

A decorative graphic featuring three blue circles of varying sizes, each composed of concentric rings. Two thin blue lines intersect diagonally across the page, one from the top-left and another from the top-right, meeting near the center. A large blue circle is partially visible in the bottom-right corner.

Como desempaque un Armadillo v8 by Apuromafo

Tenia animo de ver que novedades traia armadillo, y cuando termine de revisarlo, logre un unpacked, por eso prefiero mostrar en el escrito y no enviar ningun unpacked.exe

Apuromafo
28/10/2010

Buenas tardes, lo que voy a presentar puede ser un poco extraño pues normalmente no me animo a desempacar Los programas principales solo los unpackmes, pero ya pasado un tiempo creo que los minutos o horas pueden ser utiles para alguien que quiere hacer pruebas y por ahi me inscribi en tuts4you donde me llegan las actualizaciones al mail, asi que me anime a revisarlo.

Armadillo es un programa Comercial, que Comprime, Tiene proteccion y gran manejo de su ejecutable en general, Puede ser facilmente inlineado, por lo que desarrollo desde el comienzo un CopymemII y un debugbloquer puede ser capaz de bypasear casi cualquier unpacker , pues se maneja entre un proceso padre o hijo, o 2 procesos con un Valor asignado,por eso los loader o algun inline tiene gran repercusion, pero en este caso, siempre hay otra opcion que es desempacar.creo que le tendremos unos 30 o cuarenta tutoriales

La dinamica mas comun Para desempacar es llegar al oep, verificar si esta encriptada ciertas zonas, Verificar antidumps, reparar la Cabezera, Import Table (iat) , quitar un poco de ofuscacion o splicies)

Dumpear, reparar, Y intentar desarmar la protecci3n Con el uso de sus variables o conocer las nuevas estructuras,puede que no sean mucho, pero me imagino un armadillo version 12, ya cercano a el 2014 , asi que prefiero mirarlo ahora y no en 4 a3os mas..

90% de las veces que uno se encuentra con un armadillo, termina dumpeando el proceso, y buscando donde hacer los cambios para hacer inline o un patch que propiamente tal desempacarlo,

Pero cuando uno quiere verificar si los script que uno usa sirven y los viene heredando desde las versiones 3, 4 , 5 , 6 , pues nace la idea de probar que nuevas caracter3sticas traen los packers de versi3n 7 y 8 y si pueden ser utiles

El metodo de keygenning en Armadillo es desconocido, pero se mas o menos donde va orientado.

Proyeccion:Armadillo ha evolucionado su dll y sus integraciones con un entorno muy bonito, asi que buen trabajo fue hecho con xor, y otras mutaciones


Formas de atacarlo: parchar la DLL interna por ende propiamente tal desde armadillo o bien parchar el unpacked donde no hay armadillo,

yo prefiero desempacarlo pero me encantan los unpackers, porque ahorran tiempo para algo que podemos repetir algun dia.

El Packer a mencionar es de gran Renombre, y sobre todo sobre sus Secured Section, Nanomites, Import Destruction y otras variables , espero que por mencion de tiempo se pueda valorar lo poco y reservado de este documento.No creo que hayan referencias de como lo desempacaron , solo como esta unpacked. Asi que animo.

Actualmente en septiembre leia que habia una version 8, y la baje para cuando tuviera algun dia libre, y hoy le toco el escrito-.

Revision History[Top](#) [Previous](#) [Next](#)

 **Note: Also see the revision history in the Armadillo Help file, which will include information about any updates that affect your protected programs. Armadillo Help is included in the SoftwarePassport/Armadillo installation.**

Version 8.00, 30September2010

Yo como no tengo internet y solo aveces leo comentarios, de bajar una aplicacion pueden pasar semanas, asi que me animo a desempacar un armadillo version 8, para compartir, no creo que sea novedad pues hace no mucho vi que habia uno del team Resurrection.

Targets vistas: Armadillo 7.4.0.740+ SoftwarePassport.exe, Armadillo 4.3a private,

Armadillo 8 + SoftwarePassport.exe

Compresor/Compilador: Armadillo /Vc++ VisualBasic

Objetivos: desempacar y mostrar que es posible, como son varias herramientas conforme sirvan los usare..

Cracker: Apuromafo

Fecha: 28/10/10

Sugiero antes de leer esto, pues practicar con cada una de las teorías de armadillo, sugiero como obligatorio:

847-Armadillo 4.62 + Debug Blocker + CopyMem II + Import Table Elimination + Code Splicing + Nanomites ? PARA PRINCIPIANTES ? USANDO TOOLS ? Por Solid.rar

Y me anime a revisar el armadillo 7 y pude desempacarlo, hace poco apareció el 8 y vi que había sido crackeado por alguien de {ReS} así que cuando lo baje, vi que estaba armadillo.exe y su dll, pero se olvidaron de SoftwarePassport.exe ,

así que como quien se propone lo logra, así comento esta mini historia con fines educativos, quien quiera hacer mal uso, puede borrar este documento de su Disco Duro.

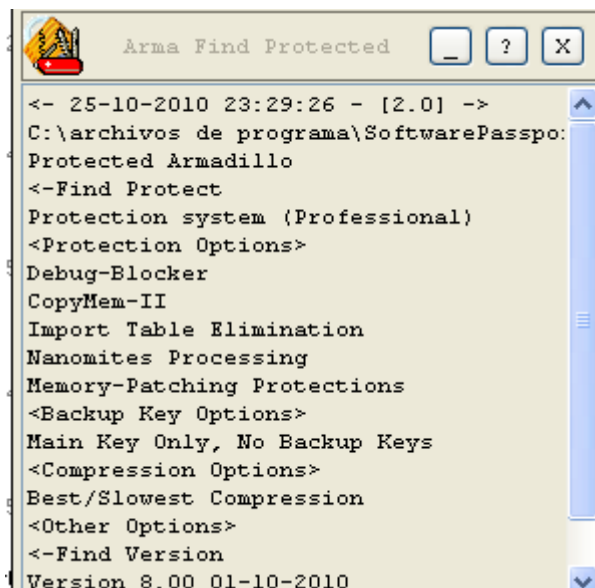
Vista previa de la carpeta Armadillo 7/8 :



Vel, codifico una gran tool para indentificar que version existe de armadillo

Comenzamos con el Armadillo 8

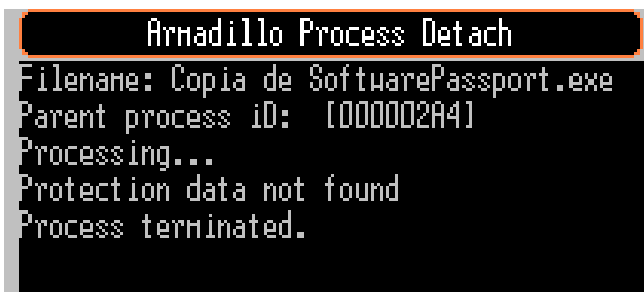
En este caso tenemos que es un armadillo con varias opciones.

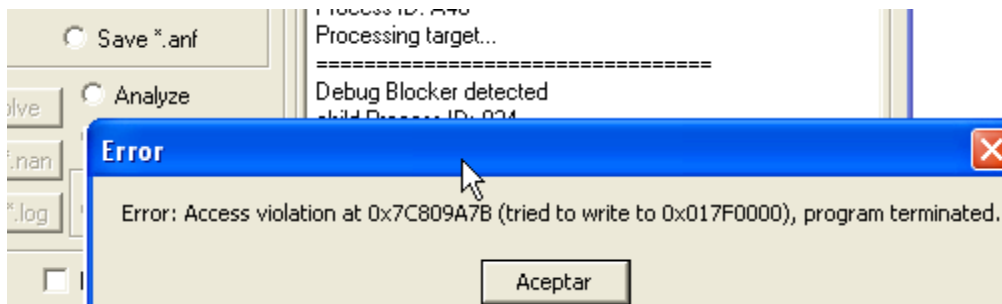


Primero, las secciones tienen otros nombres, no los tipicos de siempre

Name	VOffset	VSize	HOffset	HSize	Flags
.wkyntx	00001000	00167692	00000000	00000000	60000020
.ywnitz	00169000	00053FE0	00000000	00000000	40000040
.ritdub	0018D000	0000AC04	00000000	00000000	C0000040
.tgby	001C8000	000B0000	00001000	000A4000	E0000020
.hvnz	00278000	00010000	000A5000	0000D000	E0000020
.qlyrz	00288000	00030000	000B2000	0001D000	C0000040
.ajue	002B8000	00390000	000CF000	00388000	C0000040
.lzwpk	00648000	004E0000	00457000	00004000	40000040

por lo que las estructuras normales y genericas hacen caer a nuestras tools :

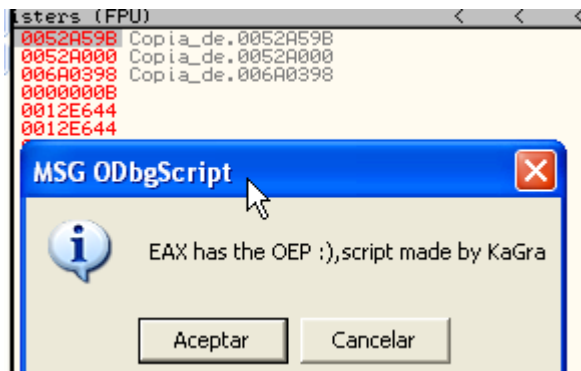




Starting the Extraction Procedure!
-> Real PDATA Size: 00387F5E
-> Appears to be 0x000B DWORDS
Processing File For DWORDS: C:\archivos de programa\SoftwarePassport8.0.0.8\Copia de Copia de Armadillo.exe*
-> Offset Calculator Initialized
-> There are 11 DWORDS in use
-> Correcting...
-> Successfully corrected the PDATA.
-> ZLIB DLL Found with Bitmap!
-> Compressed Size: 00085E41
-> Decompressed Size: 00131000
-> Decompressing...
Decompressed! Saving File...
-> Saved!
Getting correct CRC Info...
-> Could not find search String! Maybe CRC has been patched already!

Como hay copymewll, tenemos un call encriptador y si hay mas cosas, pues a dumper y comparar mas..

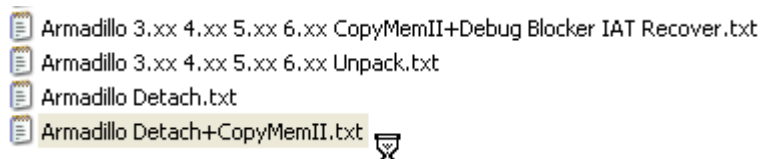
Tengo desde una version 4 que resulta el script de kagra para saber el oep



Como conozco un poco mas, se que esto se optimizo y es llamado Detach +copymewll

Fungus, una vez compartio algunos script que nos ayudan a hacer generica la acción y evito que juntara mis scripts en la carpeta.

Hay 3 script que suelo usar, pero solo usare este en este caso:



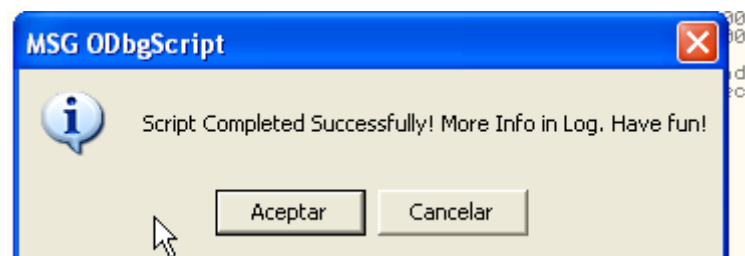
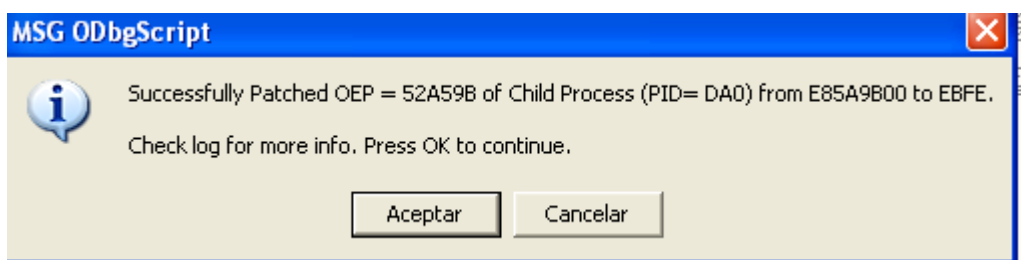
Lo especial es que llegaremos rapido al oep , y lo demas sera de nosotros

```
rtr
sti
gmeme eip, MEMORYBASE
mov CryptoProcess, $RESULT
find CryptoProcess, #8B048A50E8???????83C40C# // "mov eax, dword ptr ds:[edx+ecx*4]" "push eax" "call xxx
cmp $RESULT, 0
je Error1
mov CryptoProcess, $RESULT
add CryptoProcess, 04
mov [CryptoProcess], #9090909090#
log CryptoProcess
log "Crypto Process was nopped."

eval "Successfully Patched OEP = {ChildOEP} of Child Process (PID= {ChildProcessID}) from {OEPBytes} to EBFE.\r\n\r\n"
log $RESULT
msg $RESULT
```

Al ejecutar el script, tengo automatizado lo que compartia solid en su escrito

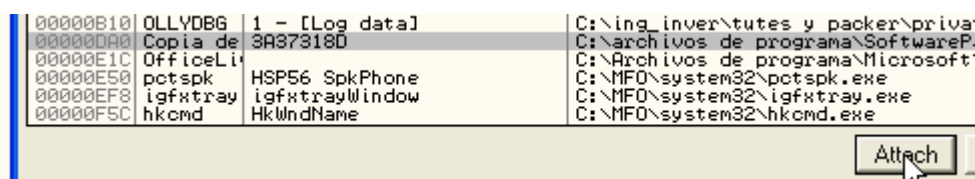
El mensaje me dice:



La informacion:

Address	Message
8EE000	CodeBegin: 00401000
8EE000	DataBegin: 00833000
678000	CodeBegin: 00401000
678000	DataBegin: 00569000
7C85A610	ProcessDebugEvent: 0012F3C0
7C85A610	OEPOffset1: 0012F3D8
7C85A610	OEPOffset2: 0012F3E4
7C85A610	OEPOffset3: 0012F3E8
7C80220F	Address: 0052A000
7C80220F	Buffer: 0142CBD0
7C80220F	OEP of Child Process was patched to EBFE
7C80220F	ChildOEP: 0052A59B
7C80220F	ChildProcessID: 00000DA0
5D443E	CryptoProcess: 005D39F1
5D443E	Crypto Process was nopped.
5D443E	\$RESULT: Successfully Patched OEP = 52A59B of Child Process (PID= DA0) from E85A9B00 to E
5D443E	Check log for more info. Press OK to continue.
7C85A610	Patch1: 005D0FEC

Sin cerrar este olly, abro otro , Coloco attach y vamos al oep para colocar un bp



Veamos los thread: si hay alguno en pause, darle resume cuando tengamos el bp en el oep:

Ident	Entry	Data block	Last error	Status	Priority	User time	System time
00000628	7C810669	7FFDD000	ERROR_INVALID_HANDLE	Active	32 + 0	0.0000 s	0.0000 s
0000076C	7C810669	7FFDD000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
00000B54	7C961E0B	7FFDE000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
00000D94	7C810669	7FFDD000	ERROR_ALREADY_EXISTS	Active	32 + 0	0.0000 s	0.0000 s
00000F44	7C810669	7FFDD000	ERROR_NO_TOKEN	Active	32 + 0	0.0000 s	0.0000 s
00000F98	00000000	7FFDF000	ERROR_FILE_NOT_FOUND	Active	32 + 0	103.2812 s	0.1718 s

Coloco go to expresion CTRL+G al oep : 52a59b y le coloco un bp en oep

```

0052A59B -EB FE      JMP SHORT Copia_de.0052A59B
0052A59D 9B          WAIT
0052A59E 0000        ADD BYTE PTR DS:[EAX],AL
0052A5A0 ^E9 16FEFFFF JMP Copia_de.0052A3BB
0052A5A5 6A 0C       PUSH 0C
0052A5A7 68 10B45B00 PUSH Copia_de.005BB410
0052A5A9 E8 2B4A0000 CALL Copia_de.0052EFDC
0052A5AB 33DB        XOR EBX,EBX
0052A5AD 895D E4     MOV DWORD PTR SS:[EBP-1C],EBX
0052A5AF 33DB        XOR EBX,EBX

```

F9(ejecutar) y llego a esto (recordar que deben estar todos los thread activos)

```

0052A59B -EB FE      JMP SHORT Copia_de.0052A59B
0052A59D 9B          WAIT
0052A59E 0000        ADD BYTE PTR DS:[EAX],AL
0052A5A0 ^E9 16FEFFFF JMP Copia_de.0052A3BB
0052A5A5 6A 0C       PUSH 0C
0052A5A7 68 10B45B00 PUSH Copia_de.005BB410
0052A5A9 E8 2B4A0000 CALL Copia_de.0052EFDC
0052A5AB 33DB        XOR EBX,EBX
0052A5AD 895D E4     MOV DWORD PTR SS:[EBP-1C],EBX
0052A5AF 33DB        XOR EBX,EBX

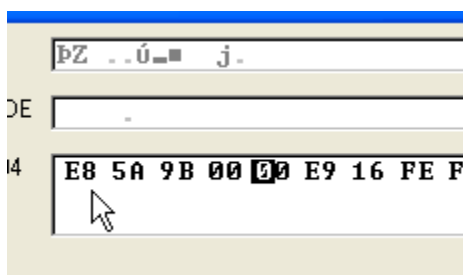
```

Detenidos, ahora cambio

```

0052A59B -EB FE      JMP SHORT Copia_de.0052A59B
0052A59D 9B          WAIT
0052A59E 0000        ADD BYTE PTR DS:[EAX],AL
0052A5A0 ^E9 16FEFFFF JMP Copia_de.0052A3BB
0052A5A5 6A 0C       PUSH 0C
0052A5A7 68 10B45B00 PUSH Copia_de.005BB410
0052A5A9 E8 2B4A0000 CALL Copia_de.0052EFDC
0052A5AB 33DB        XOR EBX,EBX
0052A5AD 895D E4     MOV DWORD PTR SS:[EBP-1C],EBX
0052A5AF 33DB        XOR EBX,EBX

```



Quedando desempacado en memoria pero ojo Tenemos Nanomites iat elimination asi que no hay que ejecutar nada aun.

```

0052A59B E8 5A9B0000 CALL Copia_de.005340FA
0052A5A0 ^E9 16FEFFFF JMP Copia_de.0052A3BB
0052A5A5 6A 0C       PUSH 0C
0052A5A7 68 10B45B00 PUSH Copia_de.005BB410
0052A5A9 E8 2B4A0000 CALL Copia_de.0052EFDC
0052A5AB 33DB        XOR EBX,EBX
0052A5AD 895D E4     MOV DWORD PTR SS:[EBP-1C],EBX
0052A5AF 33DB        XOR EBX,EBX

```

Si pruebo usar armaggedon para detener el debugbloquer,

y pruebo el script comun , tampoco da resultado pues tiene Copymewll y claramente esta usando el call encriptador:

0052A59B 125A 15 ADC BL, BYTE PTR DS:[EDX+15]
0052A59E 3AFA CMP BH, DL
0052A5A0 -E9 98C405FF JMP FF586A3D
0052A5A5 E4 36 IN AL, 36
0052A5A7 92 XCHG EAX, EDX
0052A5A8 103A ADC BYTE PTR DS:[EDX], BH
0052A5AA 61 POPAD
0052A5AB FA CLI
0052A5AC E8 A570FA00 CALL 014D1656
0052A5B1 BD E1735D6A MOV EBP, 6A5D73E1
0052A5B6 093A OR DWORD PTR DS:[EDX], EDI
0052A5B8 8BF3 MOV ESI, EBX
0052A5BA 32C1 XOR AL, CL
0052A5BC FB
0052A5C7 81AF 3FA
0052A5C8 0049 3FA
0052A5CB EC
0052A5CC 008E 3FA
0052A5CD 7000 3FA

Script Log Window
Address Message
678000 IsDebuggerPresent patched
678000 OutputDebugString patched

71F00000 Module C:\MF0\system32\snmpapi.dll
71A50000 Module C:\MF0\system32\wsock32.dll
76D00000 Module C:\MF0\system32\mprapi.dll
77C90000 Module C:\MF0\system32\activeds.dll
76D00000 Module C:\MF0\system32\adsldpc.dll
597F0000 Module C:\MF0\system32\netapi32.dll
76F20000 Module C:\MF0\system32\ldap32.dll
76AE0000 Module C:\MF0\system32\atl.dll
76E40000 Module C:\MF0\system32\rtutils.dll
71B90000 Module C:\MF0\system32\samlib.dll
778F0000 Module C:\MF0\system32\setupapi.dll
Exception C000001D (Illegal Instruction)
Breakpoint at kernel32.7C801AE9
Import Redirection patched
018FD820 Breakpoint at 018FD820
018A2879 Breakpoint at 018A2879
7C810666 Breakpoint at kernel32.7C810666
018A8767 Breakpoint at 018A8767
VA of false OEP = 0052A59B
RVA of false OEP = 0012A59B

Ejecuto ArmInline y me reconoce el Import Elimination, Ojo lo ideal es reparar toda la iat antes de usarla, pero como estamos conociendo el objetivo, lo usare ahora mismo.

Import Elimination

Base Of Existing IAT: 0x17048E4

Length Of Existing IAT: 0x60C

New Base VA Of IAT: 0x5C8000

Rebase IAT

El tamaño que indica es lo que vi :

Address	Value	Comment
+5E4	018666E0	
+5E8	7C81390C	kernel32.GetFullPathNameA
+5EC	7E39FF33	USER32.CreateWindowExA
+5F0	018AB72D	
+5F4	018AB6D7	
+5F8	7C80A7E4	kernel32.GetLocalTime
+5FC	018AB75C	
+600	7C80B862	kernel32.GetFullPathNameW
+604	018AB824	
+608	7E39DAEA	USER32.DestroyWindow
+60C	00C20004	Copia_de.00C20004
+610	010C01E7	

Diciendo

```
----- Rebasing IAT -----  
Process memory buffered successfully.  
2849 DLL calls found total.  
Analysing...  
250 API functions referenced from 11 DLLs.  
Redirecting DLL references:  
2849 calls redirected total.  
Patching process...  
Process successfully patched.
```

Ejecuto Import Rec

Y coloco el oep, el comienzo y final de la iat

```
...rva:001C8028 mod:kernel32.dll ord:0174 name:GetModuleFileNameA
...rva:001C802C mod:kernel32.dll ord:0175 name:GetModuleFileNameW
...rva:001C8030 mod:kernel32.dll ord:02BF name:SetLastError
...rva:001C8034 mod:kernel32.dll ord:0097 name:EnterCriticalSection
...rva:001C8038 mod:kernel32.dll ord:02BF name:SetLastError
...rva:001C803C ptr:018665E0
...rva:001C8040 mod:kernel32.dll ord:0032 name:CloseHandle
...rva:001C8044 mod:kernel32.dll ord:0097 name:EnterCriticalSection
...rva:001C8048 mod:kernel32.dll ord:02BF name:SetLastError
```

Pero..hay entradas extrañas

Estos son los famosos "pendientes".

18665e0

Vamos a buscarlas a mano

Por ejemplo: 532fa8

00532FA1	E8 195F0000	CALL Copia_de.00538EBF	
00532FA6	59	POP ECX	
00532FA7	50	PUSH EAX	
00532FA8	FF15 40805C00	CALL DWORD PTR DS:[5C8040]	
00532FAE	85C0	TEST EAX,EAX	
00532FB0	75 0A	JNZ SHORT Copia_de.00532FBC	
00532FB2	FF15 B8825C00	CALL DWORD PTR DS:[5C82B8]	ntdll.RtlGetLastWin32Error
00532FB8	8BF8	MOV EDI,EAX	
00532FBA	EB 02	JMP SHORT Copia_de.00532FBE	
00532FBC	33FF	XOR EDI,EDI	
00532FBE	56	PUSH ESI	
00532FBF	E8 7A5E0000	CALL Copia_de.00538E3E	
00532FC4	8B5C	MOV EBX,ECX	
DS:[005C8040]=018666E0			

Vamos a 18666e0

018666E0	55	PUSH EBP	
018666E1	8BEC	MOV EBP,ESP	
018666E3	83EC 08	SUB ESP,8	
018666E6	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
018666E9	50	PUSH EAX	
018666EA	E8 71F9FFFF	CALL 01866060	
018666EF	83C4 04	ADD ESP,4	
018666F2	0FB6C8	MOVZX ECX,AL	
018666F5	85C9	TEST ECX,ECX	
018666F7	74 4C	JE SHORT 01866745	
018666F9	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
018666FC	52	PUSH EDX	
018666FD	E8 0EFAFFFF	CALL 01866110	
01866702	83C4 04	ADD ESP,4	
01866705	6BC0 0C	IMUL EAX,EAX,0C	
01866708	05 B0FC9301	ADD EAX,193FCB0	
0186670D	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
01866710	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866713	8B08	MOV ECX,DWORD PTR DS:[EAX]	
01866715	83E9 01	SUB ECX,1	
01866718	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]	
0186671B	890A	MOV DWORD PTR DS:[EDX],ECX	
0186671D	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866720	8338 00	CMP DWORD PTR DS:[EAX],0	
01866723	75 17	JNZ SHORT 0186673C	
01866725	8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]	
01866728	8B51 08	MOV EDX,DWORD PTR DS:[ECX+8]	
0186672B	52	PUSH EDX	
0186672C	FF15 54939101	CALL DWORD PTR DS:[1919354]	kernel32.CloseHandle
01866732	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866735	C740 08 FFFFFFFF	MOV DWORD PTR DS:[EAX+8],-1	
0186673C	C745 FC 01000000	MOV DWORD PTR SS:[EBP-4],1	
01866743	EB 29	JMP SHORT 0186676E	
01866745	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
01866748	51	PUSH ECX	
01866749	E8 32000000	CALL 01866780	
0186674E	83C4 04	ADD ESP,4	
01866751	0FB6D0	MOVZX EDX,AL	

Esta api es CloseHandle

Luego busco los intermodular calls, y procedo otra vez con estas llamadas extrañas

Si coloco new origin y comienzo a investigar la api, puedo mostrar error por lo que realizo lo mismo de ahora y con ese segundo comienzo a explorar.

Gracias a la herramienta, todas las redirecciones estan ordenadas igual ,asi que encontrando aquellas entradas, luego sera 1 ultima vez y seguiremos con nanomites

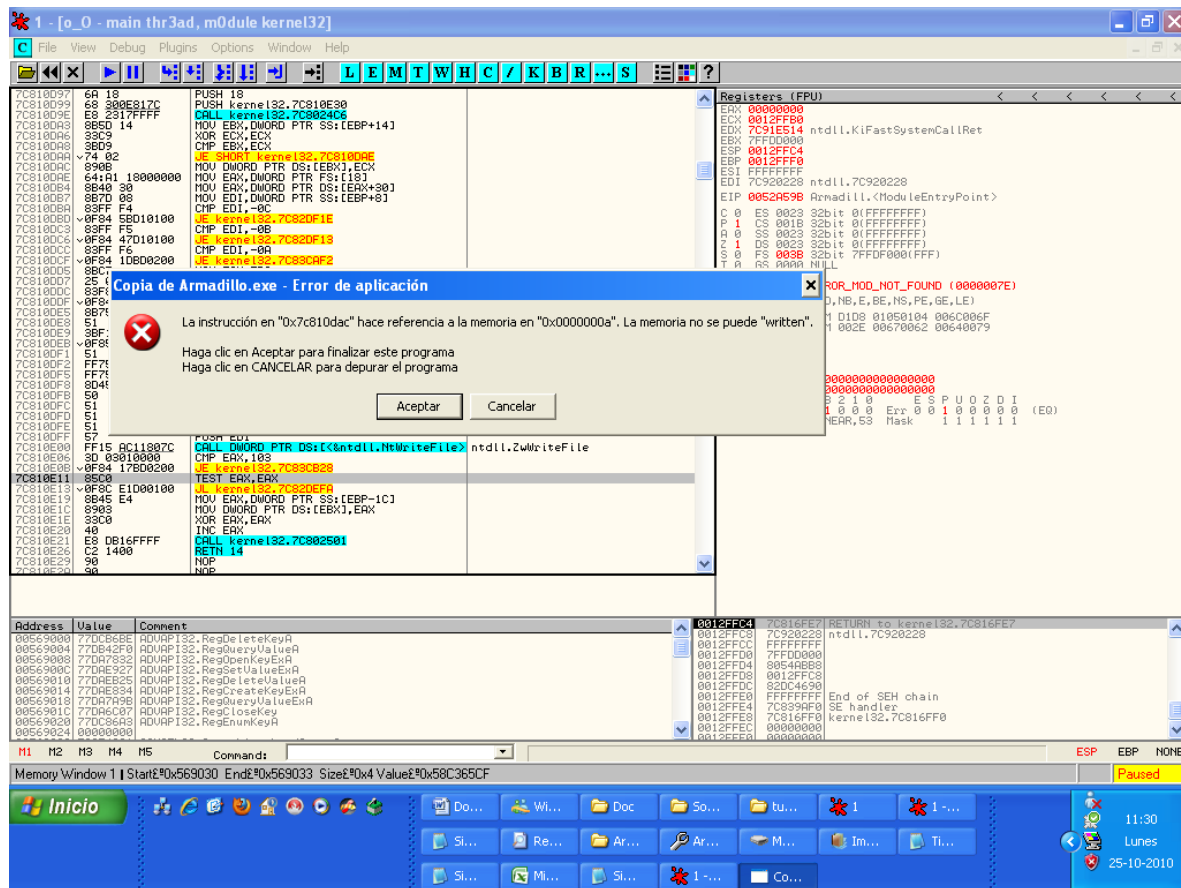
Osea

1) un olly que permite el otro unpacked y ese es el que dumpearemos

2) un olly que permite el otro unpacked->pero para buscar las entradas de la iat ->constantemente crasheando y encontrando la iat..sin ejecutar mas alla, pues tiene NANOMITES.

..

Aqui e el depudador que esta crasheando pero pillando las apis, esto no es con scripts, solo por pruebas:



Pero vemos la api en edx en algunas instancias

```

ECX 00000000
EDX 7C810097 kernel32.WriteFile
EBX 0000000A
ESP 0012F4B4 ASCII "(0"
EBP 0012F4E8
ESI 7342BE38

```

Asi vamos pillando una a una las apis

Y asi comienzo reconstruyendo

Nuestra tabla va tomando nombres de apis, y cuando terminemos, este programa estara a un paso de estar desempacado.

Address	Value	Comment
005C8000	7C80ADB0	kernel32.GetProcAddress
005C8004	7C801D77	kernel32.LoadLibraryA
005C8008	01863FA0	
005C800C	018640D0	
005C8010	01864230	
005C8014	018645D0	
005C8018	01864980	
005C801C	01864A80	
005C8020	01864AF0	
005C8024	01864BD0	
005C8028	018654D0	
005C802C	018659A0	
005C8030	01865CF0	
005C8034	7C810D97	kernel32.WriteFile
005C8038	018664A0	
005C803C	018665E0	
005C8040	018666E0	
005C8044	018667F0	
005C8048	01866B70	
005C804C	01867190	
005C8050	018673C0	
005C8054	01868E60	
005C8058	01868EA0	
005C805C	01868EE0	
005C8060	01868F20	
005C8064	01868F60	
005C8068	7E3D058A	USER32.MessageBoxA
005C806C	01869050	
005C8070	77DAE834	ADVAPI32.RegCreateKeyExA
005C8074	01869ED0	
005C8078	0186A2A0	
005C807C	0186A2B0	
005C8080	0186A2C0	

Si veo las viables hacen 38 lugares que apuntan a esos sectores de memoria asi que antes de 1 hora tenemos todo ok.

Quedando asi:

Gracias a

ECX	00000000	
EDX	7C810D97	kernel32.WriteFile
EBX	0000000A	
ESP	0012F4B4	ASCII "(0"
EBP	0012F4E8	
ESI	7342BE38	

Esto es

0040218F	5B	PUSH EBX	
00402190	FF15 34805C00	CALL DWORD PTR DS:[5C8034]	kernel32.WriteFile
00402196	8BE5	MOV ESP,EBP	
00402198	5D	POP EBP	
00402199	C3	RETN	
0040219A	CC	INT3	

Luego sigo y cuando vemos la dll arriba, load library, cuando se verifica

00406771	8BEC	PUSH EBP	
00406773	68 EC9F5600	MOV EBP,ESP	
00406775	FF15 04805C00	PUSH Copia_de.00569FEC	ASCII "ArmAccess.DLL"
00406777	A3 38215C00	CALL DWORD PTR DS:[5C8004]	kernel32.LoadLibraryA
00406779	68 E09F5600	MOV DWORD PTR DS:[5C2138],EAX	
0040677B	A1 38215C00	PUSH Copia_de.00569FEC	ASCII "CheckCode"
0040677D	50	MOV EAX,DWORD PTR DS:[5C2138]	
0040677F	FF15 00805C00	PUSH EAX	
00406781	A3 3C215C00	CALL DWORD PTR DS:[5C8000]	
00406783	68 D09F5600	MOV DWORD PTR DS:[5C213C],EAX	ASCII "SetUserString"
00406785	8B0D 38215C00	PUSH Copia_de.00569FD0	
00406787	51	MOV ECX,DWORD PTR DS:[5C2138]	
00406789	FF15 00805C00	PUSH ECX	kernel32.GetProcAddress
0040678B	A3 44215C00	CALL DWORD PTR DS:[5C8000]	
0040678D	68 C09F5600	MOV DWORD PTR DS:[5C2144],EAX	ASCII "GetString"
0040678F	8B15 38215C00	PUSH Copia_de.00569FC0	
00406791	52	MOV EDX,DWORD PTR DS:[5C2138]	
00406793	FF15 00805C00	PUSH EDX	kernel32.GetProcAddress
00406795	A3 40215C00	CALL DWORD PTR DS:[5C8000]	
00406797	68 AC9F5600	MOV DWORD PTR DS:[5C2140],EAX	ASCII "RawFingerprintInfo"
00406799	A1 38215C00	PUSH Copia_de.00569FAC	
0040679B	50	MOV EAX,DWORD PTR DS:[5C2138]	
0040679D	FF15 00805C00	PUSH EAX	
0040679F	A3 48215C00	CALL DWORD PTR DS:[5C8000]	kernel32.GetProcAddress
004067A1	5D	MOV DWORD PTR DS:[5C2148],EAX	
004067A3	C3	POP EBP	
004067A5	CC	RET	
004067A7	CC	INT3	

DS:[005C8000]=7C80ADB0 (kernel32.GetProcAddress)

Address	Value	Comment		0012F58C	01888769	RETURN to
005C8000	7C80ADB0	kernel32.GetProcAddress		0012F590	00400000	Copia_de.f
005C8004	7C801D77	kernel32.LoadLibraryA		0012F594	00000000	
				0012F598	00152258	

Este puede ser por regedit o puede ser por createfilew asi vamos entrando con enter y arreglando

004068A3	6A 00	PUSH 0	
004068A5	68 1F000200	PUSH 2001F	
004068A7	6A 00	PUSH 0	
004068A9	6A 00	PUSH 0	
004068AB	6A 00	PUSH 0	
004068AD	6A 00	PUSH 0	
004068AF	6A 00	PUSH 0	
004068B1	8B85 8CFEFFFF	MOV EAX,DWORD PTR SS:[EBP-174]	
004068B3	50	PUSH EAX	
004068B5	68 02000000	PUSH 80000002	
004068B7	FF15 70805C00	CALL DWORD PTR DS:[5C8070]	
004068B9	85C0	TEST EAX,EAX	
004068BB	74 0F	JMP SHORT Copia_de.004068C0	

Esto es un mensaje

0040CE2E	EB 16	JMP SHORT Copia_de.0040CE46	
0040CE30	6A 30	PUSH 30	
0040CE32	68 E8A05600	PUSH Copia_de.0056A0E8	ASCII "Under Construction"
0040CE34	68 BCA05600	PUSH Copia_de.0056A0BC	ASCII "Sorry, this command is not yet finished."
0040CE36	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0040CE38	51	PUSH ECX	
0040CE3A	FF15 68805C00	CALL DWORD PTR DS:[5C8068]	
0040CE3C	8B4D F4	MOV ECX,DWORD PTR SS:[EBP-C]	
0040CE3E	64:890D 00000000	MOV DWORD PTR FS:[0],ECX	
0040CE40	59	POP ECX	

Y veamos en el mensaje

00437856	6A 00	PUSH 0	
00437858	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
0043785A	52	PUSH EDX	
0043785C	FF15 54805C00	CALL DWORD PTR DS:[5C8054]	
0043785E	EB 02	JMP SHORT Copia_de.00437866	
00437860	EB CC	JMP SHORT Copia_de.00437832	
00437862	B8 01000000	MOV EAX,1	
00437864	EB 15	JMP SHORT Copia_de.00437882	
00437866	6A 00	PUSH 0	

Vemos como chequea los Breakpoints

01868E60	55	PUSH EBP	
01868E61	8BEC	MOV EBP,ESP	
01868E63	51	PUSH ECX	
01868E64	53	PUSH EBX	
01868E65	56	PUSH ESI	
01868E66	57	PUSH EDI	
01868E67	60	PUSHAD	
01868E68	8B15 E0569401	MOV EDX,DWORD PTR DS:[19456E0]	USER32.7
01868E6E	83C2 64	ADD EDX,64	
01868E71	B9 05000000	MOV ECX,5	
01868E76	003A CC	CMP BYTE PTR DS:[EDX],0CC	
01868E79	74 0A	JE SHORT 01868E85	
01868E7B	E2 F9	LOOPE SHORT 01868E76	
01868E7D	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
01868E80	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
01868E83	FFD2	CALL EDX	
01868E85	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
01868E88	61	POPAD	
01868E89	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
01868E8C	5F	POP EDI	
01868E8D	5E	POP ESI	
01868E8E	5B	POP EBX	
01868E8F	8BE5	MOV ESP,EBP	
01868E91	5D	POP EBP	
01868E92	C2 0800	RETN 8	

Y asi, Puede que no comente tanto esta parte, pero han pasado unos 20 minutos probando que apis son.

004377F8	52	PUSH EAX	
004377F9	FF15 E8825C00	CALL DWORD PTR DS:[5C82E8]	USER32.SetTimer
004377FE	C745 F0 00000000	MOV DWORD PTR SS:[EBP-10],0	
00437805	EB 09	JMP SHORT Copia_de.00437810	
00437807	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0043780A	83C0 01	ADD EAX,1	
0043780D	8945 F0	MOV DWORD PTR SS:[EBP-10],EAX	
00437810	817D F0 00010000	CMP DWORD PTR SS:[EBP-10],100	
00437817	7D 0C	JGE SHORT Copia_de.00437825	
00437819	8B4D F0	MOV ECX,DWORD PTR SS:[EBP-10]	
0043781C	51	PUSH ECX	
0043781D	FF15 88835C00	CALL DWORD PTR DS:[5C8388]	USER32.GetAsyncKeyState
00437823	EB E2	JMP SHORT Copia_de.00437887	
00437825	33C0	XOR EAX,EAX	
00437827	EB 59	JMP SHORT Copia_de.00437882	
00437829	C745 F0 00000000	MOV DWORD PTR SS:[EBP-10],0	
00437830	EB 09	JMP SHORT Copia_de.0043783B	
00437832	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]	
00437835	83C2 01	ADD EDX,1	
00437838	8955 F0	MOV DWORD PTR SS:[EBP-10],EDX	
0043783B	817D F0 00010000	CMP DWORD PTR SS:[EBP-10],100	
00437842	7D 22	JGE SHORT Copia_de.00437866	
00437844	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
00437847	5B	PUSH EAX	
00437849	FF15 88835C00	CALL DWORD PTR DS:[5C8388]	USER32.GetAsyncKeyState
0043784E	0FBFC8	MOVX ECX,AX	
00437851	83E1 01	AND ECX,1	
00437854	74 0E	JE SHORT Copia_de.00437864	
00437856	6A 00	PUSH 0	
00437858	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
0043785B	51	PUSH EDX	
0043785C	FF15 54805C00	CALL DWORD PTR DS:[5C8054]	USER32.EndDialog
00437862	EB 02	JMP SHORT Copia_de.00437866	
00437864	EB CC	JMP SHORT Copia_de.00437832	
00437866	B8 01000000	MOV EAX,1	
00437868	EB 15	JMP SHORT Copia_de.00437882	
0043786D	6A 00	PUSH 0	
0043786F	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
00437872	5B	PUSH EAX	
00437873	FF15 54805C00	CALL DWORD PTR DS:[5C8054]	USER32.EndDialog
00437879	B8 01000000	MOV EAX,1	
0043787E	EB 02	JMP SHORT Copia_de.00437882	
00437880	33C0	XOR EAX,EAX	
00437882	8B4D F4	MOV ECX,DWORD PTR SS:[EBP-C]	

Algunos son mas faciles que otros

0186A7C0	55	PUSH EBP	
0186A7C1	8BEC	MOV EBP,ESP	
0186A7C3	51	PUSH ECX	
0186A7C4	8B45 18	MOV EAX,DWORD PTR SS:[EBP+18]	
0186A7C7	50	PUSH EAX	
0186A7C8	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]	
0186A7CB	51	PUSH ECX	
0186A7CC	8B55 10	MOV EDX,DWORD PTR SS:[EBP+10]	
0186A7CF	51	PUSH EDX	
0186A7D0	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
0186A7D3	50	PUSH EAX	
0186A7D4	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0186A7D7	51	PUSH ECX	
0186A7D8	FF15 60949101	CALL DWORD PTR DS:[1919460]	USER32.DialogBoxParamA
0186A7DE	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
0186A7E1	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
0186A7E4	8BE5	MOV ESP,EBP	
0186A7E6	5D	POP EBP	
0186A7E7	C2 1400	RETN 14	
0186A7E8	75	JNZ	

018666E0	55	PUSH EBP	
018666E1	8BEC	MOV EBP,ESP	
018666E3	83EC 08	SUB ESP,8	
018666E6	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
018666E9	50	PUSH EAX	
018666EA	E8 71F9FFFF	CALL 01866060	
018666EF	83C4 04	ADD ESP,4	
018666F2	0FB6C8	MOVZX ECX,AL	
018666F5	85C9	TEST ECX,ECX	
018666F7	74 4C	JE SHORT 01866745	
018666F9	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
018666FC	52	PUSH EDX	
018666FD	E8 0EFAFFFF	CALL 01866110	
01866702	83C4 04	ADD ESP,4	
01866705	68C0 0C	IMUL EAX,EAX,0C	
01866708	05 B0FC9301	ADD EAX,193FCB0	
0186670D	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
01866710	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866713	8B08	MOV ECX,DWORD PTR DS:[EAX]	
01866715	83E9 01	SUB ECX,1	
01866718	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]	
0186671B	890A	MOV DWORD PTR DS:[EDX],ECX	
0186671D	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866720	8338 00	CMPL DWORD PTR DS:[EAX],0	
01866723	75 17	JNZ SHORT 01866730	
01866725	8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]	
01866728	8B51 08	MOV EDX,DWORD PTR DS:[ECX+8]	
0186672B	52	PUSH EDX	
0186672C	FF15 54939101	CALL DWORD PTR DS:[1919354]	
01866732	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01866735	C740 08 FFFFFFFF	MOV DWORD PTR DS:[EAX+8],-1	
0186673C	C745 FC 01000000	MOV DWORD PTR SS:[EBP-4],1	
01866743	EB 29	JMP SHORT 0186676E	
01866745	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
01866748	51	PUSH ECX	
01866749	E8 32000000	CALL 01866780	
0186674E	83C4 04	ADD ESP,4	
01866751	0FB6D0	MOVZX EDX,AL	
01866754	85D2	TEST EDX,EDX	

kernel32.CloseHandle

018654D0	55	PUSH EBP	
018654D1	8BEC	MOV EBP,ESP	
018654D3	51	PUSH ECX	
018654D4	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
018654D7	50	PUSH EAX	
018654D8	E8 F3010000	CALL 01865600	
018654DD	83C4 04	ADD ESP,4	
018654E0	0FB6C8	MOVZX ECX,AL	
018654E3	85C9	TEST ECX,ECX	
018654E5	74 25	JE SHORT 01865500	
018654E7	8B55 20	MOV EDX,DWORD PTR SS:[EBP+20]	
018654EA	52	PUSH EDX	
018654EB	8B45 1C	MOV EAX,DWORD PTR SS:[EBP+1C]	
018654EE	50	PUSH EAX	
018654EF	8B4D 18	MOV ECX,DWORD PTR SS:[EBP+18]	
018654F2	51	PUSH ECX	
018654F3	8B55 14	MOV EDX,DWORD PTR SS:[EBP+14]	
018654F6	52	PUSH EDX	
018654F7	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
018654FA	50	PUSH EAX	
018654FB	8B4D 0C	MOV ECX,DWORD PTR SS:[EBP+C]	
018654FE	51	PUSH ECX	
018654FF	E8 3C000000	CALL 01865540	
01865504	83C4 18	ADD ESP,18	
01865507	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
0186550A	EB 25	JMP SHORT 01865531	
0186550C	8B55 20	MOV EDX,DWORD PTR SS:[EBP+20]	
0186550F	52	PUSH EDX	
01865510	8B45 1C	MOV EAX,DWORD PTR SS:[EBP+1C]	
01865513	50	PUSH EAX	
01865514	8B4D 18	MOV ECX,DWORD PTR SS:[EBP+18]	
01865517	51	PUSH ECX	
01865518	8B55 14	MOV EDX,DWORD PTR SS:[EBP+14]	
0186551B	52	PUSH EDX	
0186551C	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
0186551F	50	PUSH EAX	
01865520	8B4D 0C	MOV ECX,DWORD PTR SS:[EBP+C]	
01865523	51	PUSH ECX	
01865524	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
01865527	52	PUSH EDX	
01865528	FF15 2C939101	CALL DWORD PTR DS:[191932C]	
0186552E	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
01865531	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	

kernel32.CreateFileA

01865CF0	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
01865D00	50	PUSH EAX	
01865D01	E8 5A030000	CALL 01866060	
01865D06	83C4 04	ADD ESP,4	
01865D09	0FB6C8	MOVZX ECX,AL	
01865D0E	85C9	TEST ECX,ECX	
01865D0E	0F34 1E030000	JBE 01866030	
01865D14	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
01865D17	52	PUSH EDX	
01865D18	E8 F3030000	CALL 01866110	
01865D1D	83C4 04	ADD ESP,4	
01865D20	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
01865D23	C745 F4 00000000	MOV DWORD PTR SS:[EBP-C],0	
01865D2A	6A 00	PUSH 0	
01865D2C	FF15 F4909101	CALL DWORD PTR DS:[19190F4]	ntdll.RtlSetLastWin32Error
01865D32	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01865D35	68C0 0C	IMUL EAX,EAX,0C	
01865D38	33C9	XOR ECX,ECX	
01865D3A	8388 B0FC9301 0	CMPL DWORD PTR DS:[EAX+193FCB0],0	
01865D41	0F9FC1	SETG CL	
01865D44	0FB6D1	MOVZX EDX,CL	
01865D47	85D2	TEST EDX,EDX	
01865D49	75 14	JNZ SHORT 01865D5F	
01865D4B	6A 00	PUSH 0	
01865D4D	FF15 F4909101	CALL DWORD PTR DS:[19190F4]	ntdll.RtlSetLastWin32Error
01865D53	C745 FC 00000000	MOV DWORD PTR SS:[EBP-4],0	
01865D5A	E9 D1020000	JMP 01866030	
01865D5F	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
01865D62	68C0 0C	IMUL EAX,EAX,0C	
01865D65	8B88 B4FC9301	MOV ECX,DWORD PTR DS:[EAX+193FCB4]	
01865D6B	894D E8	MOV DWORD PTR SS:[EBP-18],ECX	
01865D6E	8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	
01865D71	3B15 10FC9301	CMPL EDX,DWORD PTR DS:[193FC18]	
01865D77	72 6E	JBE SHORT 01865DE7	
01865D79	8B45 E8	MOV EAX,DWORD PTR SS:[EBP-18]	
01865D7C	8345 10	ADD EAX,DWORD PTR SS:[EBP+10]	
01865D7F	3B05 14FC9301	CMPL EAX,DWORD PTR DS:[193FC14]	
01865D85	73 60	JNB SHORT 01865DE7	Copia_de.0045B000
01865D87	8B4D 18	MOV ECX,DWORD PTR SS:[EBP+18]	
01865D8A	51	PUSH ECX	
01865D8B	8B55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	
01865D8E	52	PUSH EDX	
01865D8F	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
01865D92	50	PUSH EAX	
01865D93	8B4D 0C	MOV ECX,DWORD PTR SS:[EBP+C]	
01865D96	51	PUSH ECX	
01865D97	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]	
01865D9A	68D2 0C	IMUL EDX,EDX,0C	
01865D9D	8B82 B8FC9301	MOV EAX,DWORD PTR DS:[EDX+193FCB8]	
01865DA3	50	PUSH EAX	
01865DA4	FF15 30939101	CALL DWORD PTR DS:[1919330]	kernel32.ReadFile
01865DA8	8945 F8	MOV DWORD PTR SS:[EBP-10],EAX	
01865DA9	0000 0000	OR DWORD PTR DS:[EBP-14],0	
DS:[01919330]=7C80180E (kernel32.ReadFile)			

01866547	EB 00	OR EAX,FFFFFFFF	
01866548	83C8 FF	OR EAX,FFFFFFFF	
0186654E	E9 83000000	JMP 018665D6	
01866553	837D F8 00	CMPL DWORD PTR SS:[EBP-8],0	
01866557	7D 12	JBE SHORT 01866568	
01866559	68 33000000	PUSH 33	
0186655E	FF15 F4909101	CALL DWORD PTR DS:[19190F4]	ntdll.RtlSetLastWin32Error
01866564	83C8 FF	OR EAX,FFFFFFFF	
01866567	EB 6D	JMP SHORT 018665D6	
01866569	EB 45	JMP SHORT 018665B0	
0186656B	8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]	
0186656E	3B0D 18FC9301	CMPL ECX,DWORD PTR DS:[193FC18]	Copia_de.0045B000
01866574	73 2B	JNB SHORT 018665A1	
01866576	6A 00	PUSH 0	
01866578	8B55 10	MOV EDX,DWORD PTR SS:[EBP+10]	
0186657B	52	PUSH EDX	
0186657C	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
0186657F	50	PUSH EAX	
01866580	8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
01866583	68C9 0C	IMUL ECX,ECX,0C	
01866586	8B91 B8FC9301	MOV EDX,DWORD PTR DS:[ECX+193FCB8]	
0186658C	52	PUSH EDX	
0186658D	FF15 4C919101	CALL DWORD PTR DS:[191914C]	kernel32.SetFilePointer
01866593	8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
01866596	68C9 0C	IMUL ECX,ECX,0C	
01866599	8981 B4FC9301	MOV DWORD PTR DS:[ECX+193FCB4],EAX	
0186659F	EB 0F	JMP SHORT 018665B0	
018665A1	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
018665A4	68D2 0C	IMUL EDX,EDX,0C	

Y otras mas dificiles Comenzamos con una menor

018665E0	55	PUSH EBP	
018665E1	8BEC	MOV EBP,ESP	
018665E3	51	PUSH ECX	
018665E4	53	PUSH EBX	
018665E5	56	PUSH ESI	
018665E6	57	PUSH EDI	
018665E7	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
018665EA	50	PUSH EAX	
018665EB	E8 70FAFFFF	CALL 01866060	
018665F0	83C4 04	ADD ESP,4	
018665F3	0FB6C8	MOUZ% ECX,AL	
018665F6	85C9	TEST ECX,ECX	
018665F8	74 1E	JE SHORT 01866618	
018665FA	837D 0C 00	CMP DWORD PTR SS:[EBP+C],0	
018665FE	74 09	JE SHORT 01866609	
01866600	8B55 0C	MOV EDX,DWORD PTR SS:[EBP+C]	
01866603	C702 00000000	MOV DWORD PTR DS:[EDX],0	
01866609	A1 14FC9301	MOV EAX,DWORD PTR DS:[193FC14]	
0186660E	0305 34FC9301	ADD EAX,DWORD PTR DS:[193FC34]	
01866614	EB 18	JMP SHORT 01866631	
01866616	EB 19	JMP SHORT 01866631	
01866618	60	PUSHAD	
01866619	8B15 F0569401	MOV EDX,DWORD PTR DS:[19456F0]	kernel32.7C810A23
0186661F	83C2 64	ADD EDX,64	
01866622	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
01866625	FF75 08	PUSH DWORD PTR SS:[EBP+8]	

Ese lugar apunta a

7C810A17	90	NOP	
7C810A18	90	NOP	
7C810A19	8BFF	MOV EDI,EDI	
7C810A1B	55	PUSH EBP	
7C810A1C	8BEC	MOV EBP,ESP	
7C810A1E	83EC 20	SUB ESP,20	
7C810A21	6A 05	PUSH 5	
7C810A23	6A 18	PUSH 18	InfoClass = FileStandardInformation Bufsize = 18 (24.)
7C810A25	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
7C810A28	50	PUSH EAX	
7C810A29	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	Buffer
7C810A2C	50	PUSH EAX	pStatusBlock
7C810A2D	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hFile
7C810A30	FF15 1810807C	CALL DWORD PTR DS:[<&ntdl.NtQueryInformationFile>]	ZwQueryInformationFile
7C810A36	85C0	TEST EAX,EAX	
7C810A38	74 0C	JE kernel32.7C80C7F	

Que corresponde a

Address	Disassembly
7C810A19	MOV EDI,EDI
7C810A95	CALL kernel32.GetFileSizeEx

Obviamente debe ser getfilesize, pues lo que veiamos era la segunda parte ..por eso digo que hay que darse tiempo..

0044E9B9	51	PUSH ECX	
0044E9BA	8D95 F0DFDFFF	LEA EDX,DWORD PTR SS:[EBP-210]	
0044E9C0	52	PUSH EDX	
0044E9C1	68 F0A95600	PUSH Copia_de.0056AAF0	ASCII ".chn\shell\open\command"
0044E9C6	68 00000000	PUSH 00000000	ADVAPI32.RegQueryValueA
0044E9CB	FF15 74805C00	CALL DWORD PTR DS:[5C8074]	
0044E9D1	85C0	TEST EAX,EAX	
0044E9D3	75 04	JNZ SHORT Copia_de.0044E9D9	
0044E9D5	C645 FF 01	MOV BYTE PTR SS:[EBP-1],1	
0044E9D9	C745 F8 00010000	MOV DWORD PTR SS:[EBP-8],100	
0044E9E0	0FB645 FF	MOUZ% EAX,BYTE PTR SS:[EBP-1]	

0049A443	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
0049A446	52	PUSH EDX	
0049A447	8D45 88	LEA EAX,DWORD PTR SS:[EBP-78]	
0049A44A	50	PUSH EAX	
0049A44B	6A 00	PUSH 0	
0049A44D	FF15 64805C00	CALL DWORD PTR DS:[5C8064]	kernel32.GetModuleHandleA
0049A453	50	PUSH EAX	
0049A454	E8 C72F0600	CALL Copia_de.004FD420	
0049A459	83C4 14	ADD ESP,14	
0049A45C	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX	
0049A45F	C745 FC FFFFFFFF	MOV DWORD PTR SS:[EBP-4],-1	
0049A466	8D4D 88	LEA ECX,DWORD PTR SS:[EBP-78]	
0049A469	E8 2279F6FF	CALL Copia_de.00401D90	
0049A46E	837D E8 00	CMPL DWORD PTR SS:[EBP-18],0	
0049A472	CC	INT3	

De todas esta me sorprendio, asi que debemos prepararnos bien:

en este caso se omite la api y usa el parametro ya establecido:

0186A2AE	CC	INT3	
0186A2AF	CC	INT3	
0186A2B0	55	PUSH EBP	
0186A2B1	8BEC	MOV EBP,ESP	
0186A2B3	A1 3C579401	MOV EAX,DWORD PTR DS:[194573C]	
0186A2B8	5D	POP EBP	
0186A2B9	C3	RETN	
0186A2BA	CC	INT3	
0186A2BB	CC	INT3	
0186A2BC	CC	INT3	
0186A2BD	CC	INT3	
0186A2BE	CC	INT3	
0186A2BF	CC	INT3	
0186A2C0	55	PUSH EBP	
0186A2C1	8BEC	MOV EBP,ESP	
0186A2C3	A1 40579401	MOV EAX,DWORD PTR DS:[1945740]	
0186A2C8	5D	POP EBP	
0186A2C9	C3	RETN	
0186A2CA	CC	INT3	

[0194573C]=00152310		
---------------------	--	--

Address	Value	Comment
0194573C	00152310	ASCII ""C:\archivos de programa\SoftwarePassport8.0.0.8\Copia de Armadill
01945740	000004E4	
01945744	00A3FF5A	Copia_de.00A3FF5A
01945748	4CC57035	
0194574C	00010000	INT3CODE "-...-...\"

Tenemos la ruta y el nombre del archivo

Esto es

004FE82A	FF15 7C805C00	CALL DWORD PTR DS:[5C807C]	kernel32.GetCommandLineA
004FE830	5D	PUSH EAX	
004FE831	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
004FE834	E3 F76EF1FF	CALL Copia_de.00415730	
004FE839	C745 FC 00000000	MOV DWORD PTR SS:[EBP-4],0	
004FE840	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
004FE843	83C8 01	OR EAX,1	

Y tambien luego para lo de abajo 4e4 ->, como es vc++ es getacp

0052FCA0	FF15 80805C00	CALL DWORD PTR DS:[5C8080]	kernel32.GetACP
0052FCB6	75 0B	JMP SHORT Copia_de.0052FCB3	
0052FCB8	33FE F4	CMP ESI,4	
0052FCBB	75 12	JNZ SHORT Copia_de.0052FCBF	
0052FCBD	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0052FCB0	8B40 04	MOV EAX,DWORD PTR DS:[EAX+4]	

Otras formas sin api, pero usando getProcAddress el famoso IsdebuggerPresent

0186A820	55	PUSH EBP	
0186A821	8BEC	MOV EBP,ESP	
0186A823	55	PUSH ESI	
0186A824	B8 0A000000	MOV EAX,0A	
0186A829	C1E0 02	SHL EAX,2	
0186A82C	8B00 8CCF9301	MOV ECX,DWORD PTR DS:[193CF8C]	Copia_de.006896D0
0186A832	8B15 8CCF9301	MOV EDX,DWORD PTR DS:[193CF8C]	Copia_de.006896D0
0186A838	8B35 8CCF9301	MOV ESI,DWORD PTR DS:[193CF8C]	Copia_de.006896D0
0186A83E	8B76 34	MOV ESI,DWORD PTR DS:[ESI+34]	
0186A841	33B2 94000000	XOR ESI,DWORD PTR DS:[EDX+84]	
0186A847	339401	XOR ESI,DWORD PTR DS:[ECX+EAX]	
0186A84A	83E6 14	AND ESI,14	
0186A84D	74 04	JE SHORT 0186A853	
0186A84F	33C0	XOR EAX,EAX	
0186A851	EB 38	JMP SHORT 0186A88B	
0186A853	833D 20359401	CMP DWORD PTR DS:[1943520],0	
0186A85A	75 1C	JNZ SHORT 0186A878	
0186A85C	68 44B9101	PUSH 191B944	
0186A861	68 10C9101	PUSH 1910C18	
0186A866	FF15 90909101	CALL DWORD PTR DS:[1919090]	ASCII "IsDebuggerPresent"
0186A86C	5D	PUSH EAX	ASCII "kernel32.dll"
0186A86D	FF15 94909101	CALL DWORD PTR DS:[1919094]	kernel32.GetModuleHandleA
0186A873	A3 20359401	MOV DWORD PTR DS:[1943520],EAX	
0186A878	833D 20359401	CMP DWORD PTR DS:[1943520],0	kernel32.GetProcAddress
0186A87F	74 08	JE SHORT 0186A889	
0186A881	FF15 20359401	CALL DWORD PTR DS:[1943520]	
0186A887	EB 02	JMP SHORT 0186A88B	
0186A889	33C0	XOR EAX,EAX	
0186A88B	5E	POP ESI	
0186A88C	5D	POP EBP	
0186A88D	C3	RETN	

Y la curiosidad cuando no tiene la api, en la parte inferior y tampoco es del estilo que emula, o rescata informacion tenemos que hace como una mini desecripcion :

Vean como se logra apreciar el call

0186485C	8B55 AC	MOV EDX,DWORD PTR SS:[EBP-54]
0186485F	833A 00	CMP DWORD PTR DS:[EDX],0
01864862	74 1F	JE SHORT 01864883
01864864	8B45 AC	MOV EAX,DWORD PTR SS:[EBP-54]
01864867	8B08	MOV ECX,DWORD PTR DS:[EAX]
01864869	8B11	MOV EDX,DWORD PTR DS:[ECX]
0186486B	0315 705A9401	ADD EDX,DWORD PTR DS:[1945A701]
01864871	8B45 AC	MOV EAX,DWORD PTR SS:[EBP-54]
01864874	8B08	MOV ECX,DWORD PTR DS:[EAX]
01864876	8911	MOV DWORD PTR DS:[ECX],EDX
01864878	8B55 AC	MOV EDX,DWORD PTR SS:[EBP-54]
0186487B	83C2 04	ADD EDX,4
0186487E	8955 AC	MOV DWORD PTR SS:[EBP-54],EDX
01864881	EB D9	JMP SHORT 0186485C
01864883	52	PUSH EDX
01864884	66:87C9	XCHG CX,CX
01864887	5A	POP EDX
01864888	C745 FC 00000000	MOV DWORD PTR SS:[EBP-4],0
0186488F	FF15 18919101	CALL DWORD PTR DS:[19191181]
01864895	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX
01864898	EB 0D	JMP SHORT 018648A7
0186489A	C745 FC FFFFFFFF	MOV DWORD PTR SS:[EBP-4],-1
018648A1	B8 AE488601	MOV EAX,18648AE
018648A6	C3	RETN
018648A7	C745 FC FFFFFFFF	MOV DWORD PTR SS:[EBP-4],-1
018648AE	EB 03	JMP SHORT 018648B3
018648B0	D6	SALC
018648B1	D6	SALC
018648B2	A6	CMPS BYTE PTR DS:[ESI],BYTE PTR ES:[EDI]

Terminando con la iat luego de una hora aproximada repare la iat destruction

005C8000	7C80ADB0	kernel32.GetProcAddress
005C8004	7C801D77	kernel32.LoadLibraryA
005C8008	7C80ABEE	kernel32.FreeLibrary
005C800C	7C814B02	kernel32.GetEnvironmentVariableA
005C8010	7C81CF6B	kernel32.GetEnvironmentStringsA
005C8014	7C812F18	kernel32.GetEnvironmentStringsW
005C8018	7C81CDEA	kernel32.ExitProcess
005C801C	7C801E16	kernel32.TerminateProcess
005C8020	7C80C068	kernel32.ExitThread
005C8024	7C810647	kernel32.CreateThread
005C8028	7C801A24	kernel32.CreateFileA
005C802C	7C810770	kernel32.CreateFileW
005C8030	7C80180E	kernel32.ReadFile
005C8034	7C810D97	kernel32.WriteFile
005C8038	7C810B9E	kernel32.SetFilePointer
005C803C	7C810A87	kernel32.GetFileSize
005C8040	7C809B57	kernel32.CloseHandle
005C8044	7C810E61	kernel32.GetFileType
005C8048	7C809478	kernel32.CreateFileMappingA
005C804C	7C80B915	kernel32.MapViewOfFile
005C8050	7C80B984	kernel32.UnmapViewOfFile
005C8054	7E3A59C9	USER32.EndDialog
005C8058	7C80FD3D	kernel32.GlobalAlloc
005C805C	7C80FE92	kernel32.GlobalUnlock
005C8060	7C80CCA7	kernel32.SetHandleCount
005C8064	7C80B6B1	kernel32.GetModuleHandleA
005C8068	7E3D058A	USER32.MessageBoxA
005C806C	7E3B212B	USER32.GetWindowTextA
005C8070	77DAE834	ADVAPI32.RegCreateKeyExA
005C8074	77DB42F0	ADVAPI32.RegQueryValueA
005C8078	7C80ABD1	kernel32.GetProcessHeap
005C807C	7C812F2D	kernel32.GetCommandLineA
005C8080	7C809925	kernel32.GetACP
005C8084	7C80BE99	kernel32.FindResourceA
005C8088	7C82F7BC	kernel32.FormatMessageA
005C808C	7E3BC7A3	USER32.CreateDialogParamA
005C8090	7E3BB10C	USER32.DialogBoxParamA
005C8094	7C8130A3	kernel32.IsDebuggerPresent
005C8098	00000000	
005C809C	58C265CF	COMCTL32.InitCommonControls

Otra cosa interesante es que el create File A, y createFile W, estaban cercanos, asi tambien otras apis, GetCOMmandLine, lo descubri asi.

Sigamos, como terminamos la tablita destruida ahora debemos apuntarla denuevo para que este todo ordenado

```
----- Hebasing IAT -----
Process memory buffered successfully.
2849 DLL calls found total.
Analysing...
249 API functions referenced from 10 DLLs.
Redirecting DLL references:
2849 calls redirected total.
Patching process...
Process successfully patched.
```

Ahora bien

Despues vienen los nanomites..Use Arminline , y guarde el .nan, pero yo lo iba parchando en cuanto me daba la excepcion

0044A358	FF15 40805C00	CALL DWORD PTR DS:[5C8040]	kernel32.CloseHandle
0044A35E	0FB60D F2285C00	MOVZX ECX, BYTE PTR DS:[5C28F2]	
0044A365	85C9	TEST ECX, ECX	
0044A367	74 18	JE SHORT Copia_de.0044A381	nanomite*****
0044A369	68 B0B00000	PUSH 0BB8	
0044A36E	FF15 58815C00	CALL DWORD PTR DS:[5C8158]	kernel32.Sleep
0044A374	8B95 54FFFFFF	MOV EDX, DWORD PTR SS:[EBP-AC]	
0044A37A	52	PUSH EAX	

Pero 6 mil parches, pues ni modo.

```
----- Nanomites -----
Process memory buffered successfully.
Initialising...
43017 potential INT3 found.
36901 useless Nanomites discarded.
43017 INT3 found, 6116 successfully analysed.
```

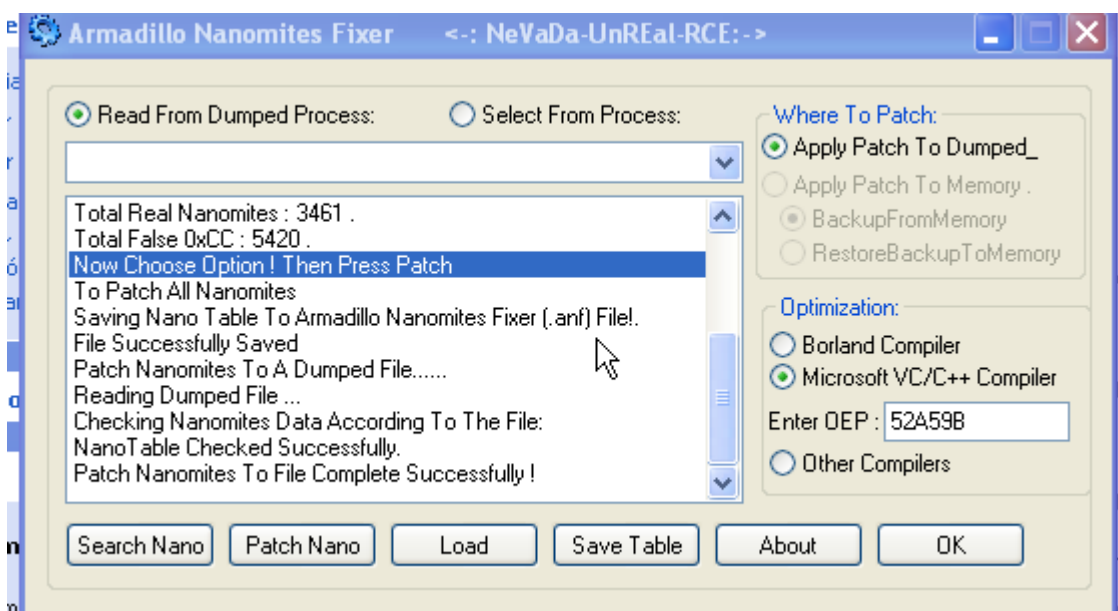
Coloco a dumppear el proceso y coloco la iat , pero me llevo la sorpresa que al agregar la nueva seccion No es Bien colocada por Armadillo-ArmInline-0.96ff y armaggedon tambien lo hace un poco mejor, pero recuerde que tenia en mi repertorio de tools de armadillo una tool creada por nevada y que venia con armaggedon

Yo de primera iba parchando los saltos guiandome con la tabla de nanomites,

00402453	0FB6C0	MOVZX EAX,AL	
00402456	85C0	TEST EAX,EAX	
00402458	74 04	JE SHORT con_nano.0040245E	
0040245A	C645 DB 01	MOV BYTE PTR SS:[EBP-25],1	
0040245E	C745 E4 00000000	MOV DWORD PTR SS:[EBP-1C],0	
00402465	C645 FC 03	MOV BYTE PTR SS:[EBP-4],3	
00402469	68 849F5600	PUSH con_nano.00569F84	
0040246E	8D8D 3CFDFFFF	LEA ECX,DWORD PTR SS:[EBP-2C4]	
00402474	E8 B7320100	CALL con_nano.00415730	
00402479	C645 FC 04	MOV BYTE PTR SS:[EBP-4],4	
0040247D	68 789F5600	PUSH con_nano.00569F78	
00402482	8D8D DCFCFFFF	LEA ECX,DWORD PTR SS:[EBP-324]	
00402488	E8 A3810F00	CALL con_nano.004FA630	ASCII "EXTRAINFO"

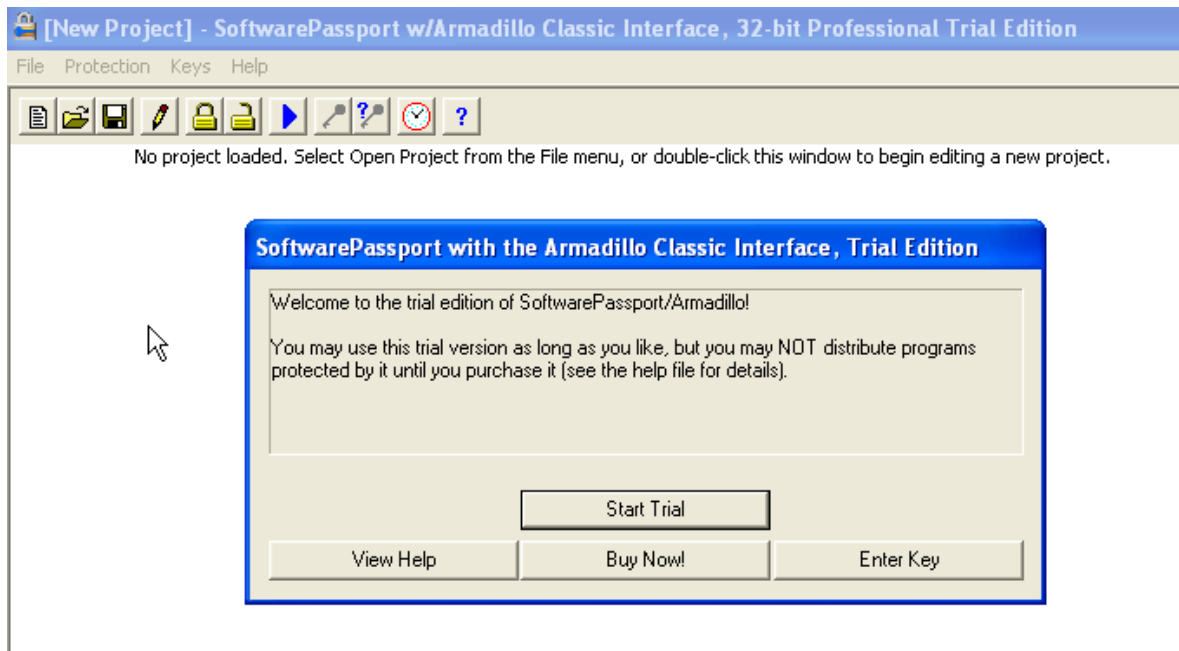
pero a los 20 minutos recorde este programa, normalmente lee el padre (original) y el hijo(unpacked)

- 1)colocar el original
- 2) colocar el dumped con la iat reparada ,3) colocar el oep
- 4)Search Nano 5) guardo la tabla
- 6)patch nano...el resultado:



Unpacked. Luego de 1 hora y media aprox





Como soy curioso, quise ver como lo habia hecho Steven y ahora que logre mi unpacked

veo que su unpacked, esta optimizado, pues el tamaño es mucho menor al que tengo, pero estoy tranquilo que el resultado debe ser similar.

Y veo "no cambio el ep para el injerto" sino que uso el "segundo jmp"

0052A59B	\$ E8 5A9B0000	CALL Armadillo.005340FA
0052A5A0	✓ E9 FBE00300	JMP Armadillo.005686A0
0052A5A5	\$ 6A 0C	PUSH 0C
0052A5A7	. 68 10B45B00	PUSH Armadillo.005BB410

Para quien quiere saber que parte es la que buscaba yo, es lo que define a algunas variables de entorno, sugiero lean la teoria 971-Reparando_un_dumpeado_de_Armadillo_por_karmany.rar

-> **GetEnvironmentVariableA** Los cuales normalmente acceden a **ArmAccess.dll**.

Esto tambien lo vimos en 1028-armadillo+aspack_parte 2 solid.rar

Esto es la unica diferencia que veo entre un unpacked y el original, la inyeccion de todas las variables.

005686A0	60	PUSHAD	
005686A1	9C	PUSHFD	
005686A2	8B1D 4C925600	MOV EBX,DWORD PTR DS:[<%KERNEL32.SetEnv	kernel32.SetEnvironmentVariableA
005686A8	68 10875600	PUSH Armadill.00568710	Value = "stephenteht (RES)"
005686AD	68 08AE5600	PUSH Armadill.0056AE08	VarName = "USERNAME"
005686B2	FFD3	CALL EBX	SetEnvironmentVariableA
005686B4	68 22875600	PUSH Armadill.00568722	Value = "TEAM RESURRECTION"
005686B9	68 509F5600	PUSH Armadill.00569F50	VarName = "USERKEY"
005686BE	FFD3	CALL EBX	SetEnvironmentVariableA
005686C0	68 7CA95600	PUSH Armadill.0056AA7C	Value = "Professional"
005686C5	68 00AE5600	PUSH Armadill.0056AE08	VarName = "VERSION"
005686CA	FFD3	CALL EBX	SetEnvironmentVariableA
005686CC	68 41875600	PUSH Armadill.00568741	Value = "Yes indeed!"
005686D1	68 F8AD5600	PUSH Armadill.0056ADF8	VarName = "PAIDFOR"
005686D6	FFD3	CALL EBX	SetEnvironmentVariableA
005686D8	68 35875600	PUSH Armadill.00568735	Value = "2008.07.25"
005686DD	68 649F5600	PUSH Armadill.00569F64	VarName = "KEYCREATED"
005686E2	FFD3	CALL EBX	SetEnvironmentVariableA
005686E4	68 48AF5600	PUSH Armadill.0056AF48	Value = "True"
005686E9	68 50AF5600	PUSH Armadill.0056AF50	VarName = "U8"
005686EE	FFD3	CALL EBX	SetEnvironmentVariableA
005686F0	68 4E875600	PUSH <Armadill.apuromaf05>	Value = "1"
005686F5	68 789F5600	PUSH Armadill.00569F78	VarName = "EXTRAINFO"
005686FA	FFD3	CALL EBX	SetEnvironmentVariableA
005686FC	9D	POPPD	
005686FD	61	POPAD	
005686FE	^E9 B81CFCFF	JMP Armadill.0052A3BB	

Si ejecuto , muestra como registrado, pero si le quito el SetenvironmentVariableA, trabaja como trial, asi que veo que no hay mas parches de por medio.

Asi que nos podemos guiar de la misma forma y tener un Unpacked+registro de variables=pro version.

Comienzo ahora con lo olvidado por steven



SoftwarePassport.exe
17-09-2010 23:45
SoftwarePassport

```
<- 28-10-2010 12:12:45 - [2.0] ->  
C:\archivos de programa\SoftwarePasspo:  
Protected Armadillo  
<-Find Protect  
Protection system (Professional)  
<Protection Options>  
Standard protection or Minimum protect:  
<Backup Key Options>  
Main Key Only, No Backup Keys  
<Compression Options>  
Best/Slowest Compression  
<Other Options>  
Store Environment Vars Externally  
<-Find Version  
Version 8.00 01-10-2010  
<- Elapsed Time 00h 00m 05s 375ms ->
```

Le coloco

- Armadillo 3.xx 4.xx 5.xx 6.xx CopyMemII+Debug Blocker IAT Recover.txt
- Armadillo 3.xx 4.xx 5.xx 6.xx Unpack.txt
- Armadillo Detach.txt

Le coloco version 6 y que tiene debug bloquer (porciacaso la nag que veo).

00421308	68 04174200	PUSH Software.00421704	DEP
0042130D	E8 F0FFFFFF	CALL Software.00421302	JMP to MSUBUH60.ThunRTMain
004213E2	0000	ADD BYTE PTR DS:[EAX],AL	
004213E4	0000	ADD BYTE PTR DS:[EAX],AL	
004213E6	0000	ADD BYTE PTR DS:[EAX],AL	
004213E8	0000	XOR BYTE PTR DS:[EAX],AL	
004213EA	0000	ADD BYTE PTR DS:[EAX],AL	
004213EC	48	DEC EAX	
004213ED	0000	ADD BYTE PTR DS:[EAX],AL	
004213EF	0000	ADD BYTE PTR DS:[EAX],AL	
004213F1	0000	ADD BYTE PTR DS:[EAX],AL	
004213F3	0053 98	ADD BYTE PTR DS:[EBX-68],DL	
004213F6	74 FC	JE SHORT Software.004213F4	
004213F8	AE	SCAS BYTE PTR ES:[EDI]	
004213F9	F8	CLC	
004213FA	D6	SALC	
004213FB	4F	DEC EDI	
004213FC	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
004213FD	C9	LEAVE	Unknown command
004213FE	FFBF		
00421400	71 A0	JNO SHORT Software.0042139C	
00421402	C6400 00000000	MOV BYTE PTR DS:[EAX+EAX],0	
0042140A	0100	ADD DWORD PTR DS:[EAX],EAX	
0042140C	0000	ADD BYTE PTR DS:[EAX],AL	
0042140E	74 6F	JE SHORT Software.0042147F	
00421410	6E	OUTS DX, BYTE PTR DS:[EDI]	
00421411	2E 149	DEC ECX	
00421413	53	PUSH EBX	
00421414	53	PUSH EBX	
00421415	6F	OUTS DX, DWORD PTR DS:[EDI]	
00421416	74 77	JE SHORT 00001490	
00421419	61	POPAD	
0042141C	50	PUSH EAX	
0042141D	61	POPAD	
0042141E	73 73	JNB SHORT Software.00421420	
00421420	70 6F	JD SHORT Software.00421422	
00421422	72 74	JB SHORT Software.00421424	
00421424	0001	ADD BYTE PTR DS:[EAX],AL	
00421426	F600 F8	TEST BYTE PTR DS:[EAX],0	
00421429	7E F9	JLE SHORT Software.00421424	
0042142B	0000	ADD BYTE PTR DS:[EAX],AL	

MSG ODbgScript
Script appears to have reached OEP, check log window for additional information, if any.
Aceptar Cancelar

8EE000	IsDebuggerPresent patched
8EE000	OutputDebugString patched
1401278	Import Redirection patched
4213D8	UA of OEP = 004213D8
4213D8	RUA of OEP = 000213D8

La informacion

Ahora el dump y la iat, no hay drama, este tiene acceso a ArmAccess.dll y CodeGen.dll

Pero el tema es que este no esta registrado,

1)agrego una seccion con topo

2)uso multimate assembler, y coloco la idea generica para armadillo mostrada en la comparacion anterior

```
MUltimate Assembler
4 5 6 7 8 9 10 11
PUSH @apuromafo15
PUSH @apuromafo16
CALL EBX
PUSH @apuromafo17
PUSH @apuromafo18
CALL EBX
PUSH @apuromafo19
PUSH @apuromafo20
CALL EBX

POPF
POPAD
JMP 052A3BB
Nop
nop

@apuromafo:
"key"
ADD BYTE PTR DS:[EAX],AL
@apuromafo2:
"USERKEY"
ADD BYTE PTR DS:[EAX],AL
@apuromafo3:
"Professional"
ADD BYTE PTR DS:[EAX],AL
@apuromafo4:
"VERSION"
```

00AB1000	60	PUSHAD	
00AB1001	9C	PUSHFD	
00AB1002	68 60D94300	PUSH topito55.0043D960	
00AB1007	B8 700F4200	MOV EAX,<JMP.&msubvm60.DllFunctionCall>	
00AB100C	FFD3	CALL EBX	
00AB100E	8BD8	MOV EBX,EAX	
00AB1010	68 8510AB00	PUSH topito55.00AB1085	ASCII "key"
00AB1015	68 8A10AB00	PUSH topito55.00AB108A	ASCII "USERKEY"
00AB101A	FFD3	CALL EBX	
00AB101C	68 9310AB00	PUSH topito55.00AB1093	ASCII "Professional"
00AB1021	68 A110AB00	PUSH topito55.00AB10A1	ASCII "VERSION"
00AB1026	FFD3	CALL EBX	
00AB1028	68 A910AB00	PUSH topito55.00AB10AA	ASCII "True"
00AB102D	68 B710AB00	PUSH topito55.00AB10B7	ASCII "PAIDFOR"
00AB1032	FFD3	CALL EBX	
00AB1034	68 C010AB00	PUSH topito55.00AB10C0	ASCII "2008.07.25"
00AB1039	68 CC10AB00	PUSH topito55.00AB10CC	ASCII "KEYCREATED"
00AB103E	FFD3	CALL EBX	
00AB1040	68 D810AB00	PUSH topito55.00AB10D8	ASCII "True"
00AB1045	68 DE10AB00	PUSH topito55.00AB10DE	ASCII "US"
00AB104A	FFD3	CALL EBX	
00AB104C	68 E210AB00	PUSH topito55.00AB10E2	
00AB1051	68 E510AB00	PUSH topito55.00AB10E5	ASCII "EXTRAINFO"
00AB1056	FFD3	CALL EBX	
00AB1058	68 F010AB00	PUSH topito55.00AB10F0	ASCII "Apuronafo CLS"
00AB105D	68 FF10AB00	PUSH topito55.00AB10FF	ASCII "ALTUSERNAME"
00AB1062	FFD3	CALL EBX	
00AB1064	68 0C11AB00	PUSH topito55.00AB110C	ASCII "yes"
00AB1069	68 1111AB00	PUSH topito55.00AB1111	ASCII "UPDATE"
00AB106E	FFD3	CALL EBX	
00AB1070	68 1911AB00	PUSH topito55.00AB1119	ASCII "8.0.0.8"
00AB1075	68 2211AB00	PUSH topito55.00AB1122	ASCII "VERSIONNUMBER"
00AB107A	FFD3	CALL EBX	
00AB107C	9D	POPF	
00AB107D	61	POPAD	
00AB107E	E9 550397FF	JMP topito55.00421308	

Y el resultado, es que ahora es operativo.y registrado

Y en el about

Version: 8.0.0.800

Registered to: Apuromafo CLS
key

Segunda parte:

El armadillo 7.4.0.740, no recuerdo haber leído que lo tenían crackeado ni unpacked, así que me propongo también a verlo

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.qhyzx	00001000	0015ACA2	00001000	0015ACA2	600000
.fvqhm	0015C000	00051690	0015C000	00051690	400000
.lctx	001AE000	0000A984	001AE000	0000A984	C00000
.vzom	001B9000	000B0000	001B9000	000B0000	E00000
.zqvif	00269000	00010000	00269000	00010000	E00000
.kijwd	00279000	00030000	00279000	00030000	C00000
.lrrayz	002A9000	00380000	002A9000	00380000	C00000

El proceso es el mismo el script

```
-----
MSG OdbgScript
-----
Successfully Patched OEP = 52006B of Child Process (PID= D9C) from E87B9B00 to EBFE.
Check log for more info. Press OK to continue.
Aceptar  Cancelar
-----
```

0052006A	C3	RETN
0052006B	E8 7B9B0000	CALL adumped_.00529BEB
00520070	E9 16FEFFFF	JMP adumped_.0051FE8B
00520075	6A 0C	PUSH 0C
00520077	68 C0BA5A00	PUSH adumped_.005ABAC0
0052007C	E8 4B4A0000	CALL adumped_.00524ACC
00520081	33DB	XOR EBX,EBX
00520083	895D E4	MOV DWORD PTR SS:[EBP-1C],EBX
00520086	33C0	XOR EAX,EAX

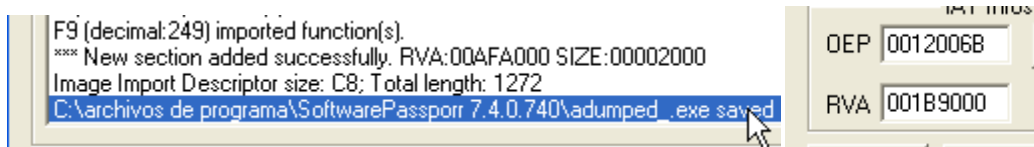
Luego en ArmlInline

Rebase IAT

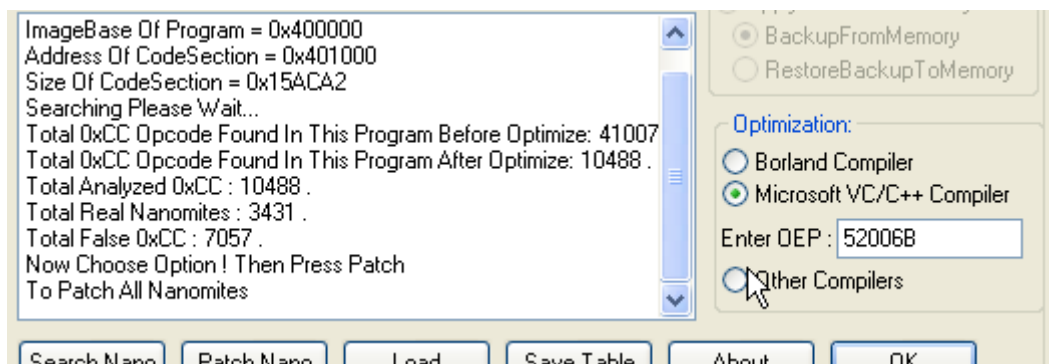
Luego reparar las importaciones externas

005B9000	7C80ADB0	kernel32.GetProcAddress
005B9004	7C801D77	kernel32.LoadLibraryA
005B9008	7C80ABEE	kernel32.FreeLibrary
005B900C	7C814B02	kernel32.GetEnvironmentVariableA
005B9010	7C81CF68	kernel32.GetEnvironmentStringsA
005B9014	7C812F18	kernel32.GetEnvironmentStringsW
005B9018	7C81CDEA	kernel32.ExitProcess
005B901C	7C801E16	kernel32.TerminateProcess
005B9020	7C80C068	kernel32.ExitThread
005B9024	7C810647	kernel32.CreateThread
005B9028	7C801A24	kernel32.CreateFileA
005B902C	7C810770	kernel32.CreateFileW
005B9030	7C80180E	kernel32.ReadFile
005B9034	7C810D97	kernel32.WriteFile
005B9038	7C810B9E	kernel32.SetFilePointer
005B903C	7C810A87	kernel32.GetFileSize
005B9040	7C809B57	kernel32.CloseHandle
005B9044	7C810E61	kernel32.GetFileType
005B9048	7C809478	kernel32.CreateFileMappingA
005B904C	7C80B915	kernel32.MapViewOfFile
005B9050	7C80B984	kernel32.UnmapViewOfFile
005B9054	7E3A59C9	USER32.EndDialog
005B9058	7C80FD30	kernel32.GlobalAlloc
005B905C	7C80FE92	kernel32.GlobalUnlock
005B9060	7C80CCA7	kernel32.SetHandleCount
005B9064	7C80B6B1	kernel32.GetModuleHandleA
005B9068	7E3D058A	USER32.MessageBoxA
005B906C	7E3B212B	USER32.GetWindowTextA
005B9070	77DAE834	ADVAPI32.RegCreateKeyExA
005B9074	77DB42F0	ADVAPI32.RegQueryValueA
005B9078	7C80ABD1	kernel32.GetProcessHeap
005B907C	7C812F2D	kernel32.GetCommandLineA
005B9080	7C809925	kernel32.GetACP
005B9084	7C80BE99	kernel32.FindResourceA
005B9088	7C82F7BC	kernel32.FormatMessageA
005B908C	7E3BC7A3	USER32.CreateDialogParamA
005B9090	7E3BB10C	USER32.DialogBoxParamA
005B9094	7C8130A3	kernel32.IsDebuggerPresent
005B9098	00000000	
005B909C	00000000	

Import Rec 1.7e +dump



Y los nanomites



Patch Nanomites To A Dumped File.....
 Reading Dumped File ...
 Checking Nanomites Data According To The File:
 NanoTable Checked Successfully.
 Patch Nanomites To File Complete Successfully !

Parcho los nanomites _____ Unpacked.

SoftwarePassport.exe de la version 7.4

Ejecuto el script

Armadillo 3.xx 4.xx 5.xx 6.xx CopyMemII+Debug Blocker IAT Recover.txt
Armadillo 3.xx 4.xx 5.xx 6.xx Unpack.txt
Armadillo Detach.txt

,luego import rec

softwarepassport_dump.exe

IAT Intros needed

DEP 00021268 IAT AutoSearch

RVA 00001000 Size 0000034C

Load Tree Save Tree Get Imports

msvbvm60.dll FT:hunk:00001000 NbFunc:D3 (decimal:211) valid:YES

Y luego las variables:

00A9E000	60	PUSHAD	
00A9E001	9C	PUSHFD	
00A9E002	BB C034837C	MOV EBX, kernel32.SetEnvironmentVariable	
00A9E007	68 7CE0A900	PUSH topo_2.00A9E07C	ASCII "Professional"
00A9E00C	68 8AE0A900	PUSH topo_2.00A9E08A	ASCII "VERSION"
00A9E011	FFD3	CALL EBX	
00A9E013	68 93E0A900	PUSH topo_2.00A9E093	
00A9E018	68 96E0A900	PUSH topo_2.00A9E096	ASCII "EXTRAINFO"
00A9E01D	FFD3	CALL EBX	
00A9E01F	68 A1E0A900	PUSH topo_2.00A9E0A1	
00A9E024	68 A6E0A900	PUSH topo_2.00A9E0A6	ASCII "UPDATE"
00A9E029	FFD3	CALL EBX	
00A9E02B	68 AEE0A900	PUSH topo_2.00A9E0AE	ASCII "2010.10.28"
00A9E030	68 BAE0A900	PUSH topo_2.00A9E0BA	ASCII "KEYCREATED"
00A9E035	FFD3	CALL EBX	
00A9E037	68 C6E0A900	PUSH topo_2.00A9E0C6	ASCII "True"
00A9E03C	68 CCE0A900	PUSH topo_2.00A9E0CC	ASCII "U7"
00A9E041	FFD3	CALL EBX	
00A9E043	68 D0E0A900	PUSH topo_2.00A9E0D0	ASCII "Apuromafo Cls"
00A9E048	68 DFE0A900	PUSH topo_2.00A9E0DF	ASCII "ALTUSERNAME"
00A9E04D	FFD3	CALL EBX	
00A9E04F	68 ECE0A900	PUSH topo_2.00A9E0EC	ASCII "Apuromafo CLS"
00A9E054	68 FBE0A900	PUSH topo_2.00A9E0FB	ASCII "USERNAME"
00A9E059	FFD3	CALL EBX	
00A9E05B	68 05E1A900	PUSH topo_2.00A9E105	ASCII "True"
00A9E060	68 0BE1A900	PUSH topo_2.00A9E10B	ASCII "PAIDFOR"
00A9E065	FFD3	CALL EBX	
00A9E067	68 14E1A900	PUSH topo_2.00A9E114	ASCII "key"
00A9E06C	68 19E1A900	PUSH topo_2.00A9E119	ASCII "USERKEY"
00A9E071	FFD3	CALL EBX	
00A9E073	9D	POPAD	
00A9E074	61	POPAD	
00A9E075	-E9 EE3198FF	JMP topo_2.00421268	

Copyright © 1996-2010 by Digital River, Inc. All

Version: 7.4.0.740

Registered to: Apuromafo Cls

key

Parte 3: version private version 4.3

Realizo un proceso de un click con armaggedon, pero siempre dicen que entre menor version mas claro es leer las versiones

La opcion trial es la default, por ende si no hay variables el progama corre como trial:

00400324	. 5F	PUP EDI	
00400325	. 85C0	TEST EAX,EAX	
00400327	.v74 50	JE SHORT unpacked.00400379	
00400329	. 8085 00FFFFFF	LEA EAX,DWORD PTR SS:[EBP-100]	
0040032F	. 68 50804500	PUSH unpacked.00458050	ASCII "Trial"
00400334	. 50	PUSH EAX	
00400335	. E8 C6FE0300	CALL unpacked.0044D200	
0040033A	. 59	POP ECX	
0040033B	. 85C0	TEST EAX,EAX	
0040033D	. 59	POP ECX	
0040033E	.v74 39	JE SHORT unpacked.00400379	
00400340	. 68 80124800	PUSH unpacked.00481280	
00400345	. E8 B69A0300	CALL unpacked.00446E00	
0040034A	. 85C0	TEST EAX,EAX	
0040034C	. 59	POP ECX	
0040034D	.v74 2A	JE SHORT unpacked.00400379	
0040034F	. 8085 00FFFFFF	LEA EAX,DWORD PTR SS:[EBP-100]	
00400355	. 68 BC734500	PUSH unpacked.004573BC	ASCII "DEFAULT"
0040035A	. 50	PUSH EAX	
0040035B	. E8 A0FE0300	CALL unpacked.0044D200	
00400360	. 59	POP ECX	

Pero aqui aparecen los otros tipos de version asi que ojo :

00400391	. C3	RETN	
00400392	> 68 847B4500	PUSH unpacked.00457B84	ASCII "Professional"
00400397	. 56	PUSH ESI	
00400398	. E8 63FE0300	CALL unpacked.0044D200	
0040039D	. F7D8	NEG EAX	
0040039F	. 59	POP ECX	
004003A0	. 1BC0	SBB EAX,EAX	
004003A2	. 59	POP ECX	
004003A3	. 40	INC EAX	
004003A4	. 5E	POP ESI	
004003A5	. C3	RETN	
004003A6	. 56	PUSH ESI	
004003A7	. BE 80154800	MOV ESI,unpacked.00481580	
004003AC	. 56	PUSH ESI	
004003AD	. E8 4E9A0300	CALL unpacked.00446E00	
004003B2	. 85C0	TEST EAX,EAX	
004003B4	. 59	POP ECX	
004003B5	.v75 04	JNZ SHORT unpacked.004003BB	
004003B7	. 32C0	XOR AL,AL	
004003B9	. 5E	POP ESI	
004003BA	. C3	RETN	
004003BB	> 68 58804500	PUSH unpacked.00458058	ASCII "eSellerate"
004003C0	. 56	PUSH ESI	
004003C1	. E8 3AFE0300	CALL unpacked.0044D200	
004003C6	. F7D8	NEG EAX	
004003C8	. 59	POP ECX	
004003C9	. 1BC0	SBB EAX,EAX	
004003CB	. 59	POP ECX	
004003CC	. 40	INC EAX	
004003CD	. 5E	POP ESI	
004003CE	. C3	RETN	
004003CF	. 56	PUSH ESI	
004003D0	. BE 80154800	MOV ESI,unpacked.00481580	
004003D5	. 56	PUSH ESI	
004003D6	. E8 259A0300	CALL unpacked.00446E00	
004003D8	. 85C0	TEST EAX,EAX	
004003DD	. 59	POP ECX	
004003DE	.v75 04	JNZ SHORT unpacked.004003E4	
004003E0	. 32C0	XOR AL,AL	
004003E2	. 5E	POP ESI	
004003E3	. C3	RETN	
004003E4	> 68 64804500	PUSH unpacked.00458064	ASCII "DigitalRiver"
004003E9	. 56	PUSH ESI	
004003EA	. E8 11FE0300	CALL unpacked.0044D200	
004003EF	. F7D8	NEG EAX	
004003F1	. 59	POP ECX	
004003F2	. 1BC0	SBB EAX,EAX	

Y vemos la variables como son llamadas en un 4.3a

0CF85	03	RET	
0CF86	55	PUSH EBP	
0CF87	8BEC	MOV EBP,ESP	
0CF89	81EC 00010000	SUB ESP,100	
0CF8F	53	PUSH EBX	
0CF90	56	PUSH ESI	
0CF91	8B35 10215000	MOV ESI,DWORD PTR DS:[<&kernel32.GetEnv	kernel32.GetEnvironmentVariableA
0CF97	BB 00010000	MOV EBX,100	
0CF9C	57	PUSH EDI	
0CF9D	53	PUSH EBX	
0CF9E	68 80124800	PUSH unpacked.00481280	[BufSize => 100 (256.) Buffer = unpacked.00481280 VarName = "USERNAME"
0CF93	68 30804500	PUSH unpacked.00458030	GetEnvironmentVariableA
0CF98	FFD6	CALL ESI	[BufSize => 100 (256.) Buffer = unpacked.00481380 VarName = "USERKEY"
0CF9A	53	PUSH EBX	GetEnvironmentVariableA
0CF9B	68 80134800	PUSH unpacked.00481380	[BufSize => 100 (256.) Buffer = unpacked.00481580 VarName = "VERSION"
0CF90	68 F4724500	PUSH unpacked.004572F4	GetEnvironmentVariableA
0CFB5	FFD6	CALL ESI	[BufSize => 100 (256.) Buffer = unpacked.00481580 VarName = "VERSION"
0CFB7	53	PUSH EBX	GetEnvironmentVariableA
0CFB8	68 80154800	PUSH unpacked.00481580	[BufSize => 100 (256.) Buffer = unpacked.00481480 VarName = "PAIDFOR"
0CFB0	68 3C7D4500	PUSH unpacked.00457D3C	GetEnvironmentVariableA
0CFB2	FFD6	CALL ESI	[BufSize => 100 (256.) Buffer = unpacked.00481480 VarName = "PAIDFOR"
0CFB4	53	PUSH EBX	GetEnvironmentVariableA
0CFB5	68 80144800	PUSH unpacked.00481480	[BufSize => 100 (256.) Buffer = unpacked.00481480 VarName = "PAIDFOR"
0CFCA	68 28804500	PUSH unpacked.00458028	GetEnvironmentVariableA
0CFCF	FFD6	CALL ESI	[BufSize => 100 (256.) Buffer = unpacked.00481480 VarName = "PAIDFOR"
0CFD1	A0 38044800	MOV AL,BYTE PTR DS:[480438]	
0CFD6	6A 3F	PUSH 3F	
0CFD8	8B85 00FFFFFF	MOV BYTE PTR SS:[EBP-100],AL	
0CFDE	59	POP ECX	
0CFDF	33C0	XOR EAX,EAX	
0CFE1	8DB0 01FFFFFF	LEA EDI,DWORD PTR SS:[EBP-FF]	
0CFE7	F3:AB	REP STOS DWORD PTR ES:[EDI]	
0CFE9	66:AB	STOS WORD PTR ES:[EDI]	
0CFEB	AA	STOS BYTE PTR ES:[EDI]	
0CFEC	8D85 00FFFFFF	LEA EAX,DWORD PTR SS:[EBP-100]	
0CFE2	53	PUSH EBX	
0CFE3	50	PUSH EAX	[BufSize => 100 (256.) Buffer = unpacked.00457308 VarName = "KEYCREATED"
0CFE4	68 08734500	PUSH unpacked.00457308	GetEnvironmentVariableA
0CFE9	FFD6	CALL ESI	
0CFEB	8D85 00FFFFFF	LEA EAX,DWORD PTR SS:[EBP-100]	
0D001	50	PUSH EAX	
0D002	E8 519F0300	CALL unpacked.00446F58	
0D007	8BF0	MOV ESI,EAX	

Vemos que tiene las estructuras

Creo que mas palabras no puedo decir, ya se han registrado 3 armadillos de diferentes versiones y creo que tarea dificil es el acostumbrarse a saber que hay Enviroment Variables y no solo parchar un salto o agregar de cero a 1, luego de establecidos los primeros entornos, comparan los proximos asi se compara el V7 o el V8 en la aplicacion.

Conclusiones,

armadillo puede desempacado, pero veo que con el tiempo armadillo crea cada vez mas capas o mas desencrptaciones basadas en entorno padre/hijo, asi que no será nada fácil en el 2014.

Proyeccion2:

El usuario y serial solo funciona en un Unpacked o inlined

Agradecimientos

A toda la lista de CLS

Saludos Apuromafo