Verificando una app con Stolen bytes DAP 0.2 by Apuromafo

Este nivel fue resuelto en el concurso 94, pero quedaba pendiente resolver el antidump, es ***simple*** como quitarlo, pero en su tiempo no lo era, espero colaborar con un granito de arena

El ollydbg que uso tiene strongOD, asi que no detecta ningún truco.

Yo simplemente lo ejecute y revise , pero no pensaba que se podría hacer algo más.

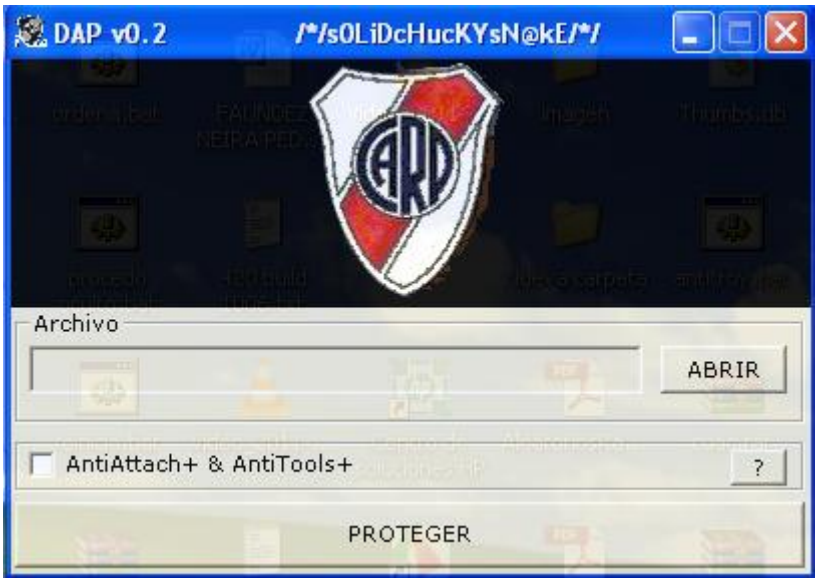
Quienes quieran leer el escrito de solid:

http://ricardonarvaja.info/WEB/CONCURSOS%20VIEJOS/CONCURSOS%202004-2006/CONCURSO%2094/C_94_N3_solid.rar

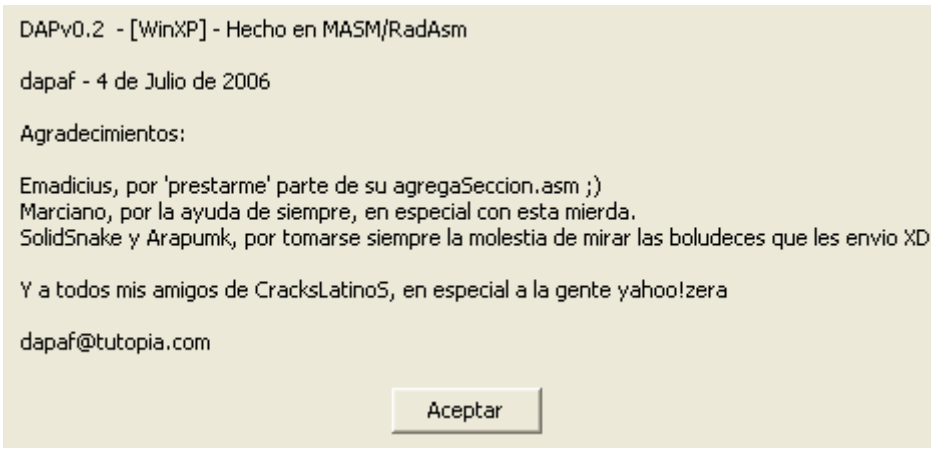
yo para escribir este escrito no lo usé, pero quien este este packer, siempre es bueno resolver los temas pendientes.

este es la apariencia del packer en si, Yo quiero acotar al antidump, agregar un manifiest y borrar el titulo que acompaña, para que se vea mas bonito ☺

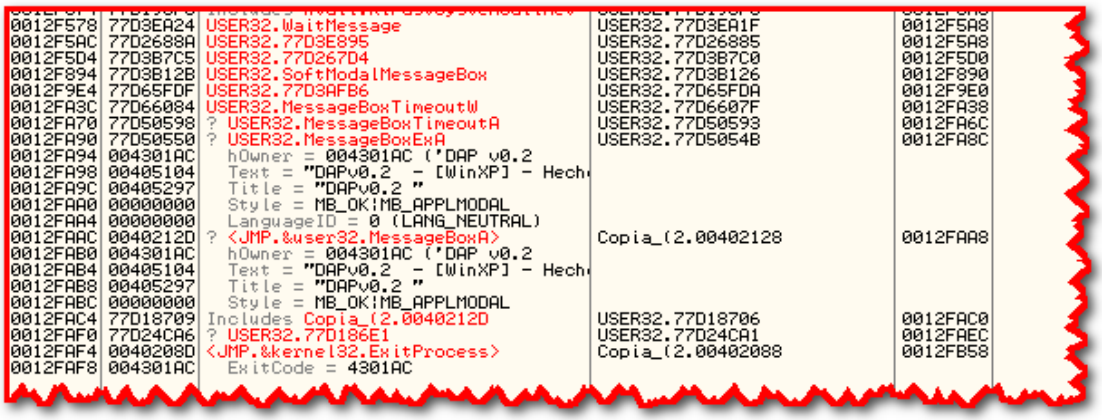
este es la apariencia inicial:



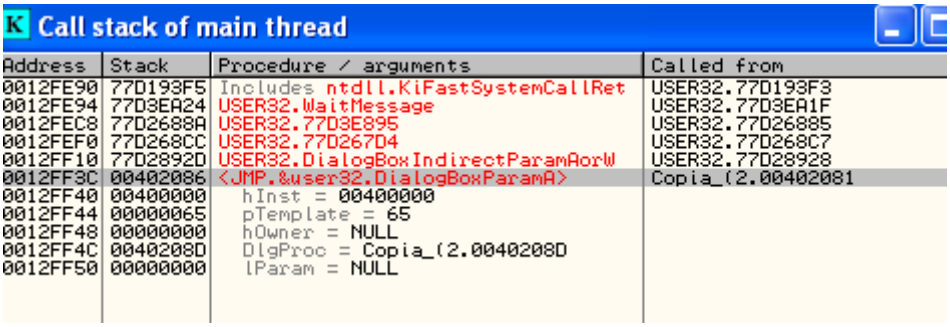
Si ejecutado y al pulsar el logo de river veo un mensaje



Veamos que me muestra en el ollydbg con este mensaje vemos:

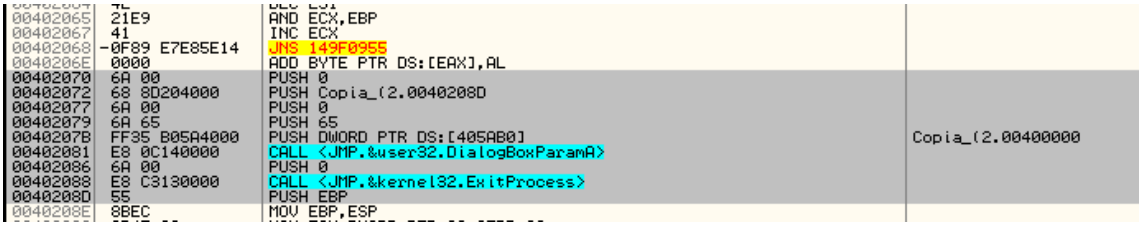


Pero si pauso sin ese mensaje veo



El dialogo comienza en 40208d

Veamos por ahí



Y al revisar a la iat se ve InitCommon

```
004034C2 -FF25 64404000  JMP DWORD PTR DS:[<&user32.SetLayeredWindowAttributes>]   USER32.SetLayeredWindowAttri
004034C8 -FF25 68404000  JMP DWORD PTR DS:[<&user32.SetWindowLongA>]              USER32.SetWindowLongA
004034CE -FF25 00404000  JMP DWORD PTR DS:[<&comctl32.InitCommonControls>]        COMCTL32.InitCommonControls
004034D4   0000          ADD BYTE PTR DS:[EAX],AL
004034D6   0000          ADD BYTE PTR DS:[EAX],AL
004034D8   0000          ADD BYTE PTR DS:[EAX],AL
```

Veamos cuando lo llama

Hardware, on execution

Al reiniciar vemos

```
0012FF50  00402070  CALL to InitCommonControls from Copia_(2.0040206B
0012FF54  00000015
```

40206b

Tenemos entonces el OEP por aquí, pero habrá algún ¿**valor escrito antes**?

```
0040206B  E8 5E140000    CALL <JMP.&comctl32.InitCommonControls>
00402070  6A 00          PUSH 0
00402072  68 8D204000    PUSH Copia_(2.0040208D
```

Si busco alguna referencia de esta dirección vemos como se hace:

```
0040206B  CALL <JMP.&comctl32.InitCommonControls>  (Initial CPU selection)
00414618  MOV DWORD PTR DS:[EAX],Copia_(2.0040206  0040206B=Copia_(2.0040206B
```

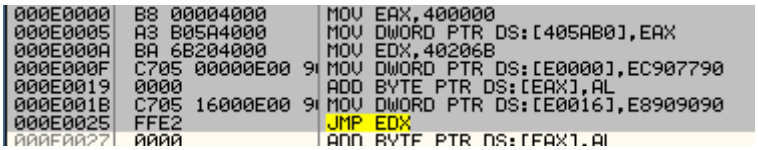
```
004145E5  . 33C0          XOR EAX,EAX
004145E7  . 33C9          XOR ECX,ECX
004145E9  . 05 00000E00   ADD EAX,0E0000
004145EE  . C700 B8000040 MOV DWORD PTR DS:[EAX],400000B8
004145F4  . 83C0 04       ADD EAX,4
004145F7  . C700 00000000 MOV DWORD PTR DS:[EAX],0
004145FD  . 83C0 01       ADD EAX,1
00414600  . C700 A3B05A40 MOV DWORD PTR DS:[EAX],405AB0A3
00414606  . 83C0 04       ADD EAX,4
00414609  . C700 00000000 MOV DWORD PTR DS:[EAX],0
0041460F  . 83C0 01       ADD EAX,1
00414612  . C600 BA       MOV BYTE PTR DS:[EAX],0BA
00414615  . 83C0 01       ADD EAX,1
00414618  . C700 6B204000 MOV DWORD PTR DS:[EAX],Copia_(2.0040206B
0041461E  . 83C0 04       ADD EAX,4
00414621  . C700 C7050000 MOV DWORD PTR DS:[EAX],5C7
00414627  . 83C0 04       ADD EAX,4
0041462A  . C700 0E009077 MOV DWORD PTR DS:[EAX],7790000E
00414630  . 83C0 04       ADD EAX,4
00414633  . C700 90EC0000 MOV DWORD PTR DS:[EAX],0EC90
00414639  . 83C0 04       ADD EAX,4
0041463C  . C700 C7051600 MOV DWORD PTR DS:[EAX],1605C7
00414642  . 83C0 04       ADD EAX,4
00414645  . C700 0E009090 MOV DWORD PTR DS:[EAX],9090000E
0041464B  . 83C0 04       ADD EAX,4
0041464E  . C700 90E80000 MOV DWORD PTR DS:[EAX],0E890
00414654  . 83C0 02       ADD EAX,2
00414657  . 91            XCHG EAX,ECX
00414658  . 66:C701 FFE2  MOV WORD PTR DS:[ECX],0E2FF
0041465D  . 61            POPAD
0041465E  .vEB 21         JMP SHORT Copia_(2.00414681
```

Bien teníamos un ejemplo de copiar trozos de memoria con rep es lo mas común en trozos de stolen , pero aquí es con MOV, mueve a 0e0000 , veamos ese trozo

Asi esta cuando ya estamos en el oep

```
000E0000   90             NOP
000E0001 -77 90           JA SHORT 000DFF93
000E0003   EC             IN AL,DX                          I/O command
000E0004   00A3 B05A4000  ADD BYTE PTR DS:[EBX+405AB0],AH
000E000A   BA 6B204000    MOV EDX,40206B
000E000F   C705 00000E00 9( MOV DWORD PTR DS:[E0000],90909090
000E0019   E8 00C70516    CALL 1613C71E
000E001E   000E           ADD BYTE PTR DS:[ESI],CL
000E0020   0090 9090E8FF  ADD BYTE PTR DS:[EAX+FFE89090],DL
000E0026 vE2 00           LOOPD SHORT 000E0028
000E0028   0000           ADD BYTE PTR DS:[EAX],AL
```

Y asi es cuando estaba, y claramente se auto/edita en la dirección 0e0000 notese la diferencia



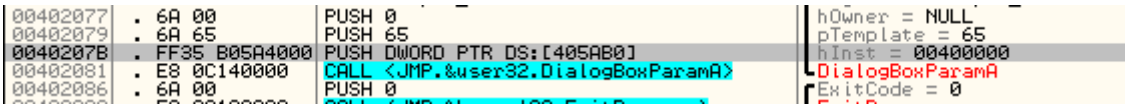
Vemos como realmente mueve el valor del

Mov eax,400000 y

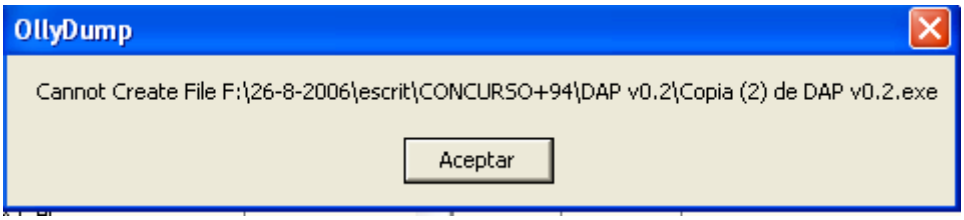
MOV DWORD PTR DS:[405AB0],EAX

Lo cual es lo mismo que **MOV DWORD PTR DS:[405AB0], 400000**

Ahora bien, es necesario ese valor pues es usado como hInst en el dialogo

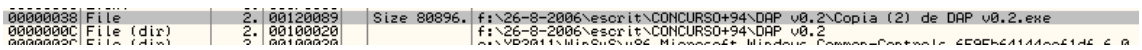


Ahora que sabemos el oep, tenemos el stolen byte, y ahora queda dumpear



Solución:

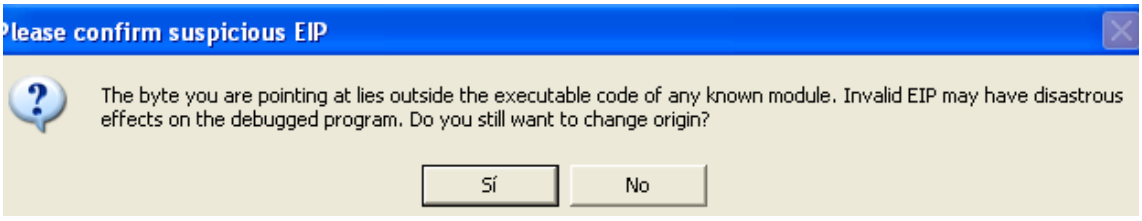
Esto pasa por usar createFileA, la forma de anular es buscar el handle y anularlo

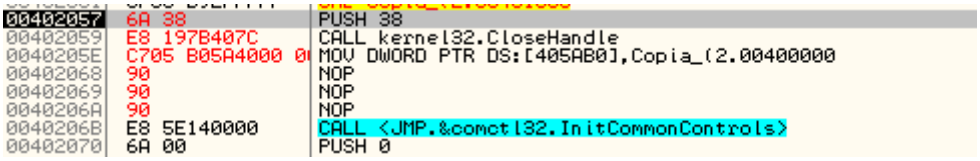


En este caso es 38 y llamar a CloseHandle mediante

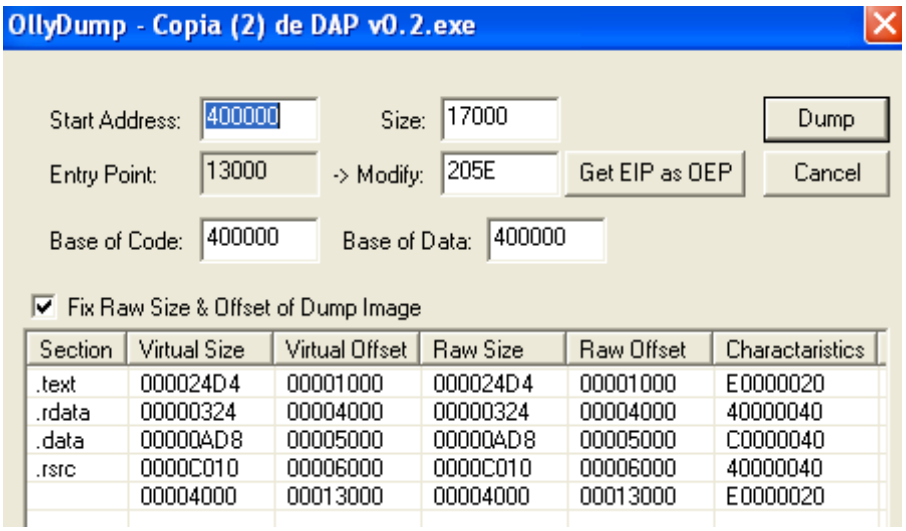
push 38 y call a esa api,

cambio el origen de eip

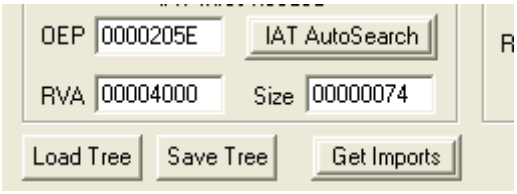




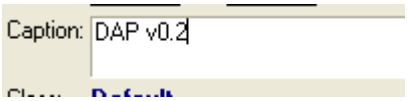
Y ahora podemos **dumpear tranquilamente** con cualquier dumper. al llegar a 40205e:



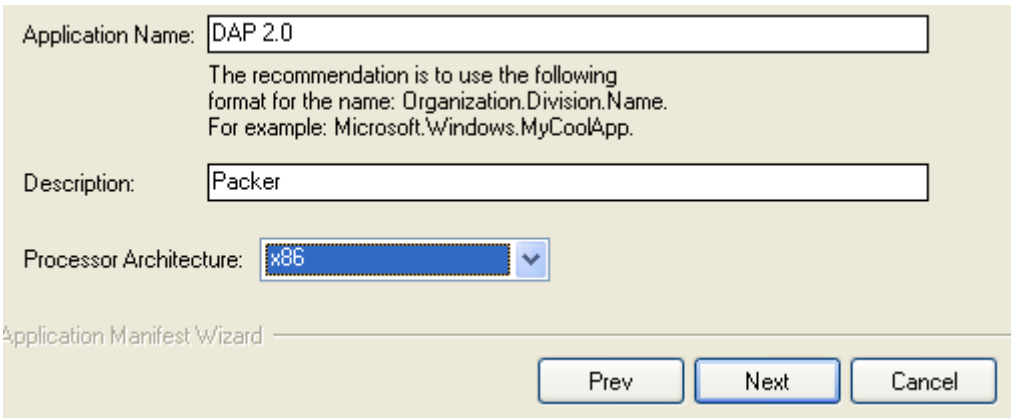
Usar import rec



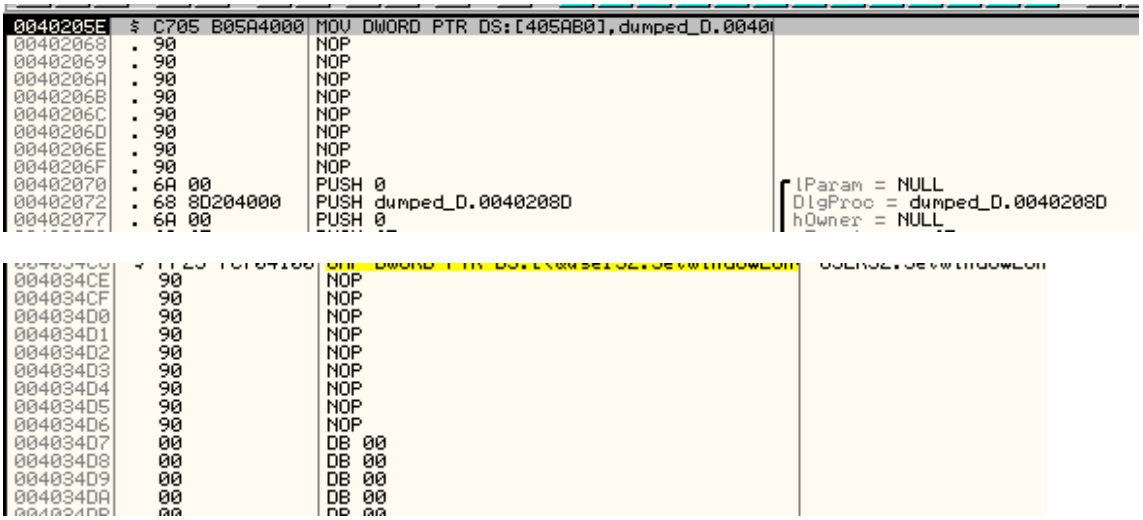
Y al objetivo: unpacked, cambiamos el titulo con PExplorer R6



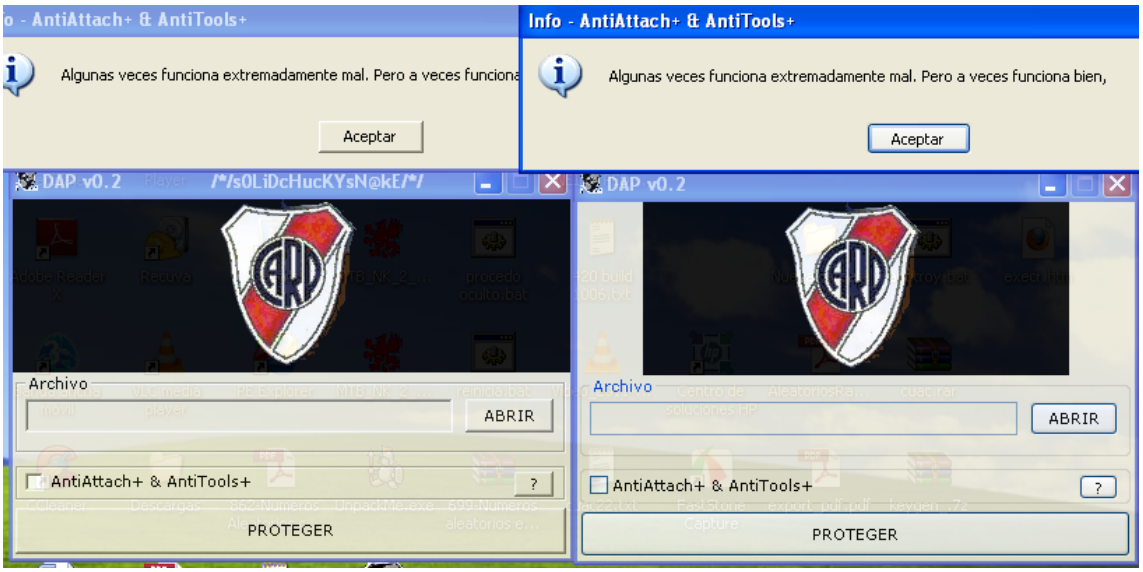
Y le agregamos el manifiest



Ahora elimino el init, porque a la larga es como un call +ret=nop



Y ahora tengo tranquilamente el Dap 0.2 unpacked y con mejor apariencia en los botones.



Eso es todo, saludos Apuromafo