|  |
| --- |
|  |

|  |
| --- |
| **Armadillo 9.x** |
| Usando las herramientas en los momentos indicados |
|  |
| Este pequeño tutorial expondrá herramientas de otros usuarios para enfrentarnos a un armadillo 9.x |
|  |
| **Apuromafo** |
| **08/07/2015** |
|  |

Contenido

[*Introducción:* 2](#_Toc424924155)

[*Análisis (que opciones tiene el packed de Armadillo)* 4](#_Toc424924156)

[Estrategias para analizar este Armadillo 9: 6](#_Toc424924157)

[Proceso 1:Unpacking 7](#_Toc424924158)

[En Windows 8.1 script v1 7](#_Toc424924159)

[En Windows 8.1 sin script 7](#_Toc424924160)

[Windows xp sp3 Script: 13](#_Toc424924161)

[Windows xp sp3 Sin script 15](#_Toc424924162)

[Proceso 2Revisando el programa desempacado+bonus packed: 23](#_Toc424924163)

[Proceso 3: Inline 26](#_Toc424924164)

[Inlined sobre el unpacked 26](#_Toc424924165)

[Inlined sobre el packed 29](#_Toc424924166)

[*Proceso 3: Keygening* 39](#_Toc424924167)

# *Introducción:*

Hola hace no mucho un buen amigo, dado su gran experiencia de reverser pidió ayuda a desempacar, pero dado que normalmente no me detengo cuando de armadillo se trata, aquí un pequeño tutorial.

Hay una manera de contribuir a la protección de la humanidad, y es no resignarse. Ernesto Sábato (1911); escritor argentino

|  |  |
| --- | --- |
| Programa | Flowcode.exe |
| Compañía | Matrix TSL , Copyright (C) 2000-2015 |
| Descarga | https://www.matrixtsl.com/flowcode/download/ |
| Tamaño | Setup: 207.3 MB / Programa: 14.6 MB |
| Versión del Programa | 6. 1. 2. 0 |
| Herramientas | OllyDbg-x64dbg y las que se necesiten para armadillo. |
| Packer/protector | Armadillo 9.x |
| Utilidad | El software FlowCode es una interfaz gráfica de programación que permite desarrollar aplicaciones para casi cualquier sistema microcontrolador basado en PIC micro, utilizando solamente diagramas de flujo y sin necesidad de programación en lenguajes de alto nivel, contando con las herramientas necesarias para implementar el diagrama, y definir componentes gráficos como LEDS, microswitches, LCD, teclados, puertos RS232, etc.  Adicionalmente, cuenta con un simulador integrado, de tal manera que es posible simular en tiempo real el programa, observando sus entradas y salidas en el monitor de la PC. |
| Entorno | Windows 8.1 / Virtual Machine Windows XP sp3 |
| Compilador | Visual C/C++ |
| Objetivos | Vencer la protección. |
| Tutorial por | Apuromafo CLS |
| Fecha | 15 -07-2015 |

Hoy, vamos a comentar hoy sobre un packed con “Armadillo”, un packer y protector bastante bueno y bastante configurable , Su web es <http://siliconrealms.com/> Actualmente refiere que ya no existe más soporte desde el primero de Julio por otro lado la última versión conocida de armadillo se baja de aqui:  
<http://web.archive.org/web/20140625052607/http://siliconrealms.com/armadillo.php>

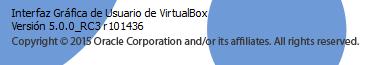
“flow code 6” es un programa que necesariamente necesita estar registrado para correr, en mi caso intentaré dejar lo más operativo que pueda para mi nivel de licencia (trial de 30 días) para que pueda funcionar desempacado, desconozco hasta donde llegaremos y que conclusiones tendremos. Mas información en <http://www.matrixtsl.com/resources/files/datasheets/FC6-datasheet.pdf>

Revisando un poco del packer después de los tutoriales anteriores las novedades:

1)De las herramientas que estaban siendo soportada, han desaparecido en general dado que han colocado bastante capas, con encriptación o ofuscación que no hace fácil la tarea de automatizar y buscar “patrones”,entonces para mejorar soporte luego de un tiempo liberaron versiones si se realiza un inline es necesario desencriptar una sección antes de intentarlo o hacer más de una revisión en otro sistema operativo , asi que terminemos de resumir en etapas para no desordenar,algunos inclusive mezclan 2 packers, en general **Para saber que versión tiene un armadillo Nuevo (9.x) sirve la tool de vel, la tool de mr.exodia llamado atk., lo demás serán script individuales según la ocasión.**

2) aparecieron muy pocas herramientas nuevas, ArmaGeddon existen versiones 1.9, 2.0 , 2.2 para Windows 7 y para xp y en la 2.2 compartió su source , mr exodia compartió también el de armadillo keygener tool con su source , entonces literalmente he de pensar que gran parte de análisis está indirectamente expuesto osea no dejan oculto las herramientas.  
Respecto a ideas y herramientas en particular, aveces saludo a Mr.exodia el cual me actualizó algunas herramientas pedidas de forma personal para codesplicit , nanomites, armadillokeygen tool dado que como bien saben , cuando se comparte información privada, muchas veces se mejoran los rendimientos para hacer pruebas, en este caso mi pc es 3 veces más lento que el pc de mr.exodia , pero 50 veces más rápido que el pc que tuve cuando hacia comienzos en keygening en armadillo, por otro lado Mr.exodia dado que comenzó con un depurador nuevo desde 0 llamado x64dbg de 32bytes y 64 bytes ya no se ha dedicado a armadillo desde entonces, a esta altura sigue desarrollando más capacidad en su depurador que con la ayuda de varios reverser llega a ser bastante útil y versátil, además brinda apoyo en exetools como moderador de foro, por mi lado creo que esto sería lo más general a dar de introducción, si alguno quiere desempacar armadillo en Windows 8.1 le sugeriría usar la versión x64dbg creada por mr.exodia por el minuto, bueno vamos a lo que hablábamos.

3) para poder ejecutar un armadillo sin problemas, es idóneo tener alguna máquina virtual, me descarguré un Windows sp3 , me descargué virtual box y comencé con las herramientas a ordenarlas otra vez, dado que aquí si pueden funcionar sin problema en general.

**Descargo Virtual box** 

Luego instalo un sistema operativo con por lo menos 2gb de ram asociada ( mi pc actualmente tiene 6gb así que no le dará lio ejecutar una máquina con 2gb ), xp funciona bien hasta con 512mb de ram

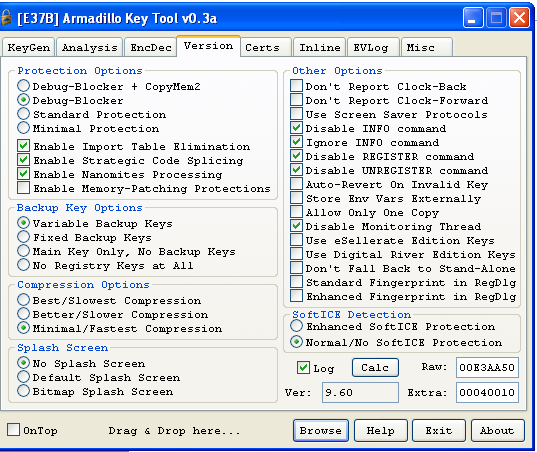
Cuando ya tengo eso, configuro una carpeta o ruta compartida para instalar lo básico que es:

1. Descompresor de archivos :7zip, winrar etc
2. Navegador :Chrome/firefox
3. Herramientas Necesarias: ArmaGeddon 1.9/2.2 /o la que encontremos xD, ollydbg para continuar o bien el mismo x64dbg de 32bytes de mr.exodia

# *Análisis (que opciones tiene el packed de Armadillo)*

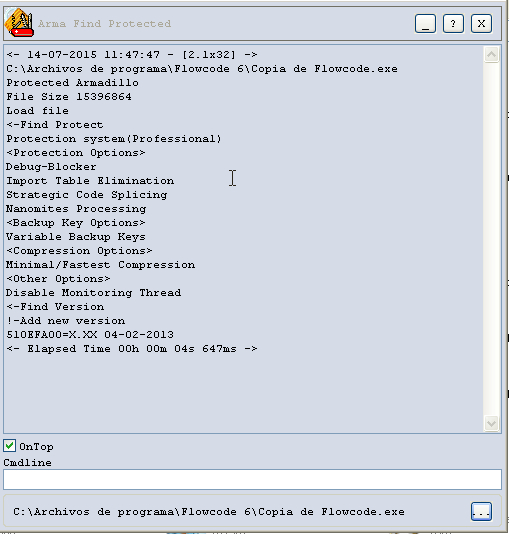
Las 2 herramientas que son de interés son de Mr.exodia (atk) y vel (arma fp 2.1)

1. [**https://github.com/mrexodia/akt**](https://github.com/mrexodia/akt) **En la opción “reléase” existen versiones ya compiladas del código en el cual podemos descargar si no tenemos CODEBlocks para compilar, si alguno le interesa el codeblock se baja el** codeblocks-13.12mingw-setup.exe de <http://www.codeblocks.org/downloads/26> para inline es necesario tener codeblock para que en el caso particular de las dll puedan hacer uso.



1. **Arma Find Protected 2.1**

|  |
| --- |
| [**http://forum.exetools.com/showthread.php?t=14131**](http://forum.exetools.com/showthread.php?t=14131)  **->**<http://rghost.ru/36912724> |
| [**http://www.manhunter.ru/underground/94\_programmi\_dlya\_analiza\_ispolnyaemih\_faylov.html**](http://www.manhunter.ru/underground/94_programmi_dlya_analiza_ispolnyaemih_faylov.html)  **->http://www.manhunter.ru/download/9329/Armadillo.Find.Protected.2.1.zip** |



# Estrategias para analizar este Armadillo 9:

Entonces aquí tenemos que diferenciar 4 procesos para no desorientarnos, todos llevan a la misma experiencia, el programa operativo.

**Proceso 1: Unpacking.**

Aquí encontramos el oep, reparamos la iat, reparamos código fuera del ejecutable (iat y code splicies), reparamos nanomites.

**Proceso 2: Revisando el Unpacked**

Agregamos parches específicos para que no extrañe el packer, esto es normalmente viendo que parte verificará si hay o no armadillo (que variable busca).

**Proceso 3: Inline .**

Aquí confirmamos si hay secciones encriptadas/desencriptadas, el inline en general se basa en mantener el crc del programa a modo de poder ejecutar sin lio ,si la sección encriptada de armadillo necesitamos que esté desencriptada existe una herramienta de apoyo para encontrar los crc y luego podemos agregar el inline, normalmente existen 2 instancias, en packed o en unpacked, es idóneo realizarlo en el packed , pues se puede habilitar opciones como “register” y evita errores de nanomites posteriores el tema es la compatibilidad, el verdadero problema es que además de parchar armadillo es necesario cambiar 1 salto para que no nos pida denuevo registrar.

**Proceso 4: Keygening .**

Cuando usamos Armadillo Keygen tool de mr.exodia , permite reconocer datos importantes en los packeds de armadillo para versiones 3.7x a la 9.x , respecto a los certificados existen diferentes niveles, siendo según el nivel la complejidad, para versión unsigned es necesario tener usuario y clave, para la última versión es necesario parchar ECDSA , en esta aplicación en particular es un Short v3 lv 5 como bien es sabido que no es ingresado el usuario/serial de forma directa, este programa no puede ser registrado sin antes haberlo inlineado, por lo menos al minuto no es posible hacer inferencias si el ingreso de las claves descodifica nuevas secciones del programa (conocido como secured section),pero en general se ve que si influye, por eso es indispensable keygenearlo.

# Proceso 1:Unpacking

Se bien que la idea es aprender, pero dado la gran documentación armadillo se ha intentado en general desempacar de forma automática comenzaré para ver que tal funciona el desempacado:

## En Windows 8.1 script v1

Para desempacar automáticamente en Windows 8.1 es necesario de un depurador llamado x64dbg yo uso el x32bytes <http://x64dbg.com/#start>

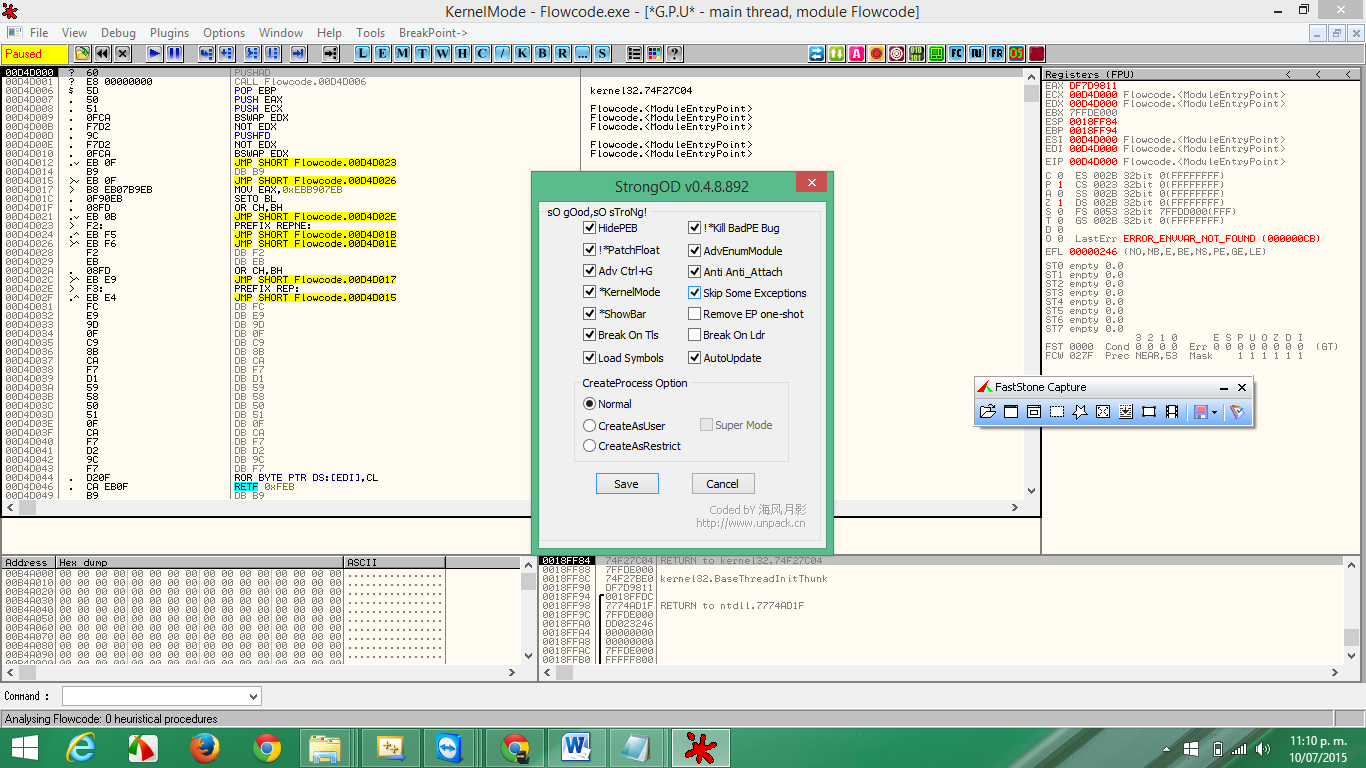
Script:

<https://github.com/x64dbg/Scripts/blob/master/Armadillo%20v9.64%20Debug-Blocker.txt>

hasta aquí está 1) oep 2) iat lista para reparar con uif 1.2 3) listo para codesplicit 4) listo para reparar las nanomites.

## En Windows 8.1 sin script

Primero en Windows 8.1 con ollydbg para que no piensen que estoy usando el script:   
en el entrypoint “Skip some exceptions”, isdebuggerpresent ya lo parcha por defecto en “HIDEPEB”



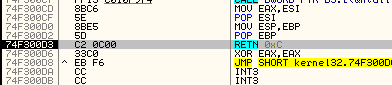
Bp en OpenmutexA (necesitamos 2 veces y luego en eax =1)

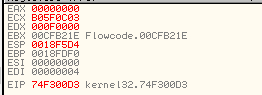
Bueno comienzo bp en OpenMutexA : la primera vez (veo esto)



En la segunda

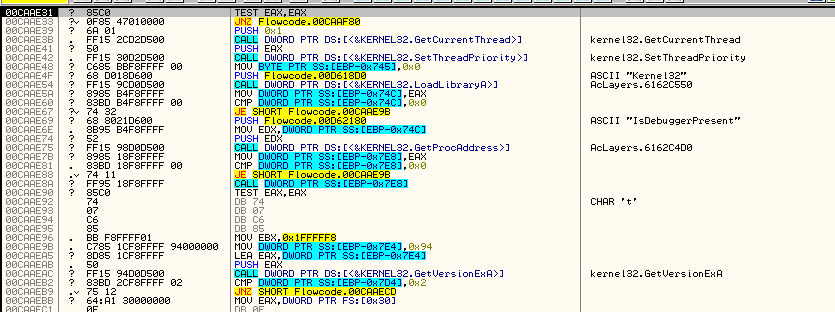


Quito el BP y voy al terminar en el return (ctrl+f9 y llego al retn)

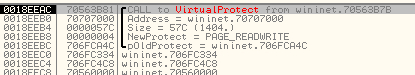
Eax dice 

Lo cambio eax=1

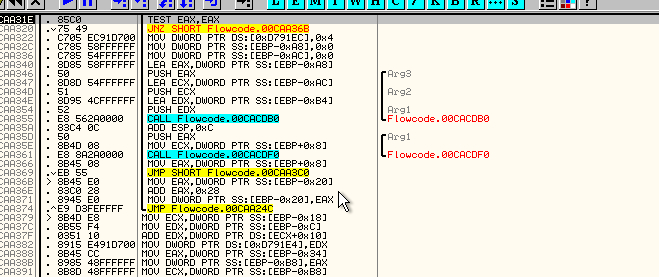
Con f7 estoy aquí



Ahora vamos por la iat, primero vamos BP en VirtualProtect (si uso Windows 8.1) veo esto



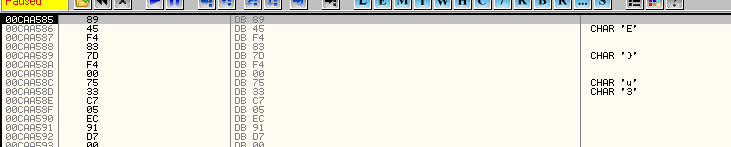
Al ejecutar quito bp y si retorno veo esto



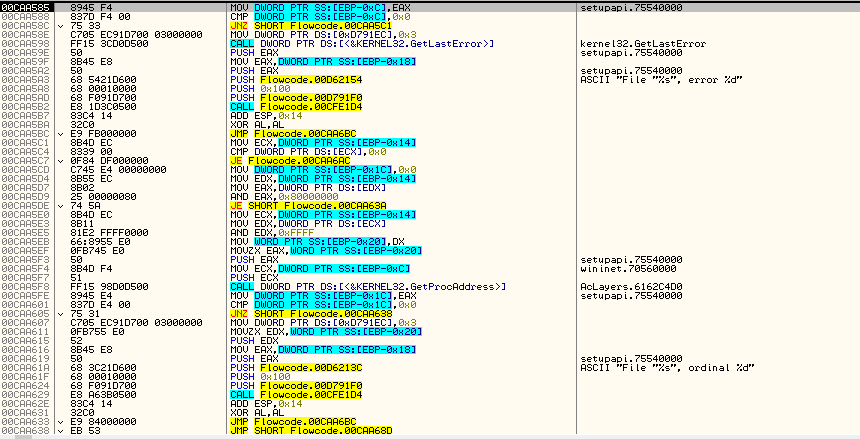
BP en LoadLibraryA y run , quito el bp, ctrl+f9 y llego al retn



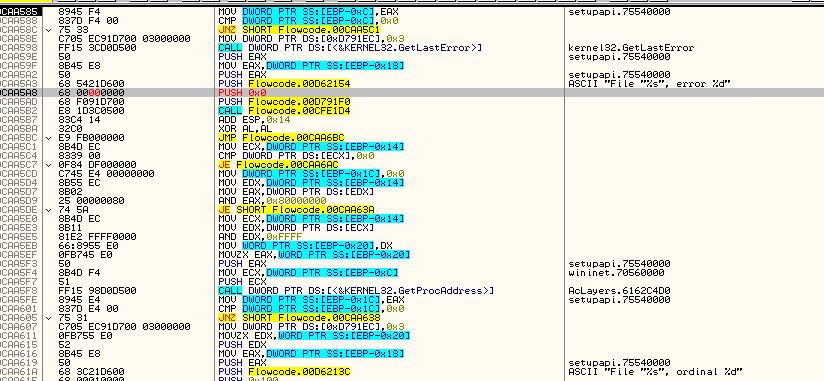
Al pasar estoy en una sección de armadillo



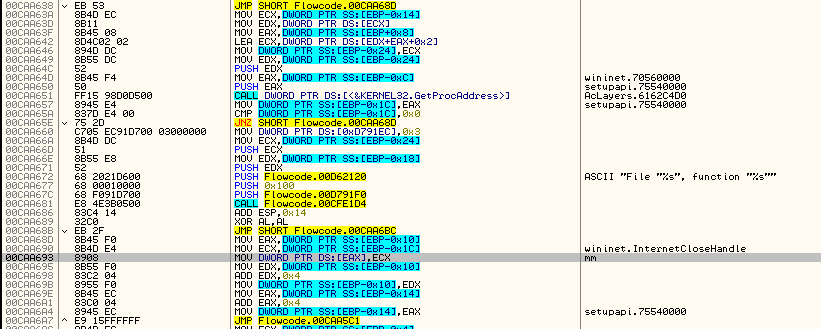
Al quitar el análisis en la sección sin ejecutar, comenzaré a parchar para reparar la iat.

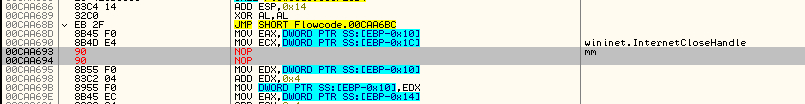


Estamos en la sección idónea para reparar la iat cambio Push 100 a push 0

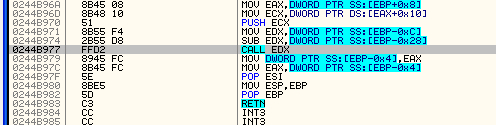


Al bajar busco algo que dice file %s function y donde coloco “mm” osea el código 8908 debe nopearse

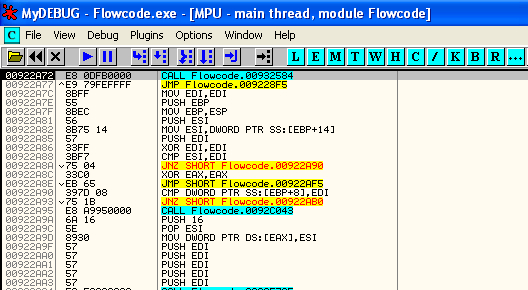




Luego de reparar la iat al esperar en los saltos a que termine o bien como segunda opción se coloca un ret en el call 0CFE104 , luego coloco bp en CreateRemoteThreadEx al salir espero la llamada al oep

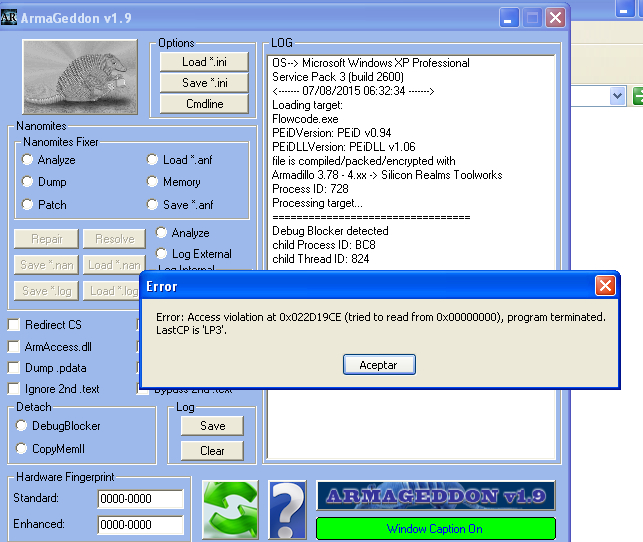
****

**Y llegamos al oep**

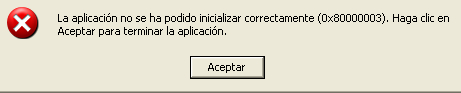
****

La iat se repara con import fixer 1.2 y luego tal cual como se podía avanzar tal cual como estaba en el otro punto.

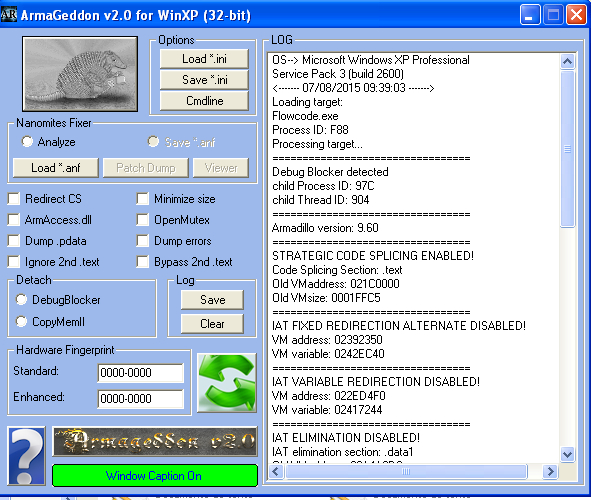
Windows xp sp3 Herramientas de Unpacking Automático:



La 2.2



La 2.0



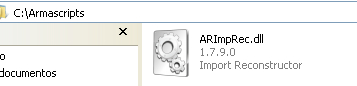
## Windows xp sp3 Script:

Existe un script fácil de configurar, soporta de la version 3.x a la 9.x. para esto requieren cuenta en tuts4you.com para lograr descargar:

<https://forum.tuts4you.com/topic/36970-armadillo-factotum-universal-10/>

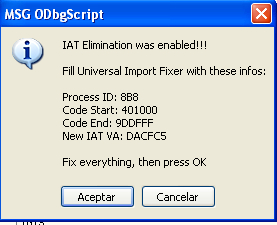
Hace uso de la dll de Nacho\_dj que usa en ArmaGeddon, por otro lado es posible automatizar el proceso completo solo le falta opción para nanomites o bien casos particulares de versión 3.x

1. Da indicaciones configurar la ruta de un dll de nacho\_dj

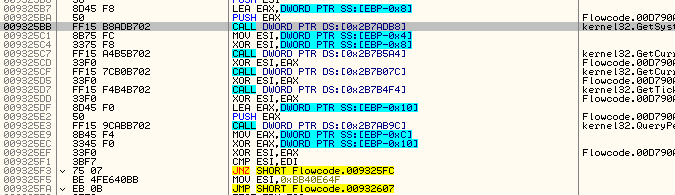


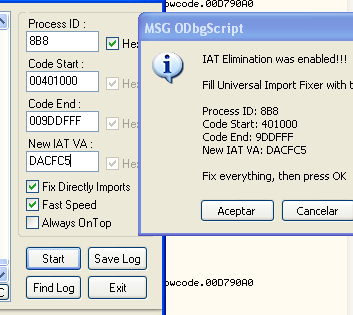
1. Luego ejecutar el script y muestra este mensaje:

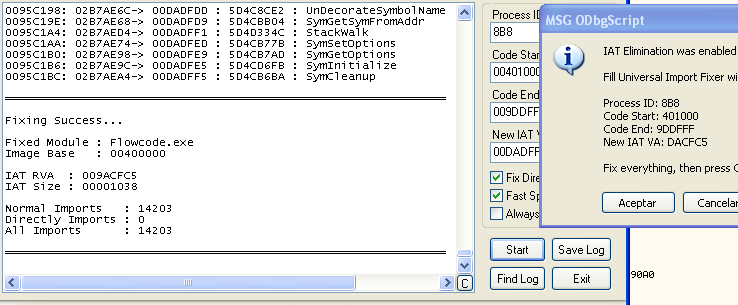
Ya desempacando sin problema



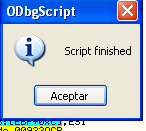
Actualmente la iat antes de la tool “Import fixer 1.2 by magic” vemos como la iat está afuera del programa



 en UIF 1.2 Pulso start



Luego de la tool está asi en una sección propia del programa

Al terminar en el oep ya se puede cerrar ollydbg y buscar en la ruta el unpacked: Flowcode\_dump\_.exe

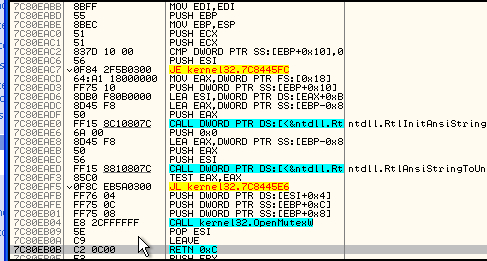
## Windows xp sp3 Sin script

1.a luego que bypaseamos el debugblocker (openmutexA por 2 y cambiamos retorno por 1), podemos buscar el salto al oep, que suele ser creado con Createthread o bp en las secciones según corresponda.

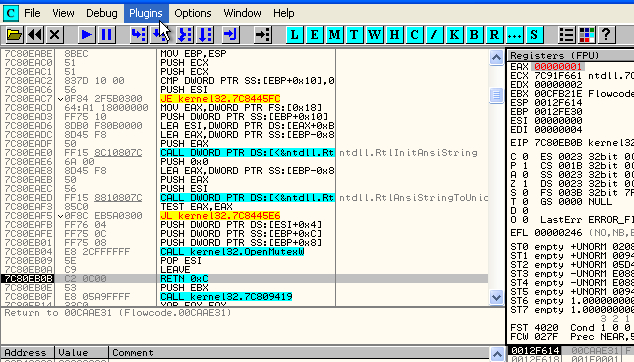
**Estoy en el entrypoint**



**Voy a la expresión OpenMutexA, en el retorno**



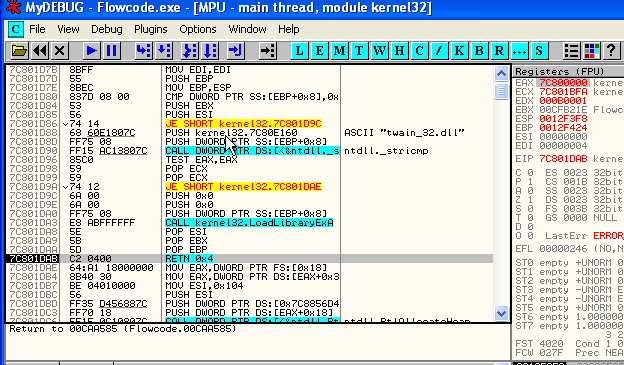
**Coloco bp y luego 2 run, cambio eax de 0 a 1**

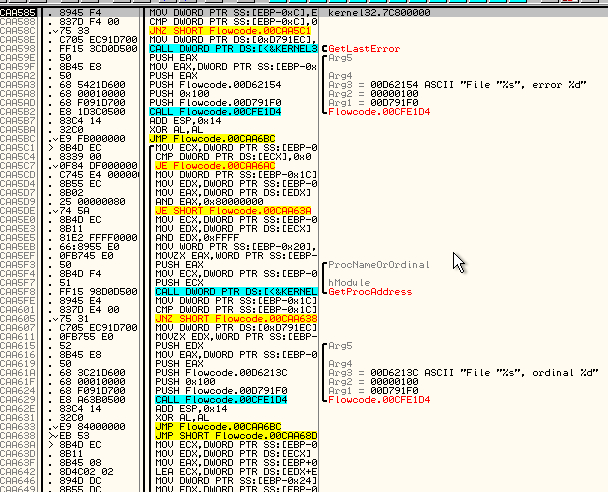


**Ahora a reparar la iat**

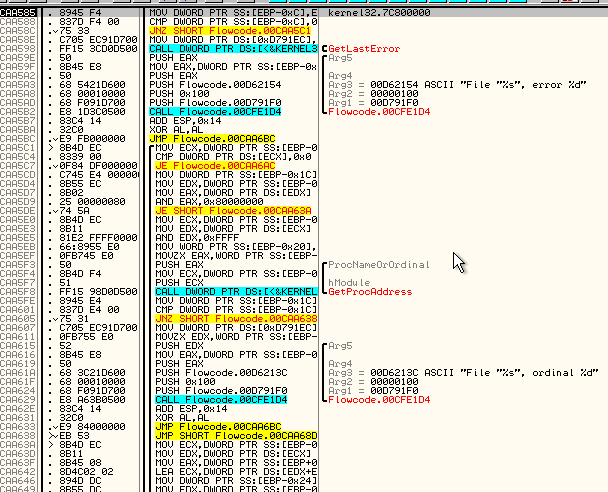
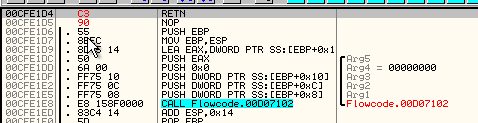
1.b desde los pasos anteriores buscar donde reparar la iat ( import adress table) una vez identifiquemos saltomágico/lugar de las apis tenemos las apis directas yo me guio con LoadlibraryA,o como anular según corresponda en el OEP. se usa Import Fixer 1.2 porque permite colocar la IAT dentro del ejecutable (import elimination lo deja fuera) y luego el importador de iat de gusto , arminline también permite el mismo proceso pero pide información técnica en la mayoría de los casos.

**Coloco LoadlibraryA y en el retorno coloco bp**

**me retornará a la sección de armadillo**

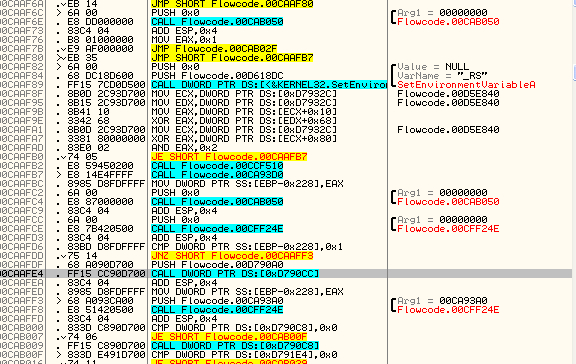


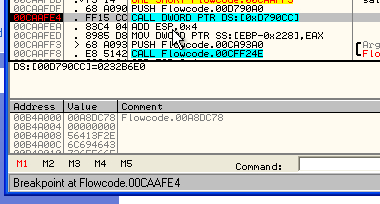
**Veo el push 100 y un call, dado que estoy contra el tiempo, haremos algo más rápido , cambiar ese call a ret**

**** 

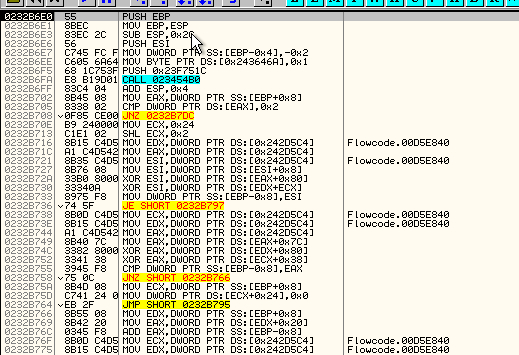
**Con esto no debería tener problema de IAT**

openmutexA es para debugblocker y rs es para copymew2 , el lugar donde muestro es donde puedo encontrar el oep luego que se reparo la iat el programa retorna de una funcion.





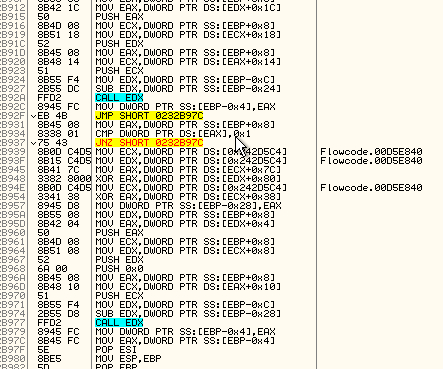
**F7 entro**



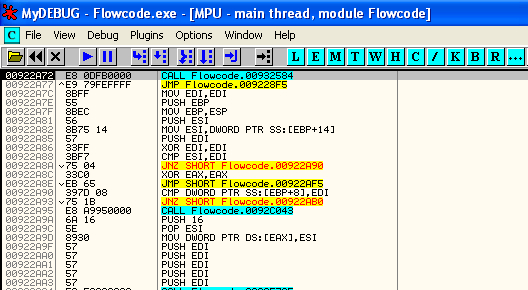
**Este lugar normalmente no se analiza porque no tiene módulo asociado**

**Y si se analizara te crasheará el depurador xD**

**Bueno siguiendo si haces un pequeño scroll, encontrarás el salto al oep**

****

**Que al entrar con f7 estás en el oep**

****

1.c Reparar code splicies (código fuera del ejecutable) normalmente se identifica por tener saltos “–“ y no “^” dejándolo que actue, desde oep arminline es la mejor herramienta, también existe una herramienta de mr.exodia para cambiar al ejecutable sin reparar los codesplicies también. [https://bitbucket.org/mrexodia/splicerebase](https://www.google.com/url?q=https%3A%2F%2Fbitbucket.org%2Fmrexodia%2Fsplicerebase&sa=D&sntz=1&usg=AFQjCNG32fmvu2TlNW9YXb3qhnM8Gv7I-A)

Actualmente tenemos un unpacked hasta este nivel sea como sea la forma.

1.d Reparar nanomites (int3 que son saltos manejados) en mi caso particular usaré nanolol que es la tool privada de mr.exodia, pero también es posible usar armtool de nevada o arminline, sin eso, deberían loguearse las llamadas de las excepciones para cada caso según se necesite , es exactamente lo mismo.

**------ NanoFixer Analyzing Nanomites ------**

**Initializing...**

**38457 INT3 Found**

**38457 INT3 Analyzed, 38210 False, 247 Real**

**Analyzing successfully completed**

**Saving Nanomites table...**

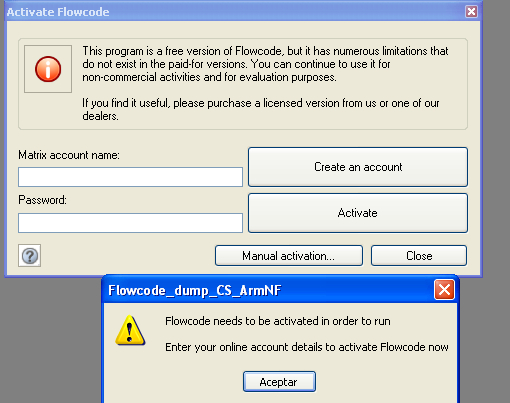
**247 nanomites saved...**

**Done.**

1.e Agregar parches y/o environmentvariables según se necesite, para esto se puede usar armadillo reducer del team FFF, o bien algun código snipet , comentaré un poco en la sección de inline , ahora a revisar el unpacked

# Proceso 2Revisando el programa desempacado+bonus packed:

Al abrir el unpacked comienza una nueva misión, que corra sin la nag que exija registrar:



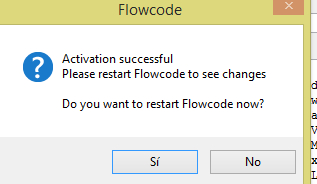
El proceso llama a un archivo también packed con armadillo, con los mismos certificados, con una forma especial de verificar la clave estilo rsa , pero tiene más algoritmos estos valores no son los mismos que nos dan usuario1, key1, key2

Y al escanear tiene estos algoritmos a primera vista:

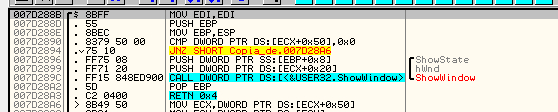
Al ver las string luego de desempacado: claramente se encarga de habilitar un comando -register

Y la variable de interés que serán verificadas online terminan en varias opciones:



El tema es que luego que se registra con los parámetros establece por setenvironmentvariableA/W una palabra “REGISTERED” los valores que necesita el programa como si hubiéramos colocado “flowcode.exe register” 

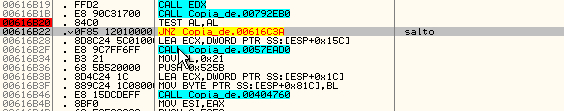
Ahora bien, depurando el programa desempacado, antes de la NAG coloco bp en algunos call hasta pillar en los calls , cual es la última que llama y encuentro la llamada a la responsable de la nag



Al retorno identifico:

Si cambio el push 5(show) a un push 0 (hide) , con esto no tengo nag al comienzo, a mi no me incomoda, asi que lo dejaré por el minuto.

Luego confirmo que valida mi nombre y usuario pero como sabemos no nos validará entonces me propongo parcharlo original y con esto nunca más llamara a la aplicación de licencia en la carpeta tool:



Parchado



Ahora no veremos mas el pedido de activación, porque con esto no ejecutará la herramienta de tool, no validará en internet y solo verificará lo que esté en memoria , tenemos el programa desempacado y crackeado en la verificación de armadillo, no en la del programador.

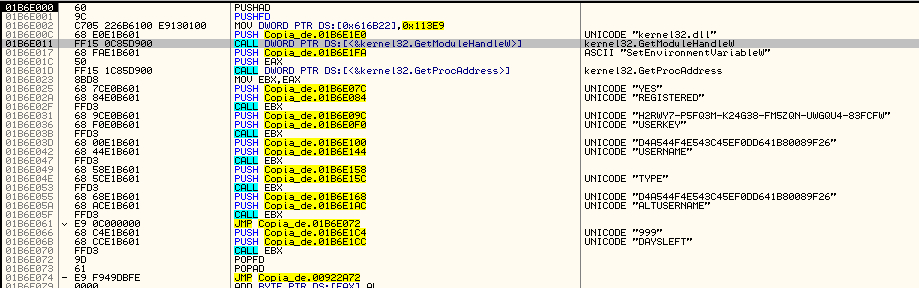
# Proceso 3: Inline

Digamos que es el proceso más largo dado que esto permitirá normalmente para un solo sistema operativo , puede que resulte para más, pero quiero pensar que nadie estará distribuyendo los Inlined, les haré un ejemplo armadillo 9 inline que se distribuye no corre en Windows 8.1, Si investigamos desde la versión de armadillo 3 a la 9, en un 99% de los inlined de armadillo corren en Windows xp sp3, pero no así en Windows 8.1, solo el 40% corre, esto hace la diferencia entre si el programa empacado correrá o no,como lo hacía en el packed en el caso particular de armadillo 9, corre sin problema en Windows 8.1, asi que como les digo, inlinear es solo para su sistema operativo siempre y cuando sepan que harán un hook sobre outputdebugstring y luego virtualprotect entre otros osea no es algo muy comentado.

## Inlined sobre el unpacked

Comenzamos en una sección vacía o bien agregamos con topo una sección, tenemos el programa packed agregamos una sección para no alterar por el minuto ninguna sección, damos permiso sobre el lugar a donde parchar el salto 616b22 de jnz/je a jmp (encontrado en el proceso 2), los EnvironmentVariable son para mostrar variables, se establecen con setEnvironmentVariableA/w y se llaman con GetEnvironmentVariableA/w cuando esta empacado desencripta la licencia y establece esas variables, cuando se desempaca, no existen estas variables, por lo cual es idóneo hacer un log de las variables si uno posee una versión registrada.

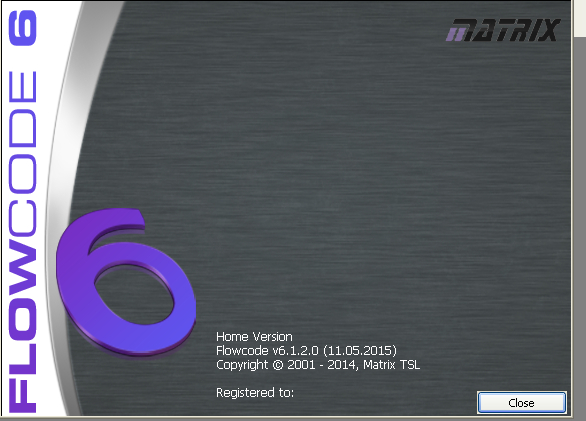
En el caso particular inyectaré variables SetEnvironmentVariableW , es una api no presente en el unpacked, solo en el packed, entonces haré llamadas a GetModuleHandleW y GetProcAdress, que son 2 apis presentes en el unpacked, luego de encontrado los valores se establecen en Unicode cada variable

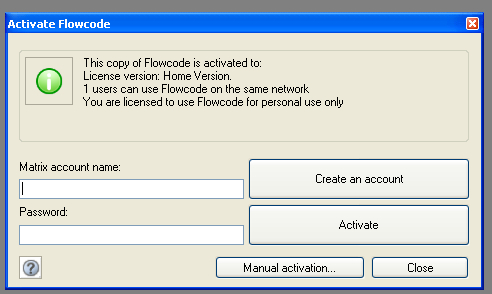


TYPE =”1” no logra apreciarse adjunto el código .

|  |
| --- |
| <01B6E000>  PUSHAD  PUSHFD  MOV Dword ptr ds:[00616B22],000113E9  PUSH @kernellUnicode  CALL DWORD PTR DS:[0D9850C] ; kernel32.GetModuleHandleW debe buscarse  PUSH @SetEnvironmentVariable ; ASCII "SetEnvironmentVariableW"  PUSH EAX  CALL DWORD PTR DS:[0d9851c] ; kernel32.GetProcAddress debe buscarse  MOV EBX,EAX  PUSH @var001  PUSH @var002  call EBX  PUSH @var003  PUSH @var004  call EBX  PUSH @var005  PUSH @var006  call EBX  PUSH @var007  PUSH @var008  call EBX  PUSH @var009  PUSH @var010  call EBX  jmp @continua  PUSH @var011  PUSH @var012  call EBX  @continua:  POPFD  POPAD  JMP 00922A72  !align 4  @var001:L"YES\0"  !align 4  @var002:L"REGISTERED\0"  !align 4  @var003:L"H2RWY7-P5FQ3M-K24G38-FM5ZQN-UWGQU4-83FCFW\0"  !align 4  @var004:L"USERKEY\0"  !align 4  @var005:L"D4A544F4E543C45EF0DD641B80089F26\0"  !align 4  @var006:L"USERNAME\0"  !align 4  @var007:L"1\0"  !align 4  @var008:L"TYPE\0"  !align 4  @var009:L"D4A544F4E543C45EF0DD641B80089F26\0"  !align 4  @var010:L"ALTUSERNAME\0"  !align 4  @var011:L"999\0"  !align 4  @var012:L"DAYSLEFT\0"  !align 4  @kernellUnicode :L"kernel32.dll\0"  @SetEnvironmentVariable :"SetEnvironmentVariableW"  !align 4 |

<probando inlined con el salto modificado, variables REGISTERED=yes y el TYPE=”1”>





Alguno ahora se pregunta, emm ¿como supiste esas variables? Esto es porque además también lo he registrado a otro nombre xD bueno ya comentaré en keygening.

## Inlined sobre el packed

El packed o empacado, contiene todos los certificados y más, pero dado que este programa usa usuario y serial de forma indirecta , es necesario habilitar la opción ENABLEREGISTER y luego tener que generar una clave para confirmar que EnvironmentVariableA/W habilita o que sección desencripta, como es el caso de tener que generar claves, no es necesario inyectar más variables, pues cuando ingreso la clave válida para su licencia automáticamente serán establecidas al ingresar el tipo de licencia.

De los Empacados hay 2 instancias: una es que tenga encriptada la sección y otra que no, el más común es parchar hiw, parchar el comando register, parchar comando info, quitar expiración de días etc .

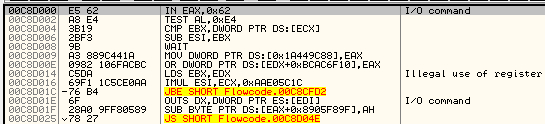
Comenzemos a ver si la sección está o no encriptada

Veamos con alt mas E veo esto:

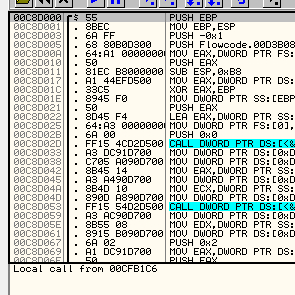


Cuando doy Enter sobre el “flowcode” con ruta de archivos de programa (enter in cpu)

Antes del run (F9):



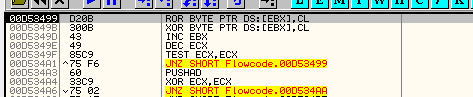
Después del run:



Reinicio el depurador

Coloco Bp en write y bp en ejecución en c8d000 (dirección de alt mas e, enter) para realmente comenzar que novedades encontramos:

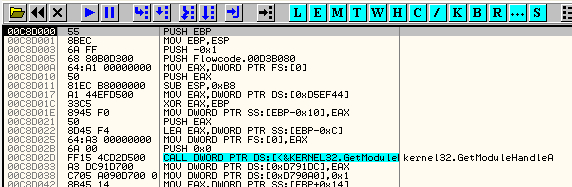
Primer run



Segundo run



Tercer run (alcanzamos el objetivo del entrypoint tipo) de esta dll llamada Security.dll

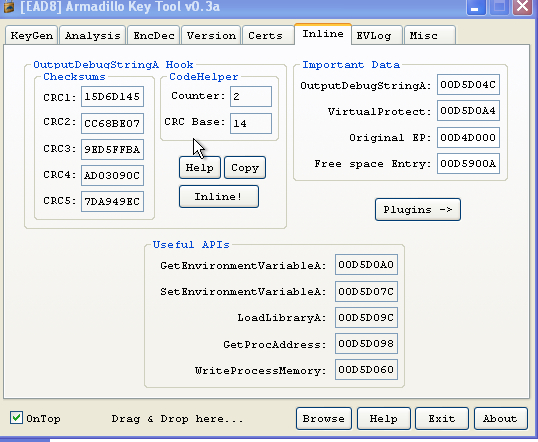


Bueno está claro que está encriptado

Dado que quiero algo operativo para Windows 8.1 haré uso de herramientas que recuerdo haber usado en el pasado cuando usaba el inline , esto es una implementación de un cracker llamado DisArm en Seek And Destroy, para desencriptar una sección.

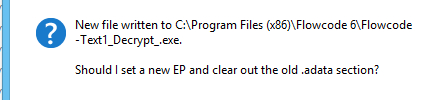
1) este archivo proviene de un tutorial , pero que le vi utilidad porque desencriptaba sin problema.

2)Armadillo Key Tool v 0.3ª en el Tab “Inline” :

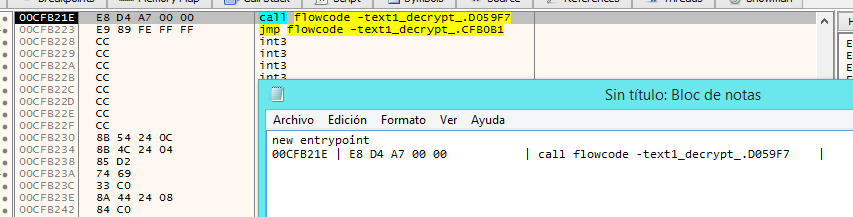


Al pulsar inline podremos hacer un código inlineado automático, te da la opción de decir donde dejas el EP donde tienes el código libre   
Tomo el programa original flowcode.exe lo copio denuevo para llamarlo Flowcode\_text1\_decrypt.exe

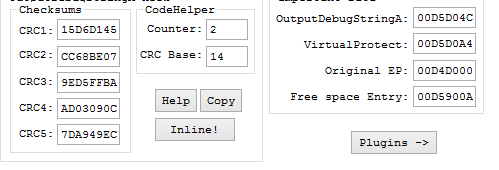
1. Haciendo uso de la herramienta 1 que desencriptaba la sección avisa : debo establecer un nuevo EP y borrar la sección anterior de .adata si/no



Bajo cff explorer http://www.ntcore.com/exsuite.php

El entrypoint de este programa el cual desencripto es 00CFB21E 

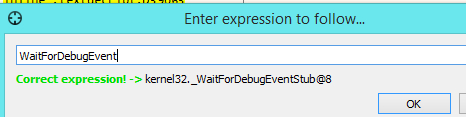
Coloco la tool de inline de mr.exodia



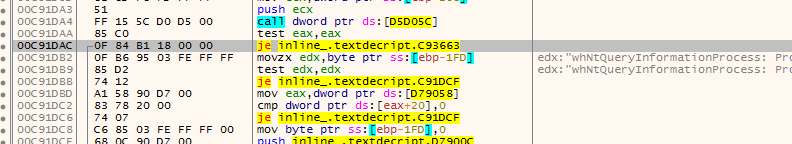


Tengo free space en 005900A pero original EP era 00D5D000 ahora lo cambiaré a 00CFB21e coloco inline para que cree un inline pero solo usaré el código, asi que no guardaré el exe

Ahora comienza lo tedioso y cansador, porqué no funciona en Windows 8.1, entonces teniendo un tiempo hablé con mr.exodia y me comentó que viera en la siguiente api “WaitfordebugEvent”



Aparece lo siguiente al retorno



Ese salto debe parcharse para que salte abajo xD cambio el salto





Y solucionado, ahora el programa Está inlineado y corre en Windows 8.1 esto es porque el comportamiento del padre/hijo es diferente, entonces al inlinear corre como padre, pero al cambiar el salto vuelve como hijo, algo asi es el tema, debe parcharse directamente, lo dejaré igual en el inline, para dejar constancia donde he parchado.

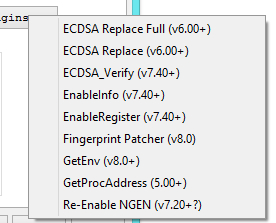
Al pulsar plugins los de interés son Enableinfo, enableregister, getenv(set…), solo necesitamos “enableinfo y enable register” para nuestro caso.

Luego por via comando colocaremos  **flowcode6.exe register**

El resultado es el siguiente

|  |
| --- |
| <$.95900A>  pushad  call @f  @@:  pop ebp  sub ebp, 959010 ; newentry+5-imagebase  ; Store imagebase  call @f  @getimagebase:  mov ebp, 0FFFFFFFF  ret  @@:  pop eax  mov dword ptr ds:[eax+1],ebp  ; Get API addresses  mov ebx, dword ptr ds:[ebp+95D04C] ; OutputDebugStringA  lea esi, dword ptr ds:[ebp+95D0A4] ; VirtualProtect  ; change page protection  call @f  "\x00\x00\x00\x00" ; oldprotect  @@:  push 40 ; newprotect  push 50 ; size  push ebx ; OutputDebugStringA  call dword ptr ds:[esi] ; VirtualProtect  ; IAT Hook VirtualProtect  call @vp\_hook\_end  @hook\_VirtualProtect:  pushad  pushfd  call @getimagebase  ; restore IAT hook  push esi  push eax  lea esi, dword ptr ds:[ebp+95D0A4] ; VirtualProtect  call @getvirtualprotect  xchg dword ptr ds:[esi],eax  pop eax  pop esi  ; go to the user code  jmp @usercode  @vp\_hook\_end:  pop ebp  xchg dword ptr ds:[esi],ebp  ; store old VirtualProtect  call @f  @getvirtualprotect:  mov eax,0FFFFFFFF  ret  @@:  pop eax  mov dword ptr ds:[eax+1],ebp  ; hook OutputDebugStringA  call @od\_skip  @od\_original\_bytes:  call @f  "\x90\x90\x90\x90\x90"  @@:  jmp short @od\_hook\_back  @od\_skip:  pop edi  add edi,5  mov esi,ebx  mov ecx,5  rep movs byte ptr es:[edi],byte ptr ds:[esi]  sub esi,5  mov byte ptr ds:[esi],0E9  call @od\_hook\_end  @hook\_OutputDebugStringA:  call @f  "\x02" ;counter  @@:  pop eax  dec byte ptr ds:[eax]  jz short @od\_execute\_hook  xor eax,eax  inc eax  ret 4  @od\_execute\_hook:  push edi  push esi  push ecx  push ebp  jmp short @od\_original\_bytes  @od\_hook\_back:  pop esi  call @getimagebase  mov eax,dword ptr ds:[ebp+95D04C] ; OutputDebugStringA  mov edi,eax  mov ecx,5  rep movs byte ptr es:[edi],byte ptr ds:[esi]  pop ebp  pop ecx  pop esi  pop edi  ; patch CRC values  push eax  mov dword ptr ds:[ebp-14],15D6D145  mov eax,dword ptr ds:[esp+4]  mov eax,dword ptr ds:[eax+A8]  mov dword ptr ds:[eax],CC68BE07  mov dword ptr ds:[eax+4],9ED5FFBA  mov dword ptr ds:[eax+8],AD03090C  mov dword ptr ds:[eax+0C],7DA949EC  pop eax  jmp eax ; OutputDebugStringA  ; continue hooking OutputDebugStringA  @od\_hook\_end:  pop eax  sub eax,5  sub eax,esi  mov dword ptr ds:[esi+1],eax  ; restore registers and jmp to oep  popad  jmp $.8FB21E ;rva of oep  @usercode:  ;PLACE YOUR CODE AFTER THIS (security base is in EDX, imagebase in EBP)  ;PLACE YOUR CODE BEFORE THIS  popfd  popad  call @getvirtualprotect  jmp eax |

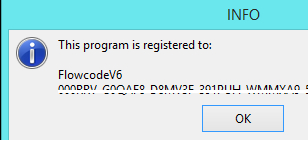
Ahora bien en esa parte amarilla habilitaremos enable register, con ayuda de los plugins cargado el programa me da los valores, pero además necesitamos cambiar para tener compatibilidad,



|  |
| --- |
| ; enable info  lea edi, dword ptr ds:[EDX+6EFE3]  mov dword ptr ds:[edi],889001B2  lea edi, dword ptr ds:[EDX+6F07E]  mov dword ptr ds:[edi],889001B0  ; enable register  lea edi, dword ptr ds:[EDX+06F030]  mov dword ptr ds:[edi],889001B1  ; Fix je win 8.1 WaitForDebugEvent return jump  ;must fix before run, if not the program is run in back as father  mov dword ptr ds:[C91DAC], 840F00  ; |

Pruebo el programa hecho con tanto cambio y Funciona!!! Pero falta el remate final el parche al salto

En palabras breves, Desencripte la sección pdata, luego inlineamos corrigiendo el crc, luego parchamos un error y ahora podemos ejecutar 2 comandos nuevos “register” y “info” pruebo un serial en certificado profesional , funciona



Si registro el programa puedo ejecutar con la versión del certificado, pero falta 1 parche más no se si recuerdan “el salto “ del programa

|  |
| --- |
| Entonces actualizo solo el final  ;PLACE YOUR CODE BEFORE THIS  popfd  popad  call @getvirtualprotect  jmp eax  @codecave2015:  pushad  pushfd  push 40 ; newprotect  push 6 ; size  push 00616B22 ; Parche salto  call @getvirtualprotect  call eax  push 40 ; newprotect  push 6 ; size  push 00616B22 ; Parche por segunda vez en caso x  call @getvirtualprotect  call eax  MOV Dword ptr ds:[00616B22],000113E9 ;parche del salto  MOV Dword ptr ds:[00CAAFDF],D790A068 ;parche del codecave  popfd  popad  jmp 00CAAFDF  <00CAAFDF>  jmp @codecave2015 |

Y listo ahora puedo probar el programa en los modos de c/u sin el mensaje de activar, hasta aquí está crackeado parcialmente.

# *Proceso 3: Keygening*

Para poder hacer bruteforce en los certificados pedí ayuda a mr.exodia porque su pc es 3 veces más rápido, pero todo comenzó por pedir ayuda en el inline, dado que en el caso particular que este Windows 8.1 me da un resultado diferente que en xp sp3 , bueno siguiendo armadillo para bruteforce en armadillo 9.xx el tiempo para lograr un resultado es considerablemente alta, desde horas a días por cada certificado, si hablamos de un rainbow con todos los hash posibles en un armadillo 6 son como 16gb, pero para un armadillo 9 son unos 76 terabytes osea mucha información.

En cuanto al tab de keygening luego de “un día de bruteforce”recuerden que es con salt y requiere inline “enableregister” que recién habíamos hablado tenemos esta información:

|  |
| --- |
| Global Information:  TimeStamp : 510FD974 First DWORD : 490A6EFA  Project ID : MatrixMultimedia\_Flowcode\_version\_6  Magic1 : 7794A181 Magic2 : 22F1  Salt : 582CB5E8 Crypt Seed : 8031601A (0xE, 0x7, 0x9, 0x4)  Public Certificate Information:  Short V3 Level 5: //trial \*UNKNOW VERSION\*  Chk : 3CA3CEC1 Sym : 24FE789B  Y : 545D3D5013E3B18E (MD5=DBC6F9E0) Pvt : C3140F6A023C03D5    Short V3 Level 5: //EXPIRED CERT  Chk : D4EE7867 Sym : DACD05FB  Y : 4A4E8289422F2947 (MD5=ED184877) Pvt : FC62F0F49B443B06  Short V3 Level 5: //Home version  Chk : C3D09F61 Sym : 40CE1811  Y : CDC49475431D0FF1 (MD5=459BFD75) Pvt : BEC99C540E5E8316  Short V3 Level 5: //Professional version  Chk : 9496226B Sym : 714304EF  Y : B8F7F77386F27509 (MD5=1FD9F1F9) Pvt : 53B541F64D50458E  Short V3 Level 5: //free/demo 30 days  Chk : B8814BF6 Sym : DEB7934D  Y : D6FD46C3C4442D30 (MD5=1EABDF32) Pvt : 5903BCC4DB9C6869  Short V3 Level 5: //50 user educational version  Chk : C2ACDD24 Sym : E9F41C4E  Y : 3392B6DFD982E40F (MD5=F92FD0B8) Pvt : 5A666B662457BD53  Short V3 Level 5: //10 user educational version  Chk : C1608250 Sym : 5FC0D4CC  Y : E85DEFFC5C967789 (MD5=8167FFEC) Pvt : 570606298BA3DAD8  Intercepted Libraries:  -\* |

Para generar una clave, podemos usar un keygen tool de armadillo, o bien la tool de vel, luego colocamos el “nivel ->short V3 level5”, “Name->lo que gusten”

nombre Sym Y private ( lo demás es generar la clave, checksum, magic, son pasos para obtener el sym y verificar que sean correctos ) en mi caso generaré claves para usuario “FlowcodeV6”

Primer Certificado:

|  |  |
| --- | --- |
| Tipo de licencia | Imagen |
| \*UNKNOW VERSION\* |  |
| Luego vemos esto:  Y en el programa: | | |

# 

Segundo Certificado

|  |  |
| --- | --- |
| Tipo de licencia | Imagen |
| Expired Version |  |

Tercer Certificado

|  |  |
| --- | --- |
| Tipo de licencia | Imagen |
| Home Version |  |

Cuarto Certificado

|  |  |
| --- | --- |
|  |  |

Quinto

|  |  |
| --- | --- |
|  |  |

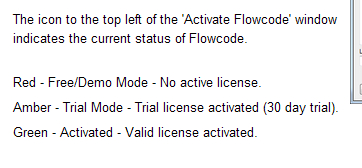
Sexto

|  |  |
| --- | --- |
|  |  |

Séptima

|  |  |
| --- | --- |
|  |  |

Si nos detenemos a saber las variables tenemos en una ayuda es que son 3 activaciones sobre el usuario actual y es según el color

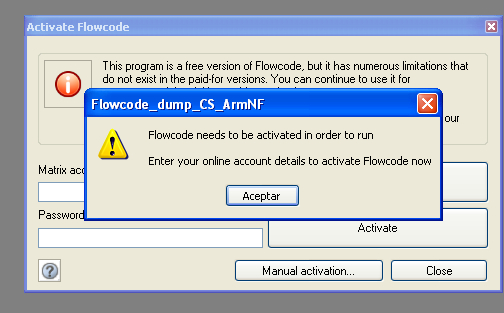


Las licencia tiene 3 versiones según los colores red=rojo, amber=amarillo +ambar , Green=verde que son determinadas según el environment Variable

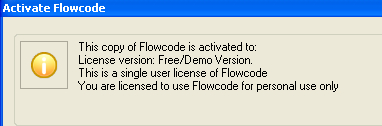
REGISTERED =”no” es rojo, REGISTERED =”yes” es ambar o verde según sea la licencia licencia “TYPE”(-1, 0, 1 entre otros).

Aquí el ejemplo

Free /demo –no active license:



Trial licence activated (30 days)



Activated-valid license activated

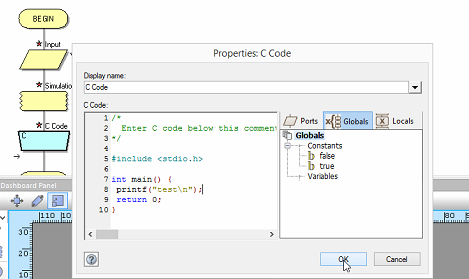
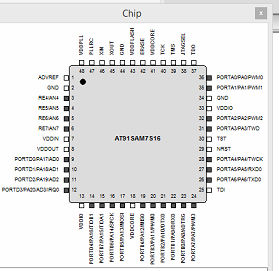


Creo que a esta altura el programa con las licencias está revisado, suficiente para dar comentarios finales

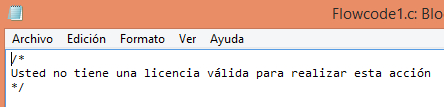
Comentarios Finales:

EL programa fue registrado, desempacado, inlineado, pero no quiere decir que armadillo 9 sea un mal protector, no, es bien complicado lograr buscar las herramientas suficientes para reparar todo , dumpeados que no corrían, inlines que no funcionaban, registros que no aceptaba la clave, programas que mostraban errores, osea realmente complicado, analizar luego de desempacar con una nag no es fácil, bueno en este caso tuvimos la oportunidad de vencer el programa en la modalida trial, ha pasado bastante desde que el programa está online , pero desde la versión 4 que aún siguen protegiendo con armadillo el mismo programa, a primera vista usa “secured section” asi que cada nivel de licencia es diferente en funcionamiento, es bien complicado registrarlo, pero gracias a la experiencia y herramientas se logró hacer bastante , desde aquí agradezco a cualquier reverser que ha revisado y se ha amanecido desempacando armadillo , desde aquí mis agradecimientos a aquellas herramientas, script, depuradores, y todo lo que permite que sea posible hacer hoy en día.

Si has logrado leer hasta el final este tutorial va dedicado a ti, pero el programa no es del todo operativo, pero en sí ya dura mas de 30 días así que con eso me conformo, pero …



Al exportar como comentario en c muestra



Asi que el programa en sí tiene otras limitantes.

Un saludo cordial a la lista y personas que suelen leer y son conocidas, saludos cordiales

Si fuera por resumir el escrito, simplemente les comento que he aprendido a usar herramientas de otras personas, lo importante es que aún sin manual es intuitivo usarlas pero mas aún las técnicas de unpacking para armadillo son públicas, pero el inline en armadillo no es algo único, y en gran parte creen que son ripper porque registran un armadillo, sin saber que han pasado un gran tiempo depurando el armadillo en sí, el código de security.dll y pdata falta múchisimo para documentar por completo, pero con lo comentado, con algo se comienza , bueno les mando un saludo es un escrito relativamente largo.

Pd: Armadillo es un buen protector, el tema es cuando se quita la protección ya no es lo mismo de bueno.

Atte

Apuromafo

