddy | code | Imag | RW | RWE | |
| 0044B000 | 0000C000 | UnPackMe | .teddy | data | Imag | RW | RWE | |
| 00457000 | 00009000 | UnPackMe | .teddy | | Imag | RW | RWE | |
| 00460000 | 00003000 | UnPackMe | .teddy | | Imag | RW | RWE | |
| 00463000 | 00002000 | UnPackMe | .rsrc | resources | Imag | RW | RWE | |
| 00465000 | 00004000 | UnPackMe | .teddy | SFX, imports | Imag | RW | RWE | |
| 00470000 | 00008000 | | | | Map | R E | R E | |
| 00530000 | 00002000 | | | | Map | R E | R E | |
| 00540000 | 00103000 | | | | Map | R | R | |
| 00650000 | 000EE000 | | | | Map | R E | R E | |
| 00970000 | 00001000 | | | | Priv | RW | RW | |
| 009F0000 | 00002000 | | | | Priv | RW | RW | |
| 00A00000 | 00002000 | | | | Priv | RW | RW | |
| 00A10000 | 00001000 | | | | Priv | RW | RW | |
| 00A20000 | 00004000 | | | | Priv | RW | RW | |
| 00A30000 | 00003000 | | | | Map | R | R | \Device\HarddiskVolume1\Win |
| 00A40000 | 00004000 | | | | Priv | RW | RW | |
| 00A50000 | 00003000 | | | | Priv | RW | RW | |
| 00A60000 | 00002000 | | | | Map | R | R | |
| 00A70000 | 00001000 | | | | Priv | RW | RW | |
| 01270000 | 00002000 | | | | Map | R | R | |
| 58C30000 | 00001000 | COMCTL32 | | PE header | Imag | R | RWE | |
| 58C31000 | 00070000 | COMCTL32 | .text | code, import | Imag | R | RWE | |
| 58CA1000 | 00003000 | COMCTL32 | .data | data | Imag | R | RWE | |
| 58CA4000 | 0001F000 | COMCTL32 | .rsrc | resources | Imag | R | RWE | |
| 58CC3000 | 00004000 | COMCTL32 | .reloc | relocations | Imag | R | RWE | |
| 58480000 | 00001000 | undmxfm | | PE header | Imag | R | RWE | |
| 58481000 | 00003000 | undmxfm | .text | code, import | Imag | R | RWE | |
| 58484000 | 00001000 | undmxfm | .data | data | Imag | R | RWE | |
| 58485000 | 00001000 | undmxfm | .rsrc | resources | Imag | R | RWE | |
| 58486000 | 00001000 | undmxfm | .reloc | relocations | Imag | R | RWE | |
| 5D160000 | 00001000 | serwvdrv | | PE header | Imag | R | RWE | |
| 5D161000 | 00003000 | serwvdrv | .text | code, import | Imag | R | RWE | |
| 5D164000 | 00001000 | serwvdrv | .data | data | Imag | R | RWE | |
| 5D165000 | 00001000 | serwvdrv | .rsrc | resources | Imag | R | RWE | |
| 5D166000 | 00001000 | serwvdrv | .reloc | relocations | Imag | R | RWE | |
| 72F80000 | 00001000 | WINSPOOL | | PE header | Imag | R | RWE | |
| 72F81000 | 00020000 | WINSPOOL | .text | code, import | Imag | R | RWE | |
| 72FA1000 | 00002000 | WINSPOOL | .data | data | Imag | R | RWE | |
| 72FA3000 | 00001000 | WINSPOOL | .rsrc | resources | Imag | R | RWE | |
| 72FA4000 | 00002000 | WINSPOOL | .reloc | relocations | Imag | R | RWE | |
| 74CC0000 | 00001000 | oledlg | | PE header | Imag | R | RWE | |
| 74CC1000 | 00011000 | oledlg | .text | code, import | Imag | R | RWE | |
| 74CD2000 | 00002000 | oledlg | .data | data | Imag | R | RWE | |
| 74CD4000 | 0000B000 | oledlg | .rsrc | resources | Imag | R | RWE | |
| 74CDF000 | 00001000 | oledlg | .reloc | relocations | Imag | R | RWE | |
| 76360000 | 00001000 | comdlg32 | | PE header | Imag | R | RWE | |
| 76361000 | 00030000 | comdlg32 | .text | code, import | Imag | R | RWE | |
| 76391000 | 00004000 | comdlg32 | .data | data | Imag | R | RWE | |
| 76395000 | 00012000 | comdlg32 | .rsrc | resources | Imag | R | RWE | |
| 763A7000 | 00003000 | comdlg32 | .reloc | relocations | Imag | R | RWE | |
| 76B00000 | 00001000 | WINMM | | PE header | Imag | R | RWE | |
| 76B01000 | 0001F000 | WINMM | .text | code, import | Imag | R | RWE | |
| 76B20000 | 00002000 | WINMM | .data | data | Imag | R | RWE | |
| 76B22000 | 0000A000 | WINMM | .rsrc | resources | Imag | R | RWE | |



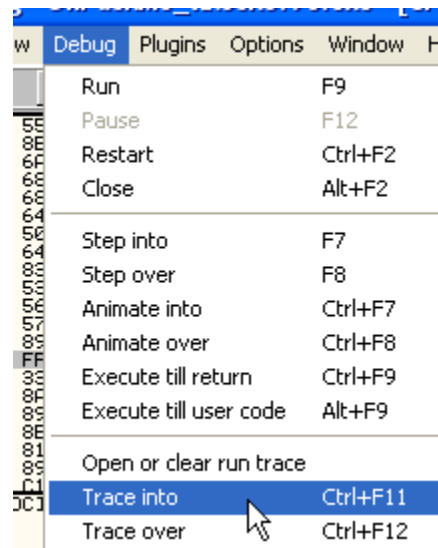
Allí vemos marcado en celeste la zona que nos interesaría que se detenga OLLYDBG al tracear, o sea desde donde se encuentra la primera dll en adelante, no queremos que pare ni en las secciones del exe, ni en las secciones creadas por el packer ya que ese es código intermedio, necesitamos que pare en las dlls, por lo tanto si marcamos la primera casilla y ponemos que EIP este entre 58000000 y 7FFFFFFF que es la dirección máxima seguro que para en alguna dll (aclaración no importa que no elegimos la dirección exacta de la primera dll ya que vemos que no hay código entre 58000000 y el inicio de la primera dll 58c30000 así que allí no va a parar, pero para evitar poner cifras exactas puedo sin problemas redondear)



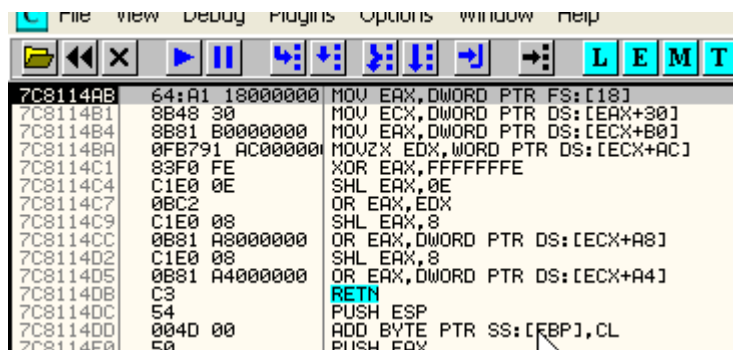
Quiere decir entonces que OLLYDBG ejecutara traceando linea a linea y en cada una verificara que se cumpla esa condicon, o sea que EIP este en la seccion de alguna dll y si es asi parara.



Lleguemos hasta el CALL poniendo un BP en el mismo.



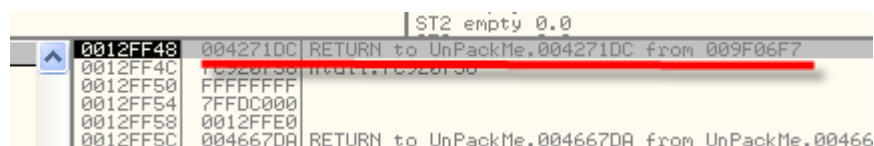
Es importante elegir TRACE INTO para que tracee linea a linea, si elegimos TRACE OVER no entrara en los calls y puede fallar.



Alli paro por la condicion que colocamos si vemos en OLLYDBG abajo

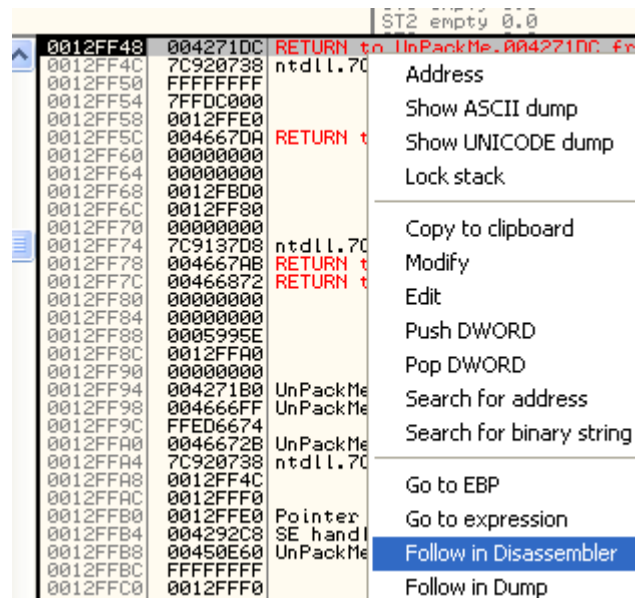


Paro porque EIP esta en el rango 58000000-7FFFFFFF, es bueno verificar esto para ver que no haya parado por alguna excepcion u otro motivo.



Lo siguiente que hay que verificar es que al ejecutar la api vuelva al programa a continuacion del CALL, porque hay packers que para confundir van primero a una api, y a continuacion a una segunda api, por lo tanto la api verdadera sera la ultima, y en la cual veamos en el valor de retorno de la misma, que retorna al programa a la instruccion siguiente al CALL.

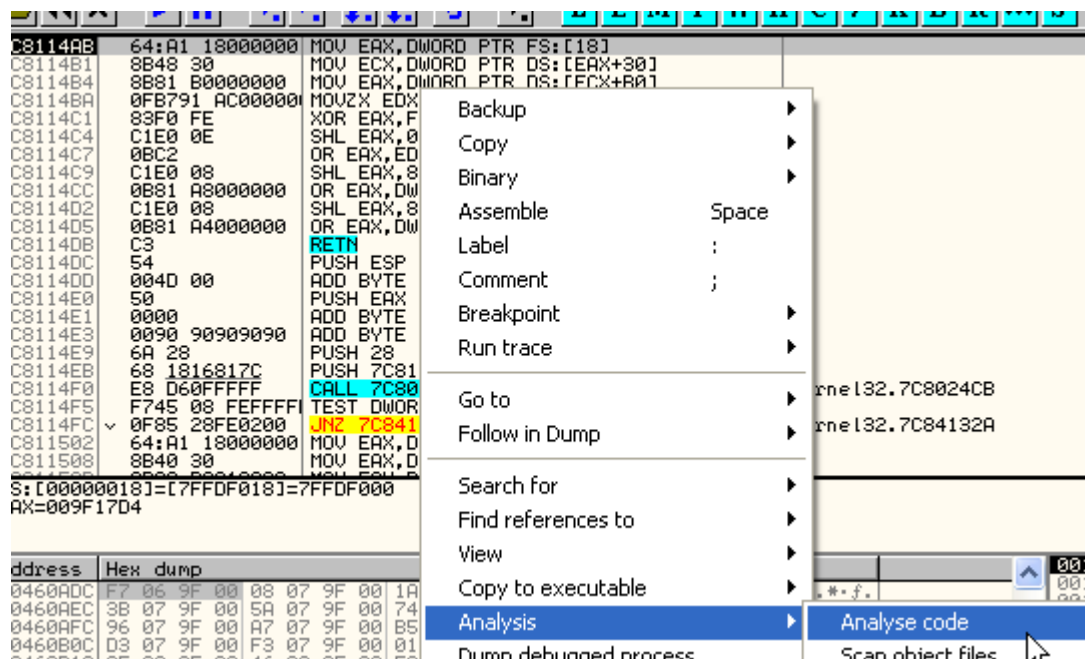
Por supuesto en este caso, retorna a la linea siguiente del CALL de donde partimos.



| | | |
|----------|---------------|-------------------------------|
| 004271D3 | 8965 E8 | MOV DWORD PTR SS:[EBP-18],ESP |
| 004271D6 | FF15 DC0A4600 | CALL DWORD PTR DS:[460ADC] |
| 004271DC | 33D2 | XOR EDX,EDX |
| 004271DE | 8AD4 | MOV DL,AH |
| 004271E0 | 8915 34E64500 | MOV DWORD PTR DS:[45E634],EDX |
| 004271E6 | 8BC8 | MOV ECX,EAX |

Vuelve a 4271DC como corresponde.

Bueno ahora nos falta ver que api es porque OLLYDBG no nos dice nada, la primera forma es analizar el codigo aquí mismo de la dll.



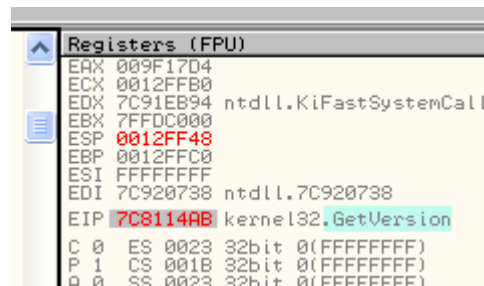
```

7C8114AB 64:A1 180000 MOV EAX,DWORD PTR FS:[18]
7C8114B1 8B48 30     MOV ECX,DWORD PTR DS:[EAX+30]
7C8114B4 8B81 B00000 MOV EAX,DWORD PTR DS:[ECX+B0]
7C8114B8 0FB791 AC000 MOVZX EDX,WORD PTR DS:[ECX+AC]
7C8114C1 83F0 FE     XOR EAX,FFFFFFF
7C8114C4 C1E0 0E     SHL EAX,0E
7C8114C7 0BC2       OR EAX,EDX
7C8114C9 C1E0 08     SHL EAX,8
7C8114CC 0B81 A80000 OR EAX,DWORD PTR DS:[ECX+A8]
7C8114D2 C1E0 08     SHL EAX,8
7C8114D5 0B81 A40000 OR EAX,DWORD PTR DS:[ECX+A4]
7C8114DB C3         RETN
7C8114DC 54         DB 54
7C8114DD 00         DB 00
7C8114DE 4D         DB 4D
7C8114DF 00         DB 00
7C8114E0 50         DB 50
7C8114E1 00         DB 00
7C8114E2 00         DB 00
7C8114E3 00         DB 00
7C8114E4 90         NOP
7C8114E5 90         NOP
7C8114E6 90         NOP
7C8114E7 90         NOP
7C8114E8 90         NOP

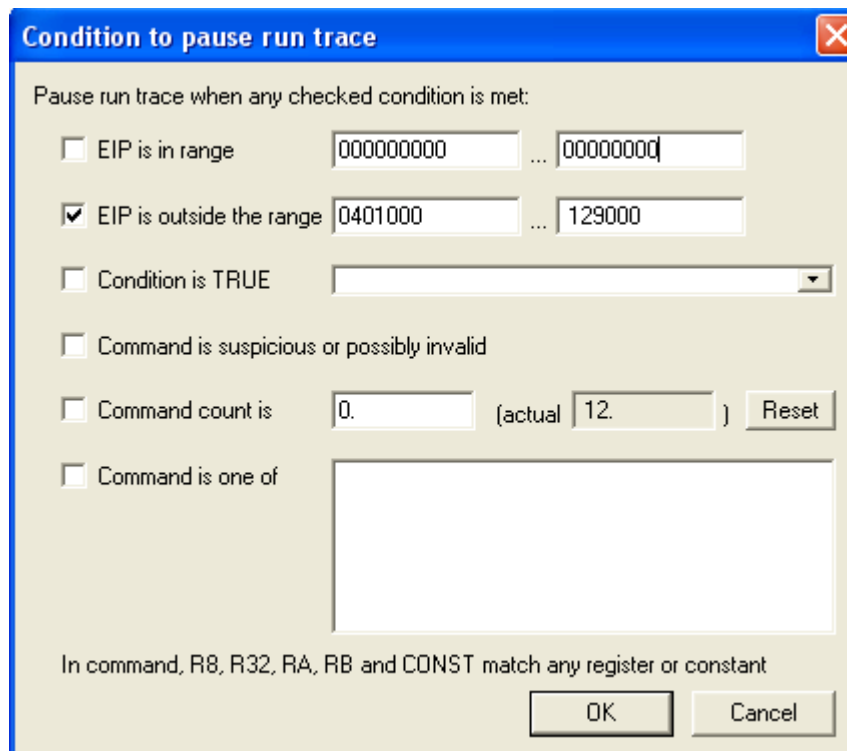
FS:[00000018]=7FFDF018=7FFDF000
EAX=009F17D4
kernel32.GetVersion

```

Alli vemos justo abajo en la aclaracion que OLLYDBG nos muestra el nombre de la api lo mismo que al lado del valor de EIP.



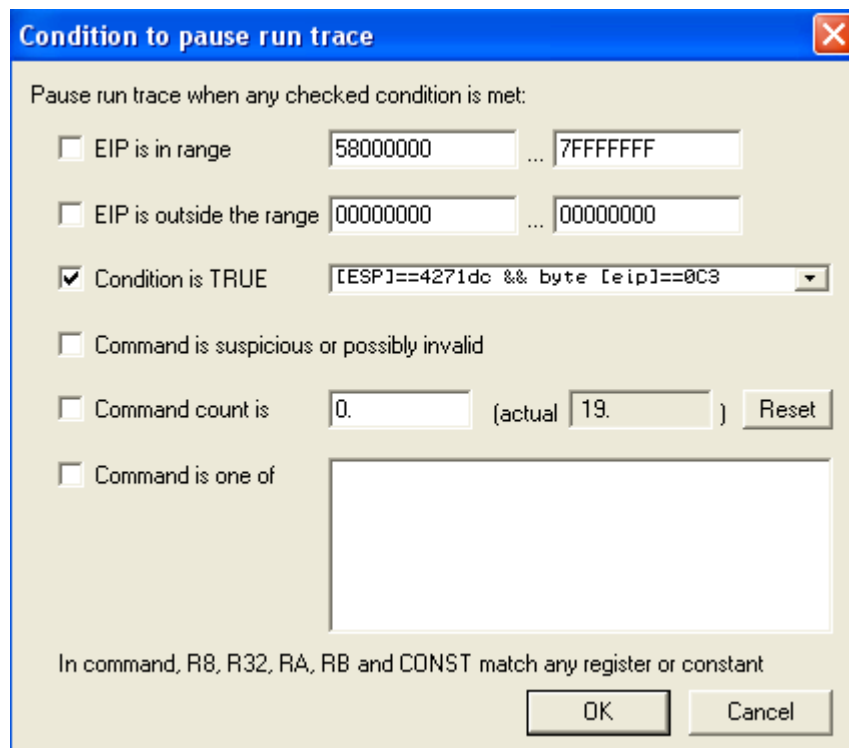
Otra condiciones posibles para el traceo desde el CALL para llegar hasta la api son:



Este caso es la inversa del anterior OLLYDBG chequea que EIP este fuera del rango de las secciones del exe,

y las secciones que hay antes de la primera dll.

Otra posibilidad



Para algun caso raro que el packer cree secciones mezcladas entre las dlls, puede utilizarse un metodo combinado como este que parara cuando halle un RET y ademas cuando la primera linea del stack sea 4271DC que es el retorno de la api. Este metodo apunta a parar en el retorno de la api, y muchas veces es efectivo sobre todo en packers que emulan las primeras lineas de la api y saltan a la tercera o cuarta linea de la misma, generalmente el RET siempre es respetado y parara en el mismo, si reinicio el OLLYDBG y luego de nuevo al call puedo probarlo rapidamente.

O sea aquí se deben cumplir dos condiciones a la vez, que estan unidas por && lo cual asegura que sean las dos condiciones las que se cumplan && equivale a Y.

Si fuera que quisiera que se detenga cuando alguna de las dos condiciones solas se cumpla, en ese caso usaria || que equivale a O.

O sea que

[ESP]==4271dc && byte [eip]==0C3

significa que se cumplan ambas condiciones a la vez, o sea

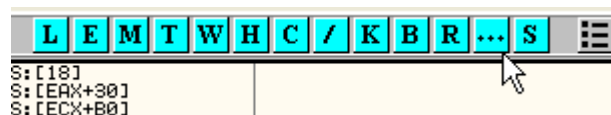
[ESP]==4271dc (que la primera linea del stack sea el punto de retorno esperado de la api)

byte [eip]==0C3 (y que el contenido de EIP o sea el byte que se esta ejecutando sea un RET o sea C3)

Si desde el CALL lanzamos a tracear con esa condicion, vemos que para en el RET de la api.

| | | | |
|----------|---------------|--------------------------------|------|
| 7C8114A8 | 90 | NOP | |
| 7C8114A9 | 90 | NOP | |
| 7C8114AA | 90 | NOP | |
| 7C8114AB | 64:A1 180000 | MOV EAX,DWORD PTR FS:[18] | |
| 7C8114B1 | 8B48 30 | MOV ECX,DWORD PTR DS:[EAX+30] | |
| 7C8114B4 | 8B81 B0000000 | MOV EAX,DWORD PTR DS:[ECX+B0] | |
| 7C8114BA | 0FB791 AC000 | MOVZX EDX,WORD PTR DS:[ECX+AC] | |
| 7C8114C1 | 83F0 FE | XOR EAX,FFFFFFFF | |
| 7C8114C4 | C1E0 0E | SHL EAX,0E | |
| 7C8114C7 | 0BC2 | OR EAX,EDX | |
| 7C8114C9 | C1E0 08 | SHL EAX,8 | |
| 7C8114CC | 0B81 A8000000 | OR EAX,DWORD PTR DS:[ECX+A8] | |
| 7C8114D2 | C1E0 08 | SHL EAX,8 | |
| 7C8114D5 | 0B81 A4000000 | OR EAX,DWORD PTR DS:[ECX+A4] | |
| 7C8114DB | C3 | RETN | |
| 7C8114DC | 54 | DB 54 | CHAR |
| 7C8114DD | 00 | DB 00 | |
| 7C8114DE | 4D | DB 4D | CHAR |

Conditional pause: [ESP]==4271dc && byte [eip]==0C3



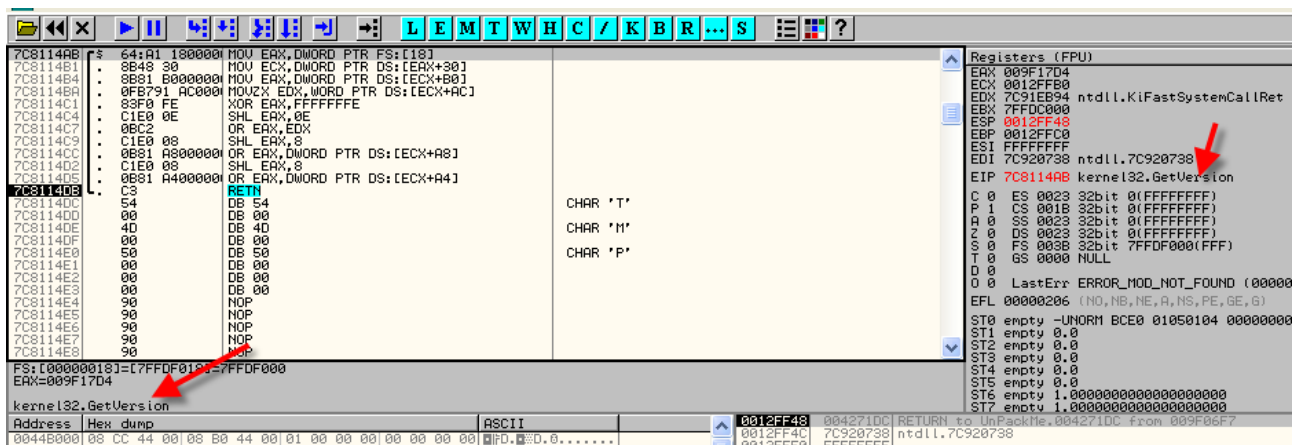
Al dispararse ambas condiciones a la vez, se detiene el traceo..

Ahora que estamos en el RET, se nos complica saber el nombre de la api que ejecuto, pues no estamos en el inicio de la misma, lo mismo nos ocurre cuando el packer simula las dos o tres primeras lineas de la api y estamos en la 4ta o 5ta linea de la misma, aquí OLLYDBG no nos muestra el nombre aunque hay varios recursos para hallarlo.

Veremos en el listado de lo que traceo, para ello vamos al boton con los tres puntitos.

| Back | Thread | Module | Address | Command | Comment or message |
|------|--------|----------|----------|--------------------------------|--------------------|
| 19. | Main | UnPackMe | 004271D6 | CALL DWORD PTR DS:[460ADC] | |
| 18. | Main | | 009F06F7 | TEST ESP,ESP | |
| 17. | Main | | 009F06F9 | JNS SHORT 009F06FE | |
| 16. | Main | | 009F06FE | INC EAX | |
| 15. | Main | | 009F06FF | MOV EAX,9F17D3 | |
| 14. | Main | | 009F0704 | INC EAX | |
| 13. | Main | | 009F0705 | PUSH DWORD PTR DS:[EAX] | |
| 12. | Main | | 009F0707 | RETN | |
| 11. | Main | kernel32 | 7C8114AB | MOV EAX,DWORD PTR FS:[18] | |
| 10. | Main | kernel32 | 7C8114B1 | MOV ECX,DWORD PTR DS:[EAX+30] | |
| 9. | Main | kernel32 | 7C8114B4 | MOV EAX,DWORD PTR DS:[ECX+B0] | |
| 8. | Main | kernel32 | 7C8114BA | MOVZX EDX,WORD PTR DS:[ECX+AC] | |
| 7. | Main | kernel32 | 7C8114C1 | XOR EAX,FFFFFFFF | |
| 6. | Main | kernel32 | 7C8114C4 | SHL EAX,0E | |
| 5. | Main | kernel32 | 7C8114C7 | OR EAX,EDX | |
| 4. | Main | kernel32 | 7C8114C9 | SHL EAX,8 | |
| 3. | Main | kernel32 | 7C8114CC | OR EAX,DWORD PTR DS:[ECX+A8] | |
| 2. | Main | kernel32 | 7C8114D2 | SHL EAX,8 | |
| 1. | Main | kernel32 | 7C8114D5 | OR EAX,DWORD PTR DS:[ECX+A4] | |
| 0. | Main | kernel32 | 7C8114DB | RETN | |

Vemos obviamente que la primera linea que traceo en la dll es la marcada por la flecha, aunque no nos muestra el nombre de la api, asi que hagamos analisis del codigo, y luego hagamos doble click en esta linea que marca la flecha.



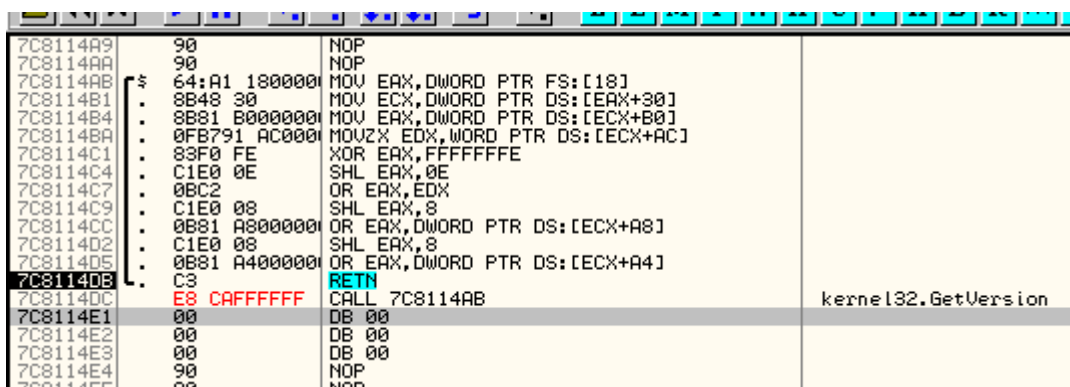
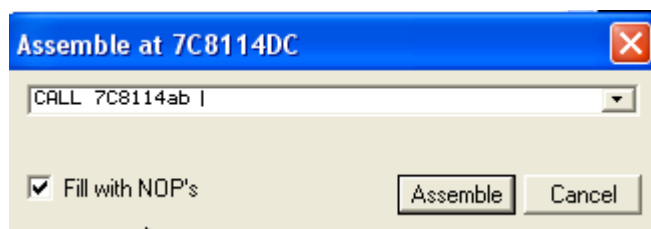
Vemos que OLLYDBG nos muestra cuanto valian los registros en ese momento, por eso se ponen en gris ya que no son los valores actuales, y ademas nos muestra la informacion como si estuviéramos ejecutando esa linea, entre lo cual podemos ver el nombre de la api, lo mismo ocurriria si luego de un traceo vamos apretando la tecla menos, OLLYDBG ira hacia atrás mostrando la informacion y los valores que guardo al pasar por cada instrucción.

En el caso de packers que emulan las dos o tres primeras lineas de la api y al tracear caemos por la cuarta o quinta linea, el traceador no pasara por la primera linea pues ellas son emuladas por el packer, por lo cual a veces hay que arreglarse un poco a ojo, pero un metodo casero que no falla es hacer lo siguiente jeje (perdon por lo berreta pero la verdad que sirve)

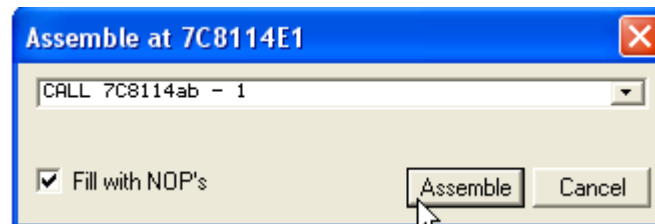
En cualquier parte vacia abajo del ret escribimos con assemble, o apretando la barra espaciadora, un CALL a la linea que sospechamos es el inicio de la api visualmente, en este caso

CALL 7C8114ab

Parece el inicio de la api en mi maquina, en la suya ponen la direccion donde les parece visualmente que puede comenzar la api.



Al aceptar, si es el inicio de una api, OLLYDBG nos mostrara el nombre, si nos equivocamos podemos repetir con.



Y así en la zona alrededor de donde parece que se inicia la api, hasta que OLLYDBG nos de el nombre aunque es generalmente bastante facil darse cuenta de donde se inicia, por los corchetes que pone OLLYDBG al analizar, alli vemos que el corchete empieza en el inicio de la api.

| | | | |
|----------|---------------|--------------------------------|---------------------|
| 7C8114AA | 90 | NOP | |
| 7C8114AB | 64:A1 180000 | MOV EAX,DWORD PTR FS:[18] | |
| 7C8114B1 | 8B48 30 | MOV ECX,DWORD PTR DS:[EAX+30] | |
| 7C8114B4 | 8B81 B0000000 | MOV EAX,DWORD PTR DS:[ECX+B0] | |
| 7C8114BA | 0FB791 AC0000 | MOVZX EDX,WORD PTR DS:[ECX+AC] | |
| 7C8114C1 | 83F0 FE | XOR EAX,FFFFFFFF | |
| 7C8114C4 | C1E0 0E | SHL EAX,0E | |
| 7C8114C7 | 0BC2 | OR EAX,EDX | |
| 7C8114C9 | C1E0 08 | SHL EAX,8 | |
| 7C8114CC | 0B81 A8000000 | OR EAX,DWORD PTR DS:[ECX+A8] | |
| 7C8114D2 | C1E0 08 | SHL EAX,8 | |
| 7C8114D5 | 0B81 A4000000 | OR EAX,DWORD PTR DS:[ECX+A4] | |
| 7C8114D8 | C3 | RETN | |
| 7C8114D9 | E8 CAFFFFFF | CALL 7C8114AB | kernel32.GetVersion |

Quitamos lo que escribimos con UNDO SELECTION

| | | | |
|----------|---------------|------------------------------|-------------------------|
| 7C8114D2 | C1E0 08 | SHL EAX,8 | |
| 7C8114D5 | 0B81 A4000000 | OR EAX,DWORD PTR DS:[ECX+A4] | |
| 7C8114D8 | C3 | RETN | |
| 7C8114DC | E8 CAFFFFFF | CALL 7C8114AB | |
| 7C8114E1 | 00 | DB 00 | Backup |
| 7C8114E2 | 00 | DB 00 | Copy |
| 7C8114E3 | 00 | DB 00 | Binary |
| 7C8114E4 | 90 | NOP | Undo selection Alt+BkSp |
| 7C8114E5 | 90 | NOP | Assemble |
| 7C8114E6 | 90 | NOP | Space |
| 7C8114E7 | 90 | NOP | |
| 7C8114E8 | 90 | NOP | |

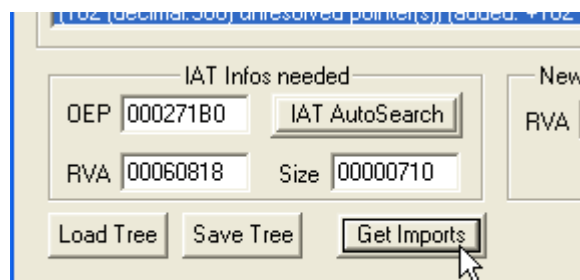
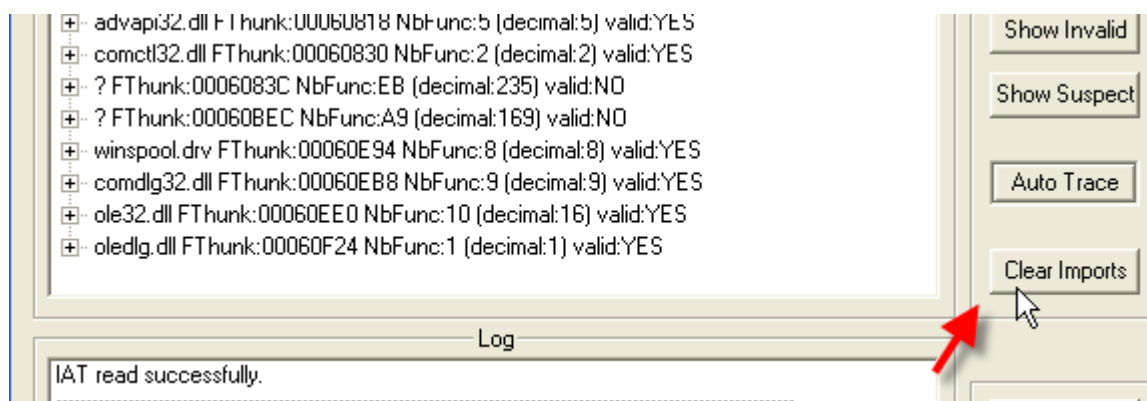
7C8114AB=kernel32.GetVersion

Bueno ya sabemos detectar a mano el nombre de la api, una vez que lo tenemos vamos al IMP REC y hacemos doble click en la entrada mala que estamos reparando y ponemos el nombre bueno.

| | | | |
|--|------|---------------|----------|
| 0042721E | 85C0 | CALL 004271F0 | 00F0071E |
| DS:[00460ADC]=7C8114AB (kernel32.GetVersion) | | TEST FAX.FAX | |

| Address | Hex dump | ASCII |
|----------|---|-----------------|
| 00460ADC | AB 14 81 7C 08 07 9F 00 1A 07 9F 00 2A 07 9F 00 | %u! .f.+*f.*f. |
| 00460AEC | 3B 07 9F 00 5A 07 9F 00 74 07 9F 00 85 07 9F 00 | .f.z.f.t.f.ä.f. |
| 00460AFC | 96 07 9F 00 A7 07 9F 00 B5 07 9F 00 C4 07 9F 00 | ü.f.ö.f.ä.f.-f. |
| 00460B0C | D3 07 9F 00 F3 07 9F 00 01 08 9F 00 24 08 9F 00 | E.f.%f.ö.f.ö.f. |
| 00460B1C | 35 08 9F 00 46 08 9F 00 58 08 9F 00 68 08 9F 00 | ö.f.ö.f.%f.h.f. |

Alli vemos la entrada reparada ahora si en el IMP REC limpiamos todas las entradas apretando el boton CLEAR IMPORTS y de nuevo apretamos GET IMPORTS veremos que ahora la entrada esta tomada como valida.

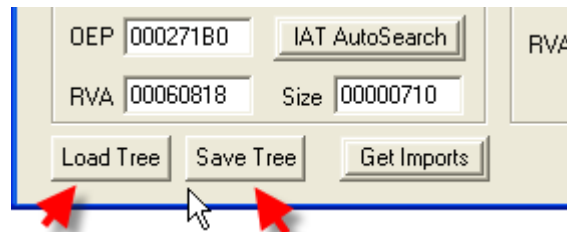


| | |
|--------------|---|
| rva:00060AD0 | ptr:009F06B5 |
| rva:00060AD4 | ptr:009F06C3 |
| rva:00060AD8 | ptr:009F06E6 |
| rva:00060ADC | mod:kernel32.dll ord:01DB name:GetVersion |
| rva:00060AE0 | ptr:009F0708 |
| rva:00060AE4 | ptr:009F071A |

Como vemos es similar colocar la direccion correcta de la api en la entrada correspondiente directamente en la IAT en el OLLYDBG o en el IMP REC colocar el nombre de la API en dicha entrada.

Bueno si hacemos eso en cada entrada redireccionada, y las reparamos una a una, y luego arreglamos el dumpeado, funcionara, el tema es que es un metodo muy tedioso, aunque es bueno conocerlo para verificar alguna entrada a mano si esta dudosa.

Algo que ayuda a los fanaticos de este metodo manual es que el IMP REC permite guardar la tabla con las reparaciones que le hayamos hecho con el boton SAVE TREE y luego si interrumpimos el trabajo podemos volver al OEP del programa, abrir el IMP REC y una vez puestos los datos del OEP; RVA Y SIZE, cargar la

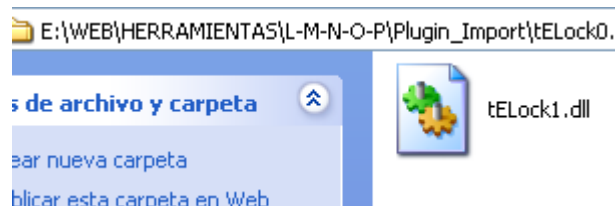


IAT que habiamos guardado en el estado que la dejamos con el boton LOAD TREE.

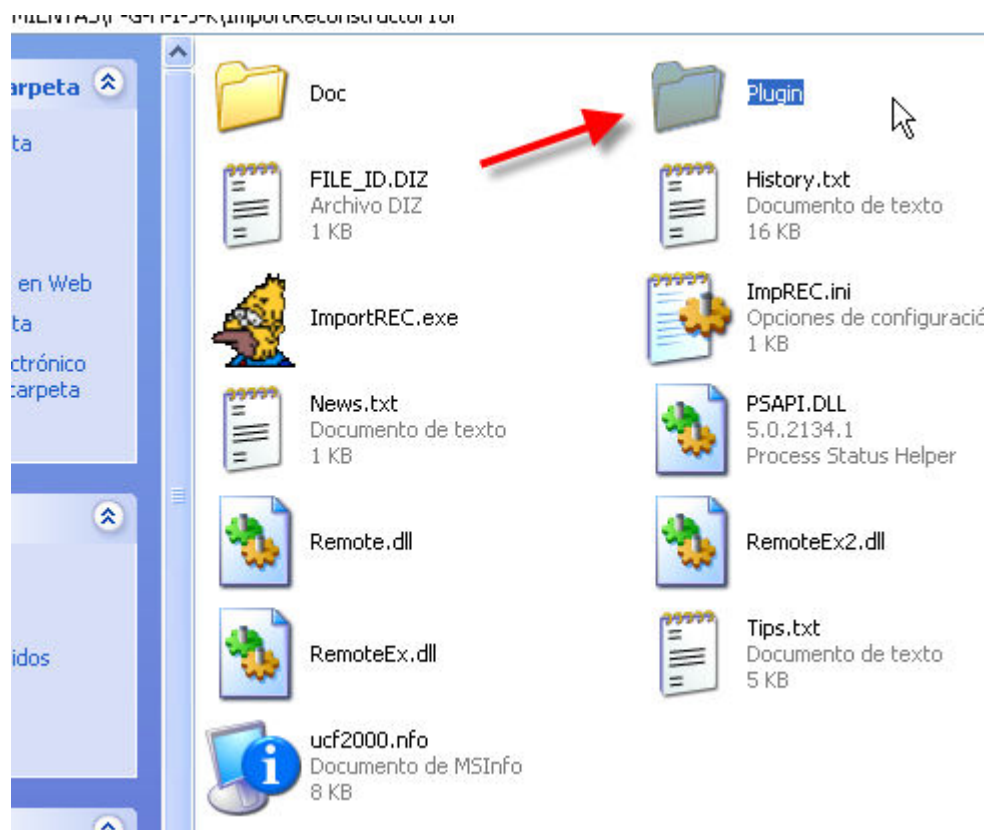
Otra forma posible de reparar las entradas redireccionadas es usando plugins del IMP REC, en este caso si vamos a la carpeta

http://www.ricnar456.dyndns.org/HERRAMIENTAS/L-M-N-O-P/Plugin_Import/

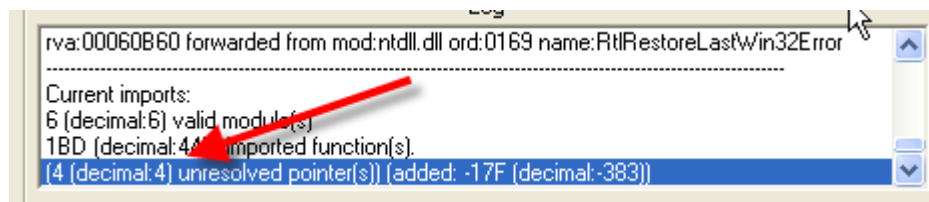
Vemos que hay un plugin para telock el cual puede ser copiado a la carpeta plugins del IMP REC veamos si funciona este metodo.



Copiamos la dll a la carpeta PLUGIN del IMP REC.

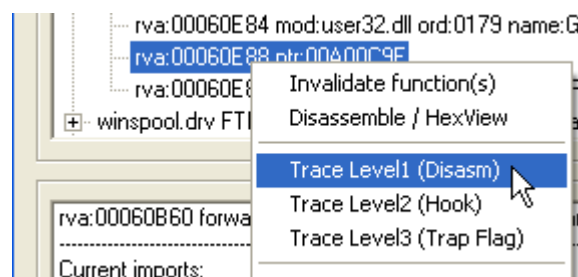


Ahora deberemos reiniciar el IMP REC y una vez que volvemos a ingresar todos los datos, apretamos SHOW INVALID, hacemos click derecho PLUGIN TRACERS y buscamos el del telock, luego de tracear vemos que arregla todas las entradas menos 4, por lo cual vemos la importancia de la reparacion manual ya que el plugin dejo pocas entradas para reparar y estas pueden hacerse a mano.

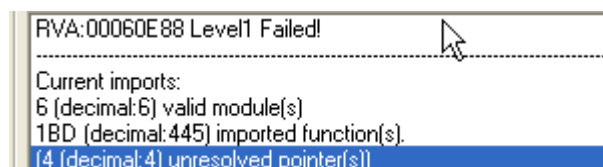


Alli vemos que el IMP REC nos avisa que quedan 4 por reparar, las cuales podemos tracear a mano con el metodo que vimos al inicio, de cualquier manera, el metodo del plugin es muy limitado ya que dependemos que exista un plugin para el packer que estamos usando lo cual es dificil.

El IMP REC posee otros trazadores genericos que en vez de usar los del plugin, se puede intentar a ver si funciona, aunque recomiendo guardar lo reparado hasta ahora, porque la mayoría de las veces terminan en cuelgues del IMP REC.



Marcando una entrada invalida y haciendo CLICK DERECHO vemos tres niveles de traceo, probemos a ver que pasa primero con el de NIVEL 1.



Fallo, probemos con los otros dos, ya con el segundo se colgo el IMP REC, por lo cual a pesar de mencionar estos metodos para otros casos, aquí no funcionan, y generalmente terminan en cuelgues solo sirven para packers muy sencillos.

El proximo metodo es el del JMP-CALL MAGICO que hicimos popular en crackslatinos en tantos tutes de diferentes packers.

El metodo se basa en tratar de buscar el momento en el que el packer guarda los valores en la IAT, y alli observar un poco y comparar lo que ocurre cuando guarda un valor malo, con lo que ocurre cuando guarda un valor bueno.

Usemos como ejemplo la entrada esta que investigamos de GetVersion que es una entrada redireccionada o mala, reiniciemos el crackme empackado con telock, y busquemos dicha entrada al inicio antes de arrancar el programa.

| 00466C11 0000 1ADD BYTE PTR DS:FEF71A7 | | |
|--|---|-------------------------------|
| 00465000=UnPackMe.00465000 | | |
| Address | Hex dump | ASCII |
| 00460ADC | 70 08 83 3D F9 F2 67 B0 9B 85 EB 16 2C 9B 64 A2 | p a = - g a u . d o |
| 00460AEC | 92 38 60 04 80 F4 74 C9 77 C7 C3 A5 A9 F1 32 42 | 8 ' e t f w t n 2 B |
| 00460AFC | 89 DC B4 7A 7C C9 6E CA 89 9E AA F2 A2 89 81 | e l f n x = o e u |
| 00460B0C | 2B 99 E9 58 60 42 69 A7 E1 3B 2E C9 7A BF 0D F4 | +00S' B i p i . f e j . i |
| 00460B1C | 0D F2 1C 6A FC 98 50 6A 0C 02 03 91 0C BA 65 96 | ! = k ? u P f _ F F # . l e u |

Alli esta en el DUMP la direccion 460ADC que al correr el crackme y antes de llegar al OEP, en algun momento se escribira alli el valor malo de la entrada redireccionada que en mi maquina era 9F06F7, quiere decir que en algun momento el packer escribira ese valor en la entrada.

| Address | Hex dump | ASCII |
|----------|---|---------------------------------|
| 00460ADC | F7 06 9F 00 08 07 9F 00 1A 07 9F 00 2A 07 9F 00 | . f . f . + . f . * . f . |
| 00460AEC | 38 07 9F 00 5A 07 9F 00 74 07 9F 00 85 07 9F 00 | . f . 2 . f . t . f . a . f . |
| 00460AFC | 96 07 9F 00 A7 07 9F 00 B5 07 9F 00 C4 07 9F 00 | u . f . 9 . f . a . f . - . f . |
| 00460B0C | 03 07 9F 00 F3 07 9F 00 01 08 9F 00 24 08 9F 00 | e . f . % . f . 0 f . s f . |
| 00460B1C | 35 08 9F 00 46 08 9F 00 58 08 9F 00 68 08 9F 00 | 5 f . F f . X f . h f . |
| 00460B2C | 79 08 9F 00 98 08 9F 00 B2 08 9F 00 C3 08 9F 00 | y f . y f . f f . h f . |
| 00460B3C | 04 08 9F 00 E5 08 9F 00 F3 08 9F 00 02 09 9F 00 | e f . 8 f . % f . 0 f . |

Para interceptar el moemnto en que escribe dicho valor generalmente se coloca al inicio un HARDWARE BPX ON WRITE en la entrada, pero como este packer detecta los HARDWARE BPX, colocaremos un MEMORY BREAKPOINT ON WRITE, para que pare al escribir el valor malo.

| Address | Hex dump | ASCII |
|----------|---|------------------------|
| 00460ADC | 70 08 83 30 F9 F2 62 B0 9B 85 FR 16 2C 9B 64 A2 | 00 33 =-g#au.,dó |
| 00460AEC | 92 38 60 0 | ES'ECItfwAt0t2B |
| 00460AFC | 89 DC B4 0 | 89 81 8-102:1fn#x=0eu |
| 00460B0C | 2B 99 E9 5 | +0US'Bi0B;.f27.7 |
| 00460B1C | 00 F2 1C 6 | !-Lk*0PfEE#.lleu |
| 00460B2C | B0 12 AB 3 | 00 33 K.9830V8+.re |
| 00460B3C | 7B B1 74 3 | 00 EF 00 0.1.=->8e-y8' |
| 00460B4C | 55 F5 B5 4 | 94 15 USAL+87uAF-J=08 |
| 00460B5C | 58 B0 B2 0 | |
| 00460B6C | 20 11 04 0 | |
| 00460B7C | 91 42 14 4 | |
| 00460B8C | 53 2D 84 3 | |
| 00460B9C | B9 C7 A0 7 | |
| 00460BAC | ED 23 61 8 | |

Al dar RUN la primera vez que para por MEMORY BREAKPOINT es aquí

| | | |
|----------|---------------|------------------------|
| 00465D91 | AA | STOS BYTE PTR ES:[EDI] |
| 00465D92 | 69D2 A5B0CD4B | IMUL EDX,EDX,4BCDB0A5 |
| 00465D98 | F9 | STC |
| 00465D99 | 72 02 | JB SHORT 00465D9D |
| 00465D9B | CD 20 | INT 20 |
| 00465D9D | D1C2 | ROL EDX,1 |
| 00465D9F | 69DB 701FFEF0 | TMII FRX FRX 60FE1F70 |

Vemos que no es el lugar buscado porque al ejecutar la linea con F8 no guarda el valor malo en la entrada, asi que damos RUN nuevamente.

La proxima vez que para es aquí

| | | |
|----------|---------------|--------------------------|
| 00465FC2 | 8807 | MOV BYTE PTR DS:[EDI],AL |
| 00465FC4 | D2C8 | ROR AL,CL |
| 00465FC6 | 32C3 | XOR AL,BL |
| 00465FC8 | 021F | ADD BL,BYTE PTR DS:[EDI] |
| 00465FCA | 12D9 | ADC BL,CL |
| 00465FCC | F6C1 01 | TEST CL,1 |
| 00465FCF | 75 0F | JNZ SHORT 00465FE0 |
| 00465FD1 | D1EB | SHR EBX,1 |
| 00465FD3 | F702 00000000 | TEST FRX,0 |

Vemos que tampoco guarda el valor malo en la entrada damos RUN de nuevo.

| | | |
|----------|--------|----------------------------|
| 00465F0B | D3C3 | RUL EBX,CL |
| 00465F0D | 8D1CDB | LEA EBX,DWORD PTR DS:[EBX] |
| 00465FE0 | AA | STOS BYTE PTR ES:[EDI] |
| 00465FE1 | 49 | DEC ECX |
| 00465FE2 | 7F D5 | JG SHORT 00465FB7 |
| 00465FE4 | 5F | POP EDI |
| 00465FE5 | 5E | POP ESI |

Tampoco luego de pasarla con F8 la entrada sigue con otros valores.

| | | |
|----------|-------|--|
| 00466128 | F3:A4 | REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI] |
| 0046612A | 5E | POP ESI |
| 0046612B | EB 8E | JMP SHORT 004660BB |
| 0046612D | 02D2 | ADD DL,DL |
| 0046612F | 75 05 | JNZ SHORT 00466136 |

Seguimos buscando el punto donde guarda el valor malo, vemos que va parando pero nunca guarda el valor buscado.

| Address | Hex dump | ASCII |
|----------|---|------------|
| 00460ADC | A0 12 06 00 96 12 06 00 82 12 06 00 6E 12 06 00 | äü.é.n. |
| 00460AEC | 60 12 06 00 54 12 06 00 3A 12 06 00 2A 12 06 00 | 'ä.T.:.ä. |
| 00460AFC | 1A 12 06 00 02 12 06 00 EE 11 06 00 08 11 06 00 | +ä.ä.~ä.i. |
| 00460B0C | C8 11 06 00 B4 11 06 00 AC 11 06 00 8E 11 06 00 | ü.ä.ä.ä. |
| 00460B1C | 70 11 06 00 52 11 06 00 36 11 06 00 24 11 06 00 | p.R.ä.ä.ä. |
| 00460B2C | 12 11 06 00 FE 10 06 00 EA 10 06 00 D0 10 06 00 | ä.ä.ü.ä. |
| 00460B3C | C0 10 06 00 AE 10 06 00 A0 10 06 00 90 10 06 00 | ü.ä.ä.ä. |
| 00460B4C | 80 10 06 00 72 10 06 00 62 10 06 00 50 10 06 00 | C.r.ä.b.P. |
| 00460B5C | 40 10 06 00 30 10 06 00 24 10 06 00 16 10 06 00 | ä.ä.ä.ä. |
| 00460B6C | 08 10 06 00 F8 0F 06 00 EC 0F 06 00 DC 0F 06 00 | ä.ä.ä.ä. |
| 00460B7C | CC 0F 06 00 BC 0F 06 00 AE 0F 06 00 A0 0F 06 00 | f.ä.ä.ä. |

Al pasar el ultimo REP MOVs la IAT quedo con estos valores y aun no tiene el valor malo definitivo, sigamos dando RUN.

| | | |
|----------|---------------|---------------------------------------|
| 004664E5 | 8908 | MOV DWORD PTR DS:[EAX],ECX |
| 004664E7 | 58 | POP EAX |
| 004664E8 | 83C0 09 | ADD EAX,9 |
| 004664EB | 0185 56D44000 | ADD DWORD PTR SS:[EBP+40D456],EAX |
| 004664F1 | EB 08 | JMP SHORT 004664FB |
| 004664F3 | 838D 52D44000 | OR DWORD PTR SS:[EBP+40D452],FFFFFFFF |
| 004664FA | 61 | POPAD |
| 004664FB | 03BD 4ED34000 | ADD EDI,DWORD PTR SS:[EBP+40D34E] |
| 00466501 | 85DB | TEST EBX,EBX |
| 00466503 | 0F84 C7000000 | JE 004665D0 |

Hasta que llega aquí que vemos que ECX tiene el valor malo y lo guardara en la entrada.

| Registers (FPU) | |
|-----------------|-----------------------------|
| EAX | 00460ADC UnPackMe.00460ADC |
| ECX | 009F06F7 |
| EDX | 00400000 UnPackMe.00400000 |
| EBX | 000612A0 |
| ESP | 0012FFA0 |
| EBP | 0005995E |
| ESI | 00460014 UnPackMe.00460014 |
| EDI | 009F166C |
| EIP | 004664E5 UnPackMe.004664E5 |
| C 0 | ES 0023 32bit 0(FFFFFFFF) |
| P 0 | CS 001B 32bit 0(FFFFFFFF) |
| A 0 | SS 0023 32bit 0(FFFFFFFF) |
| Z 0 | DS 0023 32bit 0(FFFFFFFF) |
| S 0 | FS 003B 32bit 7FFDF000(FFF) |
| T 0 | GS 0000 NULL |

ECX es 9F06F7 y lo guardara en el contenido de EAX o sea en 460ADC en la entrada que estamos investigando, este es el punto donde queriamos llegar.

| Address | Hex dump | ASCII |
|----------|---|------------|
| 00460ADC | F7 06 9F 00 96 12 06 00 82 12 06 00 6E 12 06 00 | äü.é.n. |
| 00460AEC | 60 12 06 00 54 12 06 00 3A 12 06 00 2A 12 06 00 | 'ä.T.:.ä. |
| 00460AFC | 1A 12 06 00 02 12 06 00 EE 11 06 00 08 11 06 00 | +ä.ä.~ä.i. |
| 00460B0C | C8 11 06 00 B4 11 06 00 AC 11 06 00 8E 11 06 00 | ü.ä.ä.ä. |

Vemos que al apretar F8 guardo el valor malo, este es el primer paso que debemos hacer cuando utilizamos este metodo, ahora viene la parte mas trabajosa que es hallar el salto magico.

Como vemos al llegar a ese JE un poco mas abajo vemos en los registros, el nombre de la api que corresponde a esta entrada que acaba de llenar con el valor malo o sea GetVersion.

Luego vemos que llega a una llamada a GetProcAddress para encontrar la direccion de GetVersion en mi maquina ya que los parametros son.

Yo entre en la api para ver los parametros pero no es necesario, pasamos con f8 el CALL.

Y en EAX nos devuelve la direccion de GetVersion en nuestra maquina sigamos traceando.

Alli llegamos a un salto, el cual saltara veamos que pasa sin tracearlo, haciendo click en FOLLOW

| | | | |
|----------|---------------|---------------------------------|-------------------|
| 004665B1 | 8907 | MOV DWORD PTR DS:[EDI],EAX | |
| 004665B3 | 58 | POP EAX | |
| 004665B4 | 48 | DEC EAX | |
| 004665B5 | 74 0D | JE SHORT 004665C4 | UnPackMe.004665C4 |
| 004665B7 | 40 | INC EAX | |
| 004665B8 | F8 | CLC | |
| 004665B9 | 66:8943 FE | MOV WORD PTR DS:[EBX-2],AX | |
| 004665BD | 8803 | MOV BYTE PTR DS:[EBX],AL | |
| 004665BF | 43 | INC EBX | |
| 004665C0 | 3803 | CMP BYTE PTR DS:[EBX],AL | |
| 004665C2 | 75 F9 | JNZ SHORT 004665BD | UnPackMe.004665BD |
| 004665C4 | 8385 4ED34000 | ADD DWORD PTR SS:[EBP+40D34E],4 | |
| 004665C8 | E9 BAFDFFFF | JMP 0046638A | UnPackMe.0046638A |

Vemos que alli guardara la direccion pero en otro lugar no en la IAT ya que EDI vale

| Registers (FPU) | | |
|-----------------|----------|---------------------|
| EAX | 7C8114AB | kernel32.GetVersion |
| ECX | 7C929AEB | ntdll.7C929AEB |
| EDX | 7C98C0D8 | ntdll.7C98C0D8 |
| EBX | 004612A2 | ASCII "GetVersion" |
| ESP | 0012FFA0 | |
| EBP | 0005995E | |
| ESI | 00460014 | UnPackMe.00460014 |
| EDI | 009F17D4 | |
| EIP | 0046657C | UnPackMe.0046657C |

luego de guardar esos valores llega a este JMP

| | | | |
|----------|---------------|-----------------------------------|--|
| 004665B1 | 8907 | MOV DWORD PTR DS:[EDI],EAX | |
| 004665B3 | 58 | POP EAX | |
| 004665B4 | 48 | DEC EAX | |
| 004665B5 | 74 0D | JE SHORT 004665C4 | |
| 004665B7 | 40 | INC EAX | |
| 004665B8 | F8 | CLC | |
| 004665B9 | 66:8943 FE | MOV WORD PTR DS:[EBX-2],AX | |
| 004665BD | 8803 | MOV BYTE PTR DS:[EBX],AL | |
| 004665BF | 43 | INC EBX | |
| 004665C0 | 3803 | CMP BYTE PTR DS:[EBX],AL | |
| 004665C2 | 75 F9 | JNZ SHORT 004665BD | |
| 004665C4 | 8385 4ED34000 | ADD DWORD PTR SS:[EBP+40D34E],4 | |
| 004665C8 | E9 BAFDFFFF | JMP 0046638A | |
| 004665D0 | 83C6 14 | ADD ESI,14 | |
| 004665D3 | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 004665D9 | E9 48FCFFFF | JMP 00466226 | |
| 004665DE | 61 | POPAD | |
| 004665DF | C3 | RETN | |

Que nos devuelve al inicio del proceso porque al retornar del mismo ya el valor de EAX es machacado.

Veamos con FOLLOW donde iria

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 0046638A | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 00466390 | 8B06 | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | TEST EAX,EAX | |
| 00466394 | 75 0B | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | TEST EAX,EAX | |
| 0046639B | 0F84 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | ADD EAX,EDX | |
| 004663A3 | 0385 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34E] | |
| 004663A9 | 8B18 | MOV EBX,DWORD PTR DS:[EAX] | |
| 004663AB | F7C3 00000000 | TEST EBX,80000000 | |
| 004663B1 | 74 06 | JE SHORT 004663B9 | UnPackMe.004663B9 |

Alli se ve claramente que EAX en la segunda linea perdiera el valor correcto de la api en mi maquina y seguro pasara a repetir el ciclo para la siguiente entrada asi que debemos estar cerca. Podemos fijarnos para entradas malas cuales son los saltos condicionales que saltan, para luego cuando traceemos para entradas buenas podemos comparar.

Mucha veces yo me hago una tablita que suele ayudar mucho y es que en la rutina de este loop anoto que

ocurre con los saltos tanto con entradas buenas como con entradas malas.

PARA ENTRADAS MALAS (pondre imagenes de todos los saltos a medida que paso por ellos traceando, hasta llegar a donde guarda el valor)

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 0046638A | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 00466390 | 8B06 | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | TEST EAX,EAX | |
| 00466394 | 75 0B | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | TEST EAX,EAX | |
| 0046639B | 0F84 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | ADD EAX,EDX | |
| 004663A3 | 0385 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34E] | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 0046638A | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 00466390 | 8B06 | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | TEST EAX,EAX | |
| 00466394 | 75 0B | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | TEST EAX,EAX | |
| 0046639B | 0F84 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | ADD EAX,EDX | |
| 004663A3 | 0385 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34E] | |
| 004663A9 | 8B18 | MOV EBX,DWORD PTR DS:[EAX] | |
| 004663AB | F7C3 00000000 | TEST EBX,00000000 | |
| 004663B1 | 74 06 | JE SHORT 004663B9 | UnPackMe.004663B9 |
| 004663B3 | 8120 00000000 | AND DWORD PTR DS:[EAX],00000000 | |
| 004663B9 | 8B7E 10 | MOV EDI,DWORD PTR DS:[ESI+10] | |
| 004663BC | 03FA | ADD EDI,EDX | |
| 004663BE | 80A5 D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 30010000 | JE 004664FB | UnPackMe.004664FB |
| 004663CB | 80A5 D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |
| 004663E5 | 0F84 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 40 | DEC EAX | |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | PUSHAD | |
| 004663F3 | 8BF7 | MOV FSI,FDI | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 0046638A | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 00466390 | 8B06 | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | TEST EAX,EAX | |
| 00466394 | 75 0B | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | TEST EAX,EAX | |
| 0046639B | 0F84 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | ADD EAX,EDX | |
| 004663A3 | 0385 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34E] | |
| 004663A9 | 8B18 | MOV EBX,DWORD PTR DS:[EAX] | |
| 004663AB | F7C3 00000000 | TEST EBX,00000000 | |
| 004663B1 | 74 06 | JE SHORT 004663B9 | UnPackMe.004663B9 |
| 004663B3 | 8120 00000000 | AND DWORD PTR DS:[EAX],00000000 | |
| 004663B9 | 8B7E 10 | MOV EDI,DWORD PTR DS:[ESI+10] | |
| 004663BC | 03FA | ADD EDI,EDX | |
| 004663BE | 80A5 D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 30010000 | JE 004664FB | UnPackMe.004664FB |
| 004663CB | 80A5 D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004663BC | 03FA | ADD EDI,EDX | |
| 004663BE | 80A5 D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 30010000 | JE 004664FB | UnPackMe.004664FB |
| 004663CB | 80A5 D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |
| 004663E5 | 0F84 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 48 | DEC EAX | |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | PUSHAD | |
| 004663F3 | 8BF7 | MOV ESI,EDI | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |
| 004663E5 | 0F84 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 48 | DEC EAX | |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | PUSHAD | |
| 004663F3 | 8BF7 | MOV ESI,EDI | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004664A4 | 8D85 14BB4000 | LEA EAX,DWORD PTR SS:[EBP+40BB14] | |
| 004664AA | FFB5 FBCA4000 | PUSH DWORD PTR SS:[EBP+40CAFB] | |
| 004664B0 | 0FB608 | MOVZX ECX,BYTE PTR DS:[EAX] | |
| 004664B3 | FF0C24 | DEC DWORD PTR SS:[ESP] | |
| 004664B6 | 7E 05 | JLE SHORT 004664B0 | UnPackMe.004664B0 |
| 004664B8 | 40 | INC EAX | |
| 004664B9 | 03C1 | ADD EAX,ECX | |
| 004664BB | EB F3 | JMP SHORT 004664B0 | UnPackMe.004664B0 |
| 004664BD | 890C24 | MOV DWORD PTR SS:[ESP],ECX | |
| 004664C0 | FFB5 FBCA4000 | INC DWORD PTR SS:[EBP+40CAFB] | |

Esto es un miniloop luego de salir de el

| | | | |
|----------|-----------------|-----------------------------------|-------------------|
| 004664FB | 03BD 4ED34000 | ADD EDI,DWORD PTR SS:[EBP+40D34E] | |
| 00466501 | 85DB | TEST EBX,EBX | |
| 00466503 | 0F84 C7000000 | JE 004665D0 | UnPackMe.004665D0 |
| 00466509 | F7C3 00000000 | TEST EBX,80000000 | |
| 0046650F | 6A 00 | PUSH 0 | |
| 00466511 | 75 0F | JNZ SHORT 00466522 | UnPackMe.00466522 |
| 00466513 | 8D5C13 02 | LEA EBX,DWORD PTR DS:[EBX+EDX+2] | |
| 00466517 | 803B 00 | CMP BYTE PTR DS:[EBX],0 | |
| 0046651A | 0F84 93000000 | JE 004665B3 | UnPackMe.004665B3 |
| 00466520 | EB 45 | JMP SHORT 00466567 | UnPackMe.00466567 |
| 00466522 | FF0424 | INC DWORD PTR SS:[ESP] | |
| 00466525 | 66:85DB | TEST BX,BX | |
| 00466528 | 0F84 85000000 | JE 004665B3 | UnPackMe.004665B3 |
| 0046652E | 8B85 4AD34000 | MOV EAX,DWORD PTR SS:[EBP+40D34A] | |
| 00466534 | 3B85 42D44000 | CMP EAX,DWORD PTR SS:[EBP+40D442] | |
| 0046653A | 75 2B | JNZ SHORT 00466567 | UnPackMe.00466567 |
| 0046653C | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |
| 00466542 | 8BD3 | MOV EDX,EBX | |
| 00466544 | 8D1495 FCFFFFFF | LEA EDX,DWORD PTR DS:[EDX*4-4] | |
| 0046654B | 8B9D 4AD34000 | MOV EBX,DWORD PTR SS:[EBP+40D34A] | |
| 00466551 | 8B43 3C | MOV EAX,DWORD PTR DS:[EBX+3C] | |
| 00466554 | 8B4418 78 | MOV EAX,DWORD PTR DS:[EAX+EBX+78] | |
| 00466558 | 035C18 1C | ADD EBX,DWORD PTR DS:[EAX+EBX+1C] | |
| 0046655C | 8B041A | MOV EAX,DWORD PTR DS:[EDX+EBX] | |
| 0046655F | 0385 4AD34000 | ADD EAX,DWORD PTR SS:[EBP+40D34A] | |
| 00466565 | EB 13 | JMP SHORT 0046657A | UnPackMe.0046657A |
| 00466567 | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |
| 0046656D | 53 | PUSH EBX | |
| 0046656E | FFB5 4AD34000 | PUSH DWORD PTR SS:[EBP+40D34A] | |
| 00466574 | FF95 E0BA4000 | CALL DWORD PTR SS:[EBP+40BAE0] | |
| 0046657D | 40 | TNC EBX | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004664FB | 03BD 4ED34000 | ADD EDI,DWORD PTR SS:[EBP+40D34E] | |
| 00466501 | 85DB | TEST EBX,EBX | |
| 00466503 | 0F84 C7000000 | JE 004665D0 | UnPackMe.004665D0 |
| 00466509 | F7C3 00000000 | TEST EBX,80000000 | |
| 0046650F | 6A 00 | PUSH 0 | |
| 00466511 | 75 0F | JNZ SHORT 00466522 | UnPackMe.00466522 |
| 00466513 | 8D5C13 02 | LEA EBX,DWORD PTR DS:[EBX+EDX+2] | |
| 00466517 | 803B 00 | CMP BYTE PTR DS:[EBX],0 | |
| 0046651A | 0F84 93000000 | JE 004665B3 | UnPackMe.004665B3 |
| 00466520 | EB 45 | JMP SHORT 00466567 | UnPackMe.00466567 |
| 00466522 | FF0424 | INC DWORD PTR SS:[ESP] | |
| 00466525 | 66:85DB | TEST BX,BX | |

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004664FB | 03B0 4ED34000 | ADD EDI,DWORD PTR SS:[EBP+40D34E] | |
| 00466501 | 85D8 | TEST EBX,EBX | |
| 00466503 | 0F84 C7000000 | JE 004665D0 | UnPackMe.004665D0 |
| 00466509 | F7C3 00000000 | TEST EBX,00000000 | |
| 0046650F | 6A 00 | PUSH 0 | |
| 00466511 | 75 0F | JNZ SHORT 00466522 | UnPackMe.00466522 |
| 00466513 | 8D5C13 02 | LEA EBX,DWORD PTR DS:[EBX+EDX+2] | |
| 00466517 | 803B 00 | CMP BYTE PTR DS:[EBX],0 | |
| 00466519 | 0F84 93000000 | JE 004665B3 | UnPackMe.004665B3 |
| 00466520 | EB 45 | JMP SHORT 00466567 | UnPackMe.00466567 |
| 00466522 | FF0424 | INC DWORD PTR SS:[ESP] | |
| 00466525 | 66:85D8 | TEST BX,BX | |
| 00466528 | 0F84 85000000 | JE 004665B3 | UnPackMe.004665B3 |
| 0046652E | 8B85 4AD34000 | MOV EAX,DWORD PTR SS:[EBP+40D34A] | |
| 00466534 | 3B85 42D44000 | CMP EAX,DWORD PTR SS:[EBP+40D442] | |
| 0046653A | 75 2B | JNZ SHORT 00466567 | UnPackMe.00466567 |
| 0046653C | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |

| | | | |
|----------|-----------------|-----------------------------------|-------------------------|
| 0046651A | 0F84 93000000 | JE 004665B3 | UnPackMe.004665B3 |
| 00466520 | EB 45 | JMP SHORT 00466567 | UnPackMe.00466567 |
| 00466522 | FF0424 | INC DWORD PTR SS:[ESP] | |
| 00466525 | 66:85D8 | TEST BX,BX | |
| 00466528 | 0F84 85000000 | JE 004665B3 | UnPackMe.004665B3 |
| 0046652E | 8B85 4AD34000 | MOV EAX,DWORD PTR SS:[EBP+40D34A] | |
| 00466534 | 3B85 42D44000 | CMP EAX,DWORD PTR SS:[EBP+40D442] | |
| 0046653A | 75 2B | JNZ SHORT 00466567 | UnPackMe.00466567 |
| 0046653C | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |
| 00466542 | 8BD3 | MOV EDX,EBX | |
| 00466544 | 8D1495 FCFFFFFF | LEA EDX,DWORD PTR DS:[EDX*4-4] | |
| 00466548 | 8B9D 4AD34000 | MOV EBX,DWORD PTR SS:[EBP+40D34A] | |
| 00466551 | 8B43 3C | MOV EAX,DWORD PTR DS:[EBX+3C] | |
| 00466554 | 8B4418 78 | MOV EAX,DWORD PTR DS:[EAX+EBX+78] | |
| 00466558 | 035C18 1C | ADD EBX,DWORD PTR DS:[EAX+EBX+1C] | |
| 0046655C | 8B041A | MOV EAX,DWORD PTR DS:[EDX+EBX] | |
| 0046655F | 0385 4AD34000 | ADD EAX,DWORD PTR SS:[EBP+40D34A] | |
| 00466565 | EB 13 | JMP SHORT 0046657A | UnPackMe.0046657A |
| 00466567 | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |
| 0046656D | 53 | PUSH EBX | |
| 0046656E | FFB5 4AD34000 | PUSH DWORD PTR SS:[EBP+40D34A] | |
| 00466574 | FF95 E0BA4000 | CALL DWORD PTR SS:[EBP+40BAE0] | kernel32.GetProcAddress |
| 0046657A | 40 | INC EAX | |
| 0046657B | 48 | DEC EAX | |
| 0046657C | 75 33 | JNZ SHORT 004665B1 | UnPackMe.004665B1 |
| 0046657E | 58 | POP EAX | |
| 0046657F | F9 | STC | |
| 00466580 | 0F82 61FDFFFF | JB 004665E7 | UnPackMe.004665E7 |
| 00466586 | 47 | INC EDI | |

Alli ya llega a GetProcAddress y luego salta donde lo guarda

| | | | |
|----------|---------------|--------------------------------|-------------------|
| 00466567 | 81E3 FFFFFFFF | AND EBX,7FFFFFFF | |
| 0046656D | 53 | PUSH EBX | |
| 0046656E | FFB5 4AD34000 | PUSH DWORD PTR SS:[EBP+40D34A] | |
| 00466574 | FF95 E0BA4000 | CALL DWORD PTR SS:[EBP+40BAE0] | |
| 0046657A | 40 | INC EAX | |
| 0046657B | 48 | DEC EAX | |
| 0046657C | 75 33 | JNZ SHORT 004665B1 | UnPackMe.004665B1 |
| 0046657E | 58 | POP EAX | |
| 0046657F | F9 | STC | |
| 00466580 | 0F82 61FDFFFF | JB 004665E7 | UnPackMe.004665E7 |
| 00466586 | 47 | INC EDI | |

| | | | |
|----------|---------------|---------------------------------|---------------------------|
| 004665B1 | 8907 | MOV DWORD PTR DS:[EDI],EAX | kernel32.CreateDirectoryA |
| 004665B3 | 58 | POP EAX | |
| 004665B4 | 48 | DEC EAX | |
| 004665B5 | 74 0D | JE SHORT 004665C4 | UnPackMe.004665C4 |
| 004665B7 | 40 | INC EAX | |
| 004665B8 | F8 | CLC | |
| 004665B9 | 66:8943 FE | MOV WORD PTR DS:[EBX-2],AX | |
| 004665BD | 8803 | MOV BYTE PTR DS:[EBX],AL | |
| 004665BF | 43 | INC EBX | |
| 004665C0 | 3B03 | CMP BYTE PTR DS:[EBX],AL | |
| 004665C2 | 75 F9 | JNZ SHORT 004665D0 | UnPackMe.004665D0 |
| 004665C4 | 8385 4ED34000 | ADD DWORD PTR SS:[EBP+40D34E],4 | |
| 004665C8 | E9 BAFDFFFF | JMP 0046638A | UnPackMe.0046638A |
| 004665D0 | 83C6 14 | ADD ESI,14 | |

Alli lo guarda este es un traceo completo para una entrada mala, mirando los saltos condicionales como funcionan en ese caso, ahora llegaremos el OEP y buscaremos una entrada buena para hacer el mismo trabajo y comparar.

Llego al OEP

| Address | Hex dump | ASCII |
|----------|---|--|
| 00460ADC | F7 06 9F 00 08 07 9F 00 1A 07 9F 00 2A 07 9F 00 | ~*f.~*f.+*f.*~*f. |
| 00460AEC | 3B 07 9F 00 5A 07 9F 00 74 07 9F 00 85 07 9F 00 | !~*f.2~*f.t~*f.~*f. |
| 00460AFC | 96 07 9F 00 A7 07 9F 00 B5 07 9F 00 C4 07 9F 00 | ü~*f.0~*f.~*f.~*f.-~*f. |
| 00460B0C | D3 07 9F 00 F3 07 9F 00 01 08 9F 00 24 08 9F 00 | E~*f.~*f.0~*f.~*f.~*f. |
| 00460B1C | 35 08 9F 00 46 08 9F 00 58 08 9F 00 68 08 9F 00 | 5~*f.F~*f.~*f.~*f.h~*f. |
| 00460B2C | 79 08 9F 00 98 08 9F 00 B2 08 9F 00 C3 08 9F 00 | u~*f.0~*f.~*f.~*f.H~*f. |
| 00460B3C | D4 08 9F 00 E5 08 9F 00 F3 08 9F 00 02 09 9F 00 | E~*f.0~*f.~*f.~*f.0~*f. |
| 00460B4C | 11 09 9F 00 31 09 9F 00 3F 09 9F 00 62 09 9F 00 | ~*f.1~*f.~*f.~*f.b~*f. |
| 00460B5C | 73 09 9F 00 84 09 9F 00 96 09 9F 00 A6 09 9F 00 | s~*f.~*f.~*f.~*f.~*f. |
| 00460B6C | B7 09 9F 00 D6 09 9F 00 F0 09 9F 00 01 0A 9F 00 | A~*f.i~*f.~*f.~*f.0~*f. |
| 00460B7C | 12 0A 9F 00 23 0A 9F 00 31 0A 9F 00 40 0A 9F 00 | +~*f.~*f.1~*f.0~*f. |
| 00460B8C | 4F 0A 9F 00 6F 0A 9F 00 7D 0A 9F 00 A0 0A 9F 00 | O~*f.o~*f.~*f.~*f.~*f. |
| 00460B9C | B1 0A 9F 00 C2 0A 9F 00 D4 0A 9F 00 C0 4B 0F 77 | ~*f.T~*f.E~*f.~*f.~*f. |
| 00460BAC | 3B 4C 0F 77 94 A5 11 77 59 48 0F 77 82 4E 0F 77 | ~*f.L~*f.0~*f.~*f.~*f.~*f. |
| 00460BBC | 3F 50 0F 77 D9 66 0F 77 50 48 0F 77 55 4C 0F 77 | yE~*f.w~*f.P~*f.w~*f.O~*f.P~*f.w~*f.P~*f.w~*f. |
| 00460BDC | C2 4B 0F 77 95 D2 11 77 80 5D 15 77 00 00 00 00 | ?P~*f.w~*f.~*f.w~*f.P~*f.H~*f.w~*f.U~*f.L~*f.w~*f. |
| 00460BEC | 00 00 A7 00 11 00 A7 00 22 00 A7 00 33 00 A7 00 | ..0~*f.~*f.~*f.~*f.~*f.~*f. |
| 00460BFC | 00 00 A0 00 11 00 A0 00 22 00 A0 00 33 00 A0 00 | ..~*f.~*f.~*f.~*f.~*f.~*f. |

Como entrada buena elegimos 460BAC, ahora realizamos el mismo proceso que hicimos con la mala.

Reiniciamos y la buscamos antes de ejecutar el programa.

| 00466C0D | 0000 | ADD BYTE PTR DS:[EAX],AL |
|----------------------------|---|--|
| 00466C0F | 0000 | ADD BYTE PTR DS:[EAX],AL |
| 00466C11 | 0000 | ADD BYTE PTR DS:[EAX],AL |
| 00465000=UnPackMe.00465000 | | |
| Address | Hex dump | ASCII |
| 00460BAC | ED 23 61 8C BB 96 AD 21 5F 45 86 73 E3 8D 73 81 | Y#a~*f.ü~*f.~*f.~*f.~*f.~*f. |
| 00460BBC | 82 0C 87 10 C9 6C E2 B3 BB 7D E9 F4 9F D8 C3 D7 | 6~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f. |
| 00460BCC | 81 03 46 64 8F DB CB 4A BC 6F 01 BF CC EC 37 C8 | ü~*f.FdA~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f. |
| 00460BDC | 86 B3 6B 2F A9 6B A3 2B 1A 47 19 6F 9C C1 62 DF | ~*f.k~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f. |
| 00460BEC | EE 0E 92 B6 66 EF BE F3 33 D3 FA 6E 21 B0 49 BC | ~*f.AE~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f.~*f. |

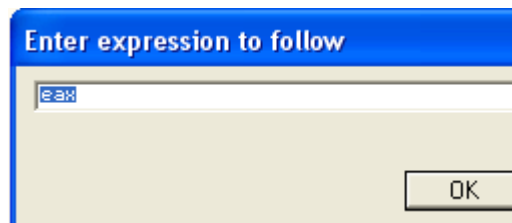
Ahora ponemos un MEMORY BREAKPOINT ON WRITE para tratar de interceptar el momento en que el packer guarda la entrada buena.

| Address | Hex dump | Instruction | Comment |
|----------|------------|----------------------------|---------|
| 004665B1 | 8907 | MOV DWORD PTR DS:[EDI],EAX | |
| 004665B3 | 58 | POP EAX | |
| 004665B4 | 48 | DEC EAX | |
| 004665B5 | 74 0D | JE SHORT 004665C4 | |
| 004665B7 | 40 | INC EAX | |
| 004665B8 | F8 | CLC | |
| 004665B9 | 66:8943 FE | MOV WORD PTR DS:[EBX-2],AX | |
| 004665BD | 8803 | MOV BYTE PTR DS:[EBX],AL | |
| 004665BF | 43 | TNC EAX | |

Alli para cuando va a guardar en la entrada el valor bueno.

| Registers (FPU) |
|--------------------------------|
| EAX 770F4C3B |
| ECX 7C929AEB ntdll.7C929AEB |
| EDX 7C98C0D8 ntdll.7C98C0D8 |
| EBX 00000007 |
| ESP 0012FFA0 |
| EBP 0005995E |
| ESI 004600DC UnPackMe.004600DC |
| EDI 00460BAC UnPackMe.00460BAC |
| EIP 004665B1 UnPackMe.004665B1 |
| C 0 ES 0023 32bit 0(FFFFFFFF) |
| P 0 CS 001B 32bit 0(FFFFFFFF) |
| A 0 SS 0023 32bit 0(FFFFFFFF) |

Vemos que EAX tiene el valor de una posible api aunque aun no dice el nombre, pero si miramos con GOTO EXPRESSION -EAX



| | | |
|----------|---------|------------------------------|
| 770F4C39 | 90 | NOP |
| 770F4C3A | 90 | NOP |
| 770F4C3B | 8BFF | MOV EDI,EDI |
| 770F4C3D | 55 | PUSH EBP |
| 770F4C3E | 8BEC | MOV EBP,ESP |
| 770F4C40 | 8B45 08 | MOV EAX,DWORD PTR SS:[EBP+8] |
| 770F4C43 | 85C0 | TEST EAX,EAX |
| 770F4C45 | 74 05 | JE SHORT 770F4C4C |
| 770F4C47 | 8B40 FC | MOV EAX,DWORD PTR DS:[EAX-4] |
| 770F4C4A | 01E8 | SHR EAX,1 |
| 770F4C4C | 5D | POP EBP |
| 770F4C4D | C2 0400 | RETN 4 |
| 770F4C50 | 90 | NOP |
| 770F4C51 | 90 | NOP |

Vemos que nos lleva alli, y al volver al codigo magia aparecio el nombre jeje.

| Registers (FPU) | | |
|-----------------|----------|-----------------------|
| EAX | 770F4C3B | OLEAUT32.SysStringLen |
| ECX | 7C929AEB | ntdll.7C929AEB |
| EDX | 7C98C0D8 | ntdll.7C98C0D8 |
| EBX | 00000007 | |
| ESP | 0012FFA0 | |
| EBP | 0005995E | |
| ESI | 004600DC | UnPackMe.004600DC |
| EDI | 00460BAC | UnPackMe.00460BAC |
| EIP | 004665B1 | UnPackMe.004665B1 |
| C 0 | ES 0023 | 32bit 0(FFFFFFFF) |
| P 0 | CS 001B | 32bit 0(FFFFFFFF) |
| A 0 | SS 0023 | 32bit 0(FFFFFFFF) |

Ahora vamos a hacer el mismo trabajo que hicimos con la entrada mala, esperemos que siga en orden y que la siguiente sea buena tambien si no deberemos seguir hasta que trabaje con una buena.

PARA ENTRADAS BUENAS

Llegamos hasta el inicio del loop y comenzamos a tracear.

| | | | |
|----------|---------------|-------------------------------------|-------------------|
| 0046638A | 8B95 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D34000] | UnPackMe.00400000 |
| 00466390 | 8B06 | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | TEST EAX,EAX | |
| 00466394 | 75 0B | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | TEST EAX,EAX | |
| 0046639B | 0F84 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | ADD EAX,EDX | |
| 004663A3 | 0385 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34000] | |
| 004663A9 | 8B18 | MOV EBX,DWORD PTR DS:[EAX] | |
| 004663AB | F7C3 00000000 | TEST EBX,80000000 | |
| 004663B1 | 74 06 | JE SHORT 004663B9 | UnPackMe.004663B9 |
| 004663B3 | 8120 00000000 | AND DWORD PTR DS:[EAX],80000000 | |
| 004663B9 | 8B7E 10 | MOV EDI,DWORD PTR DS:[ESI+10] | |
| 004663BC | 03FA | ADD EDI,EDX | |
| 004663BE | 80A5 D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 30010000 | JE 004664FB | UnPackMe.004664FB |
| 004663CB | 80A5 D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |
| 004663E5 | 0F84 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 48 | DEC EAX | |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | PUSHAD | |

| | | | | |
|----------|-------|----------|-----------------------------------|-------------------|
| 0046638A | 8B95 | 62D34000 | MOV EDX,DWORD PTR SS:[EBP+40D362] | |
| 00466390 | 8B06 | | MOV EAX,DWORD PTR DS:[ESI] | |
| 00466392 | 85C0 | | TEST EAX,EAX | |
| 00466394 | 75 0B | | JNZ SHORT 004663A1 | UnPackMe.004663A1 |
| 00466396 | 8B46 | 10 | MOV EAX,DWORD PTR DS:[ESI+10] | |
| 00466399 | 85C0 | | TEST EAX,EAX | |
| 0046639B | 0F84 | 46FFFFFF | JE 004662E7 | UnPackMe.004662E7 |
| 004663A1 | 03C2 | | ADD EAX,EDX | |
| 004663A3 | 0385 | 4ED34000 | ADD EAX,DWORD PTR SS:[EBP+40D34E] | |

Este salta igual que en las entradas malas

| | | | | |
|----------|-------|----------|----------------------------------|-------------------|
| 004663A7 | 8B18 | | MOV EBX,DWORD PTR DS:[EAX] | |
| 004663AB | F7C3 | 00000000 | TEST EBX,80000000 | |
| 004663B1 | 74 06 | | JE SHORT 004663B9 | UnPackMe.004663B9 |
| 004663B3 | 8120 | 00000000 | AND DWORD PTR DS:[EAX],80000000 | |
| 004663B9 | 8B7E | 10 | MOV EDI,DWORD PTR DS:[ESI+10] | |
| 004663BC | 03FA | | ADD EDI,EDX | |
| 004663BE | 80A5 | D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |

Este en las entradas malas no saltaba pero es un salto muy pequeño no creo que sea decisivo, sigamos.

| | | | | |
|----------|------|----------|-----------------------------------|--|
| 004663B7 | 8B7E | 10 | MOV EDI,DWORD PTR DS:[ESI+10] | |
| 004663BC | 03FA | | ADD EDI,EDX | |
| 004663BE | 80A5 | D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 | 30010000 | JE 004664FB | |
| 004663CB | 80A5 | D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 | 23010000 | JE 004664FB | |
| 004663D8 | 89BD | 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 | 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |

Ese no salta igual que en las malas

| | | | | |
|----------|------|----------|-----------------------------------|-------------------|
| 004663D7 | 03FA | | ADD EDI,EDX | |
| 004663BC | 80A5 | D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF | |
| 004663C5 | 0F84 | 30010000 | JE 004664FB | UnPackMe.004664FB |
| 004663CB | 80A5 | D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF | |
| 004663D2 | 0F84 | 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD | 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 | 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | | INC EAX | |
| 004663E5 | 0F84 | 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 48 | | DEC EAX | |
| 004663EC | 0F85 | B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | | PUSHAD | |

aquí vemos una gran diferencia este salto en las entradas malas no saltaba y aca si y vemos que el codigo que salta es mucho.

| | | |
|----------|---------------|-----------------------------------|
| 004663BE | 80A5 D6CC4000 | AND BYTE PTR SS:[EBP+40CCD6],0FF |
| 004663C5 | 0F84 30010000 | JE 004664FB |
| 004663CB | 80A5 D7CC4000 | AND BYTE PTR SS:[EBP+40CCD7],0FF |
| 004663D2 | 0F84 23010000 | JE 004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] |
| 004663E4 | 40 | INC EAX |
| 004663E5 | 0F84 10010000 | JE 004664FB |
| 004663EB | 48 | DEC EAX |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 |
| 004663F2 | 60 | PUSHAD |
| 004663F3 | 8BF7 | MOV ESI,EDI |
| 004663F5 | 2BC0 | SUB EAX,EAX |
| 004663F7 | 40 | INC EAX |
| 004663F8 | 833F 00 | CMP DWORD PTR DS:[EDI],0 |
| 004663FB | 8D7F 04 | LEA EDI,DWORD PTR DS:[EDI+4] |
| 004663FE | 75 F7 | JNZ SHORT 004663F7 |
| 00466400 | 48 | DEC EAX |
| 00466401 | 0F84 EC000000 | JE 004664F3 |
| 00466407 | 8B08 | MOV EBX,EAX |
| 00466409 | 6BC0 31 | IMUL EAX,EAX,31 |
| 0046640C | 6A 04 | PUSH 4 |
| 0046640E | 68 00100000 | PUSH 1000 |
| 00466413 | 50 | PUSH EAX |
| 00466414 | 6A 00 | PUSH 0 |
| 00466416 | FF95 ECBA4000 | CALL DWORD PTR SS:[EBP+40BAEC] |
| 0046641C | 85C0 | TEST EAX,EAX |
| 0046641E | 0F84 CF000000 | JE 004664F3 |
| 00466424 | 8BFE | MOV EDI,ESI |
| 00466426 | 8BCB | MOV ECX,EBX |
| 00466428 | 8BF8 | MOV EDI,EAX |
| 0046642A | 8985 56D44000 | MOV DWORD PTR SS:[EBP+40D456],EAX |
| 00466430 | 8BCB | MOV ECX,EBX |
| 00466432 | 6BDB 29 | IMUL EBX,EBX,29 |
| 00466435 | 03DF | ADD EBX,EDI |
| 00466437 | 891C24 | MOV DWORD PTR SS:[ESP],EBX |
| 0046643A | B0 B8 | MOV AL,0B8 |
| 0046643C | 6A 00 | PUSH 0 |
| 0046643E | 50 | PUSH EAX |
| 0046643F | 53 | PUSH EBX |
| 00466440 | 0FB74424 08 | MOVZX EAX,WORD PTR SS:[ESP+8] |
| 00466445 | 50 | PUSH EAX |
| 00466446 | 8D85 14BB4000 | LEA EAX,DWORD PTR SS:[EBP+40BB14] |
| 0046644C | 0FB618 | MOVZX EBX,BYTE PTR DS:[EAX] |
| 0046644F | FF0C24 | DEC DWORD PTR SS:[ESP] |
| 00466452 | 7E 09 | JLE SHORT 0046645D |
| 00466454 | 40 | INC EAX |

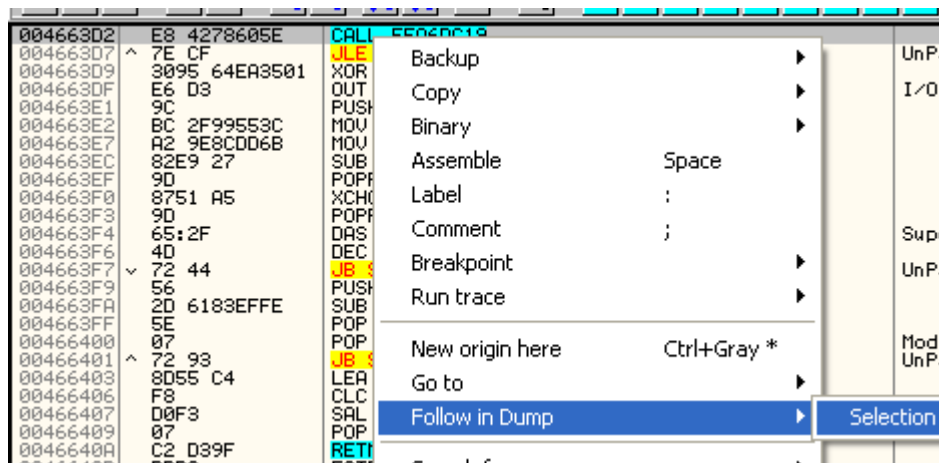
Jump is taken

Alli vemos que es un salto bastante largo lo cual puede ser la diferencia de la desicion entre si una entrada se guardara buena o mala, o sea este si todo va como imaginamos seria el posible salto magico, un salto que decide si la entrada se guardara buena o mala, quiere decir que para que una entrada se guarde buena este salto debe saltar asi que una posibilidad seria hacerlo JMP.

Pero ese salto no se encuentra en el inicio del programa debemos ser muy cuidadosos si reiniciamos el OLLYDBG y buscamos el salto

| | | | |
|----------|---------------|----------------------------------|-------------------|
| 004663D2 | E8 4278605E | CALL 5EA6DC19 | |
| 004663D7 | 7E CF | JLE SHORT 004663A8 | UnPackMe.004663A8 |
| 004663D9 | 3095 64EA3501 | XOR BYTE PTR SS:[EBP+135EA64],DL | |
| 004663DF | E6 D3 | OUT 0D3,AL | I/O command |
| 004663E1 | 9C | PUSHFD | |
| 004663E2 | BC 2F99553C | MOV ESP,3C55992F | |
| 004663E3 | 00 0F000000 | MOV BYTE PTR DS:[EIP+0],0 | |

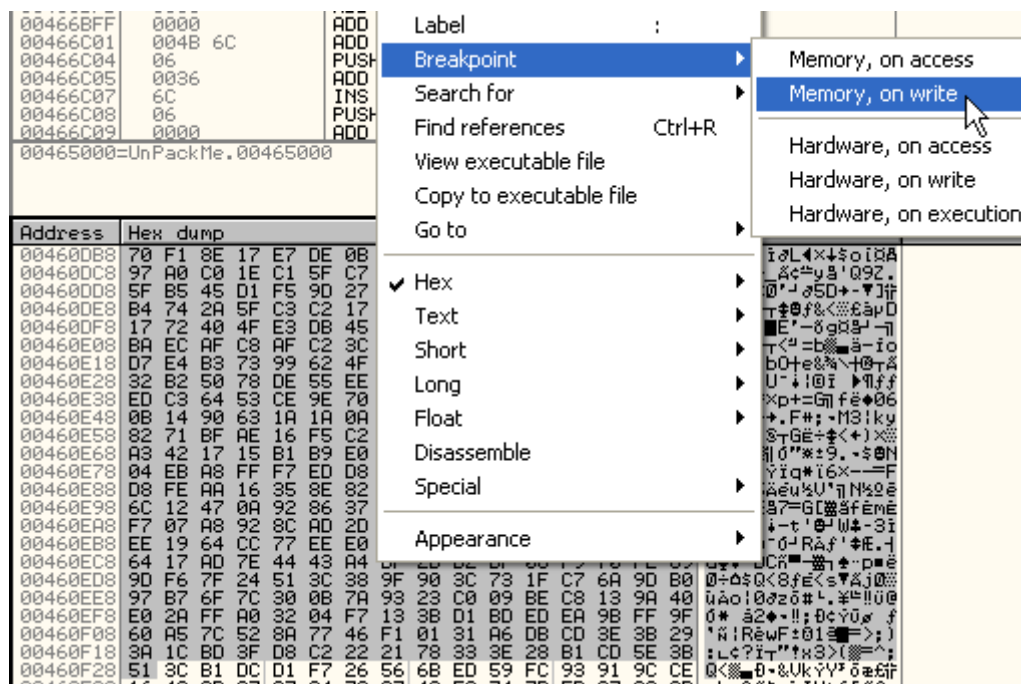
Vemos que al inicio alli hay pura basura, asi que el packer creara ese salto mas adelante.



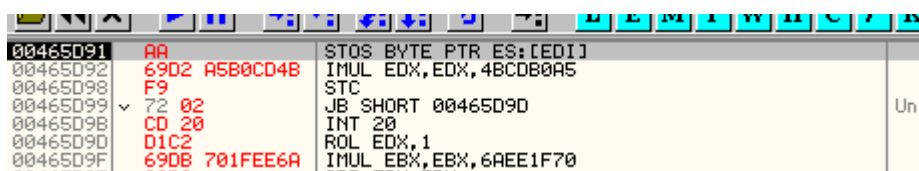
El tema es como parar alli ya que HBP no acepta y BP los detecta asi que deberemos aguzar la imaginacion.

Pondremos un BPM ON WRITE que abarque a toda la IAT para que pare cuando guarda el primer valor.

El inicio de la IAT era 460818 y el final 460f28 abarquemos toda esta zona con el BPM ON WRITE.



Alli esta toda la IAT con el BPM ON WRITE ahora lleguemos adonde guarda el primer valor.



En los STOS para muchisimas veces ya que abarca toda la IAT el BPM, asi que miremos si en este momento ya esta visible el posible salto magico

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004663D2 | 0F84 23010000 | JE 004664FB | UnPackMe.004664FB |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |
| 004663E5 | 0F84 10010000 | JE 004664FB | UnPackMe.004664FB |
| 004663EB | 48 | DEC EAX | |
| 004663EC | 0F85 B2000000 | JNZ 004664A4 | UnPackMe.004664A4 |
| 004663F2 | 60 | PUSHAD | |
| 004663F3 | 8BF7 | MOV ESI,EDI | |
| 004663F5 | 2BC0 | SUB EAX,EAX | |
| 004663F7 | 40 | INC EAX | |
| 004663F8 | 833F 00 | CMP DWORD PTR DS:[EDI],0 | |
| 004663FB | 8D7F 04 | LEA EDI,DWORD PTR DS:[EDI+4] | |
| 004663FE | 75 F7 | JNZ SHORT 004663F7 | UnPackMe.004663F7 |
| 00466400 | 48 | DEC EAX | |
| 00466401 | 0F84 EC000000 | JE 004664F3 | UnPackMe.004664F3 |
| 00466407 | 8BD8 | MOV EBX,EAX | |
| 00466409 | 6BC0 31 | IMUL EAX,EAX,31 | |
| 0046640C | 6A 04 | PUSH 4 | |
| 0046640F | 50 | PUSH 0 | |

Pues esta alli visible veamos que pasa si lo cambiamos a JMP y quitamos el BPM y llegamos al OEP.

| | | | |
|----------|---------------|-----------------------------------|-------------------|
| 004663D2 | E9 24010000 | JMP 004664FB | UnPackMe.004664FB |
| 004663D7 | 90 | NOP | |
| 004663D8 | 89BD 5AD44000 | MOV DWORD PTR SS:[EBP+40D45A],EDI | |
| 004663DE | 8B85 52D44000 | MOV EAX,DWORD PTR SS:[EBP+40D452] | |
| 004663E4 | 40 | INC EAX | |

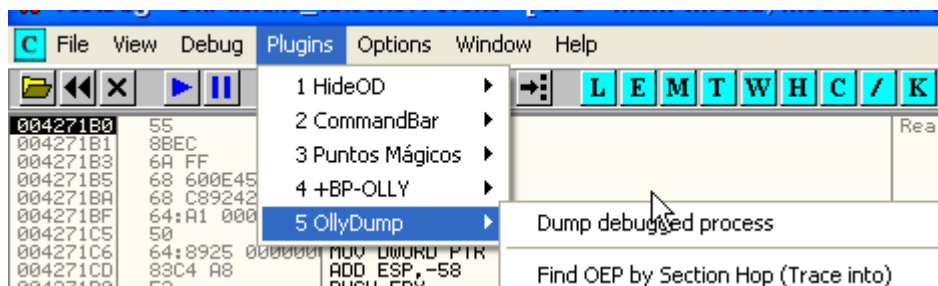
Hay dos posibilidades que el packer detecte los cambios y que falle o que corra y llegue al OEP, demos RUN.

| | | | |
|----------|------------------|-------------------------------|-----------------------------|
| 004271B0 | 55 | PUSH EBP | Real entry point of SFX cod |
| 004271B1 | 8BEC | MOV EBP,ESP | |
| 004271B3 | 6A FF | PUSH -1 | |
| 004271B5 | 68 600E4500 | PUSH 450E60 | |
| 004271BA | 68 C8924200 | PUSH 4292C8 | |
| 004271BF | 64:A1 00000000 | MOV EAX,DWORD PTR FS:[0] | |
| 004271C5 | 50 | PUSH EAX | |
| 004271C6 | 64:8925 00000000 | MOV DWORD PTR FS:[0],ESP | |
| 004271CD | 83C4 A8 | ADD ESP,-58 | |
| 004271D0 | 53 | PUSH EBX | |
| 004271D1 | 56 | PUSH ESI | |
| 004271D2 | 57 | PUSH EDI | |
| 004271D3 | 8965 E8 | MOV DWORD PTR SS:[EBP-18],ESP | |
| 004271D6 | FF15 DC0A4600 | CALL DWORD PTR DS:[460ADC] | kernel32.GetVersion |
| 004271DC | 33D2 | XOR EDX,EDX | |
| 004271DE | 8AD4 | MOV DL,AH | |
| 004271E0 | 8915 34E64500 | MOV DWORD PTR DS:[45E634],EDX | |
| 004271F6 | 8BC8 | MOV ECX,EAX | |

Alli llegamos al OEP miremos la IAT

| Address | Hex dump | ASCII |
|----------|---|----------------------|
| 004607FC | 78 2B 06 00 60 2B 06 00 4C 2B 06 00 2E 2B 06 00 | x+..'+..L+..+.. |
| 00460800 | 00 00 00 00 00 00 00 00 00 00 00 00 00 68 DA 77 |C.....-k rw |
| 0046081C | 1B 76 DA 77 F4 EA DA 77 E7 EB DA 77 83 78 DA 77 | +Orw?U rw?U rw?X rw |
| 0046082C | 00 00 00 00 DD 15 C5 58 2E BD C3 58 00 00 00 00 | ...!\$+X.c!X... |
| 0046083C | 04 6A EF 77 66 95 EF 77 89 6A EF 77 F3 AD EF 77 | Ej'wfo'weJ'w?..'w |
| 0046084C | ED 09 EF 77 99 88 EF 77 C0 B5 EF 77 2A 7D EF 77 | Y-'w0i'w!?'w*}'w |
| 0046085C | B2 7C EF 77 77 53 F2 77 1E C9 F1 77 0C BC EF 77 | #!'wWS=w?F?w..'w |
| 0046086C | 52 04 EF 77 FA 80 EF 77 F1 00 EF 77 51 B2 EF 77 | Re'w.'i'w?!'w?..'w |
| 0046087C | 26 05 EF 77 2A E3 EF 77 5F 39 F2 77 71 B4 EF 77 | &'w*0'w_9=wq]'w |
| 0046088C | 2E AD EF 77 E1 61 EF 77 B8 85 EF 77 CC 02 EF 77 | ..?'wB?w0?wlf?'w |
| 0046089C | 43 70 EF 77 FB EA F0 77 12 83 EF 77 01 72 F0 77 | Cp'w!?'w??'w0?-'w |
| 004608AC | A9 34 F0 77 D5 93 EF 77 68 EF EF 77 AA 02 EF 77 | 04-w'?'wh'?'w?'w |
| 004608BC | B2 6F EF 77 3F 38 F2 77 D6 E8 EF 77 68 E0 EF 77 | #o'w?8=wip'wh?'w |
| 004608CC | 00 60 EF 77 90 58 EF 77 6D AC EF 77 94 6C F0 77 | ..'wEL'wm?'w0l-'w |
| 004608DC | 22 8D EF 77 3D C8 F1 77 3D 6D F0 77 6F C0 EF 77 | ..'w?='w=m-'wol-'w |
| 004608EC | 85 78 EF 77 26 D9 EF 77 FB 5E EF 77 36 8A EF 77 | ac'w&'w!?'w6?'w |
| 004608FC | FC 8A EF 77 0F 62 EF 77 49 5E EF 77 97 50 EF 77 | ??'w*b'wI?'w?'w |
| 0046090C | 1A 9A EF 77 6B FA EF 77 7B C9 F0 77 DA 98 F2 77 | +?'wk'?'w(f-'w?'w |
| 0046091C | 1A 04 F2 77 55 EA EF 77 C5 61 EF 77 70 E6 EF 77 | +?'wU?'w?'w?'w |
| 0046092C | F0 81 EF 77 2D 6C EF 77 98 6E EF 77 4F 83 EF 77 | -?'w-'l'w?'w0?'w |
| 0046093C | 09 ED EF 77 EB AA EF 77 26 69 F0 77 B1 95 EF 77 | ..'w?'w&'l-'w?'w |
| 0046094C | 6F 80 EF 77 8A 5A EF 77 E9 49 F2 77 26 F1 F0 77 | c?'w??'w0l='w&'-'w |
| 0046095C | C9 D0 F0 77 51 E0 F0 77 33 8C EF 77 6C EC EF 77 | f!'-'w00-'w3i'w?'w |
| 0046096C | 29 94 EF 77 00 00 00 6B 17 80 7C C1 C9 80 7C |)?'w...'k?C!+'fC! |
| 0046097C | 69 10 81 7C EE 1E 80 7C 8D 2C 81 7C 40 7A 94 7C | i!?'-'?C!l'?'@?! |
| 0046098C | E1 EA 81 7C A2 CA 81 7C 16 1E 80 7C 43 99 80 7C | B0!l'?'?!'-'?C!C0! |
| 0046099C | 10 11 81 7C 29 29 81 7C 14 9B 80 7C 81 9A 80 7C | !?'l'?)!?'wC!?'wC! |
| 004609AC | FB 2C 82 7C AE 94 83 7C 2B 2E 83 7C 04 CE 80 7C | '?'e!?'??'+'?'-'fC! |
| 004609BC | 8A 2B 86 7C 3F DC 81 7C 5F 48 81 7C 23 CC 81 7C | e+'??'?'_Hu!?'fu! |
| 004609CC | 78 2C 81 7C 86 03 81 7C B9 8C 83 7C 80 A4 80 7C | x'?'?'?'?'?'?'?'?'? |
| 004609DC | 57 BB 80 7C 7F D4 80 7C 72 17 81 7C 93 D2 80 7C | lmC!?'?C!?'?'?'?'?'? |

Todos valores correctos, jeje ya tenemos la IAT reparada, ahora dumpeemos, podiamos haberlo hecho antes no hay diferencia con esto.



OllyDump - UnPackMe_tElack0.98.exe

Start Address: 400000 Size: 69000

Entry Point: 66BD6 -> Modify: 271B0 Get EIP as OEP

Base of Code: 1000 Base of Data: 4B000

☒ Fix Raw Size & Offset of Dump Image

| Section | Virtual Size | Virtual Offset | Raw Size | Raw Offset | Charact |
|---------|--------------|----------------|----------|------------|---------|
| .teddy | 0004A000 | 00001000 | 0004A000 | 00001000 | C0000C |
| .teddy | 0000C000 | 0004B000 | 0000C000 | 0004B000 | C0000C |
| .teddy | 00009000 | 00057000 | 00009000 | 00057000 | C0000C |
| .teddy | 00003000 | 00060000 | 00003000 | 00060000 | C0000C |
| .rsrc | 00002000 | 00063000 | 00002000 | 00063000 | C0000C |
| .teddy | 00004000 | 00065000 | 00004000 | 00065000 | C0000C |

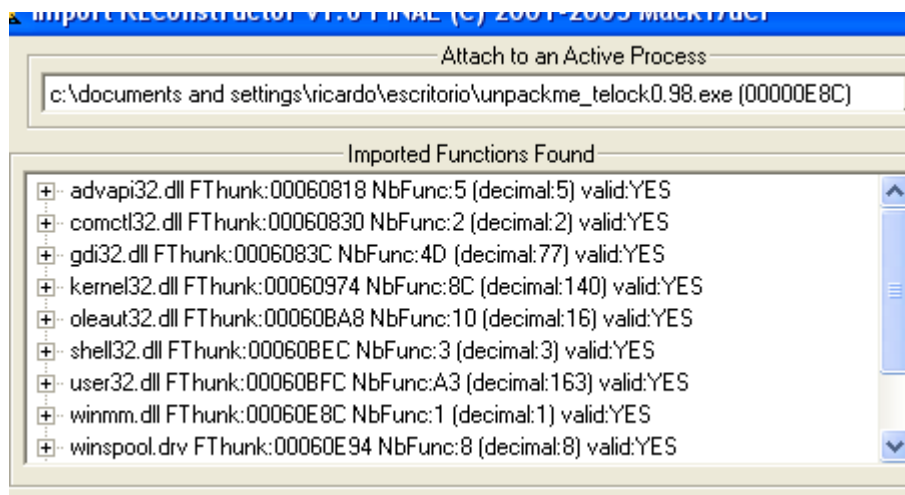
☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

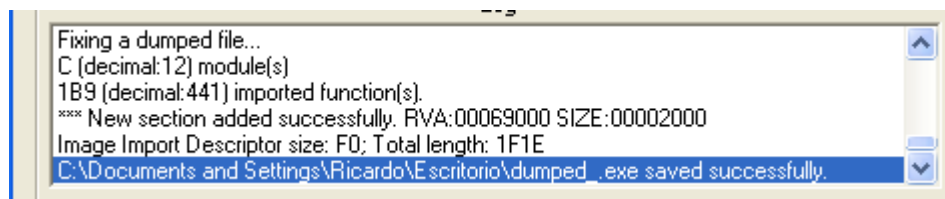
Le quito la tilde a REBUILD IMPORT.

Y dumpeo, ahora abro el IMP REC y le pongo nuevamente los valores de RVA; SIZE Y OEP.

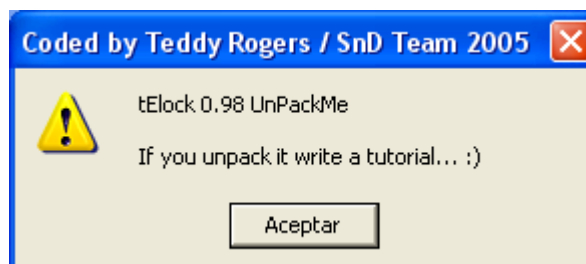


Veo que ahora sale todo valido, el salto magico hizo su magia jeje.

Ahora reparo el dumpeado apretando FIX DUMP.



Alli lo guardo como dumped_.exe y lo ejecuto a ver si funciona.



Funciona perfecto jeje

Bueno hemos visto un metodo para hallar el salto magico que siempre varia de un packer a otro, pero comparando y mirando un poco entre lo que hace con una entrada buena y una mala, siempre lo podemos llegar a ubicar facilmente, continuaremos con mas packers en la parte 38.

hasta la parte 38
Ricardo Narvaja
22/03/06