ESTUDIO DE PROTECCIONES BASICO PARA PRINCIPIANTES

(También llamado cracking)
Impartido por Ratón (Nivel principiante)

Nota
Cada capitulo ira acompañado de su crackme correspondiente.No facilitare paginas de donde bajarse herramientas ni enlaces a paginas de crackers, la intención es que busquéis en Internet todo lo necesario. Seguro que encontráis mas paginas de herramientas, tutoriales y utilidades relacionadas con este tema que las que yo pueda deciros.
Con esto solo quiero fomentar vuestro interés, además se que la búsqueda os proporcionara gratas sorpresas.
A todos un saludo.

Capitulo 5

Victima

abexcrackme3 creado por Abex

Herramientas

Olly Debugger.
Peid – Detector de protecciones.
Cracker tools o calculadora de Windows
Instinto

Objetivo

Conocer otro tipo de protección.
Comenzar a ver la utilidad de las APIs de Windows
Ver conversión hexadecimal – decimal.

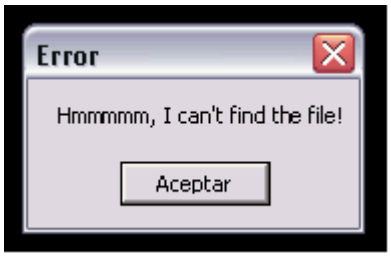
Al ataque

Este tipo de protección esta basado en que para registrarlo debemos tener un archivo conteniendo alguna clave o “algo” que el programa comparara y si es el archivo correcto y con el contenido de registro que nos pide nos dejara registrarnos.
Seguro que habéis probado algún programa share que para registrarlo os pide un archivo con extensión *.reg o *.key ese es el tipo de protección que tiene este crackme.
Se la conoce como protección keyfile.

Copia del crackme.
Análisis con Peid.

Ejecutamos crackme

Pulsamos aceptar



Buscamos la cadena en Olly, la vemos en 0040107C - vamos al principio de la instrucción (00401075) - pulsamos Control + R - vemos que esta referenciado desde el salto 00401037 – justo encima del salto un Call.
La misma rutina de siempre Call – Cmp –Je
Ponemos un BP (Breakpoint) con F2 en ese Call 0040102A

Nos fijamos en el Call en el que hemos puesto el BP que es una llamada a CreateFileA una API de Windows (cracker notes para aprender mas)

```
00401000 $ 6A 00          PUSH 0                                   ; Style = MB_OK|MB_APPLMODAL
00401002 . 68 00204000    PUSH abexcrac.00402000                   ; Title = "abex' 3rd crackme"
00401007 . 68 12204000    PUSH abexcrac.00402012                   ; Text = "Click OK to check for the keyfile."
0040100C . 6A 00          PUSH 0                                   ; hOwner = NULL
0040100E . E8 8C000000    CALL <JMP.&USER32.MessageBoxA>           ; MessageBoxA
00401013 . 6A 00          PUSH 0                                   ; hTemplateFile = NULL
00401015 . 68 80000000    PUSH 80                                  ; Attributes = NORMAL
0040101A . 6A 03          PUSH 3                                   ; Mode = OPEN_EXISTING
0040101C . 6A 00          PUSH 0                                   ; pSecurity = NULL
0040101E . 6A 00          PUSH 0                                   ; ShareMode = 0
00401020 . 68 00000080    PUSH 80000000                            ; Access = GENERIC_READ
00401025 . 68 B9204000    PUSH abexcrac.004020B9                   ; FileName = "abex.l2c"
0040102A . E8 5E000000    CALL <JMP.&KERNEL32.CreateFileA>         ; CreateFileA
0040102F . A3 CA204000    MOV DWORD PTR DS:[4020CA],EAX
00401034 . 83F8 FF        CMP EAX,-1
00401037 .v74 3C          JE SHORT abexcrac.00401075
00401039 . 6A 00          PUSH 0                                   ; pFileSizeHigh = NULL
0040103B . FF35 CA204000  PUSH DWORD PTR DS:[4020CA]               ; hFile = NULL
00401041 . E8 4D000000    CALL <JMP.&KERNEL32.GetFileSize>         ; GetFileSize
00401046 . 83F8 12        CMP EAX,12
00401049 .v75 15          JNZ SHORT abexcrac.00401060
0040104B . 6A 00          PUSH 0                                   ; Style = MB_OK|MB_APPLMODAL
0040104D . 68 35204000    PUSH abexcrac.00402035                   ; Title = "Well done!"
00401052 . 68 40204000    PUSH abexcrac.00402040                   ; Text = "Yep, keyfile found!"
00401057 . 6A 00          PUSH 0                                   ; hOwner = NULL
00401059 . E8 41000000    CALL <JMP.&USER32.MessageBoxA>           ; MessageBoxA
0040105E .vEB 28          JMP SHORT abexcrac.00401088
00401060 > 6A 00          PUSH 0                                   ; Style = MB_OK|MB_APPLMODAL
00401062 . 68 79204000    PUSH abexcrac.00402079                   ; Title = "Error"
00401067 . 68 7F204000    PUSH abexcrac.0040207F                   ; Text = "The found file is not a valid keyfile!"
0040106C . 6A 00          PUSH 0                                   ; hOwner = NULL
0040106E . E8 2C000000    CALL <JMP.&USER32.MessageBoxA>           ; MessageBoxA
00401073 .vEB 13          JMP SHORT abexcrac.00401088
00401075 > 6A 00          PUSH 0                                   ; Style = MB_OK|MB_APPLMODAL
00401077 . 68 54204000    PUSH abexcrac.00402054                   ; Title = "Error"
0040107C . 68 5A204000    PUSH abexcrac.0040205A                   ; Text = "Hmmmm, I can't find the file!"
00401081 . 6A 00          PUSH 0                                   ; hOwner = NULL
```

Corremos el programa con F9, aceptamos la primera ventana y para el Olly en el Call.
Entramos en el Call con F7 (una vez).

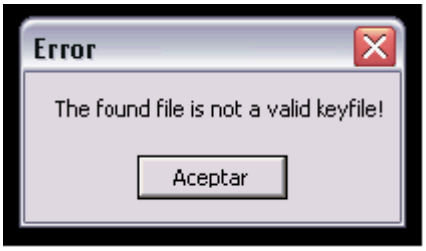
Al pulsar F7 aparecemos en 0040108D vemos la llamada a la API.
Vamos examinando con F8

Con F8 unas pocas instrucciones después vemos una cadena en la ventana registers abex.l2c
Recordamos que para registrar este crackme tenemos que tener un archivo “extra”, en los programas comerciales tendrían la extensión *.reg o *.key generalmente
Vemos que la cadena es abex- extensión- .l2c

Como saber no sabemos mucho, la verdad, pero tenemos una gran intuición pudiera ser que el programa nos pidiera ese archivo.
Cerramos Olly
Vamos a la carpeta del crackme y con el bloc de notas creamos un archivo llamado abex.l2c y lo dejamos vació, no escribimos nada en el.

Abrimos Olly y cargamos el crackme, ponemos un BP en el salto 00401037 a continuación del Call F9, aceptamos y para Olly en el Call
F9 y para Olly en el salto
Observamos que el salto no es tomado, por tanto no nos lleva al cartel que nos dice que no puede encontrar el archivo (can´t find the file)
Hemos creado el archivo que el crackme nos pedia.

Si ejecutamos el programa ahora no aparecerá e cartel que nos decía que no encontraba el archivo, pero aparece este otro



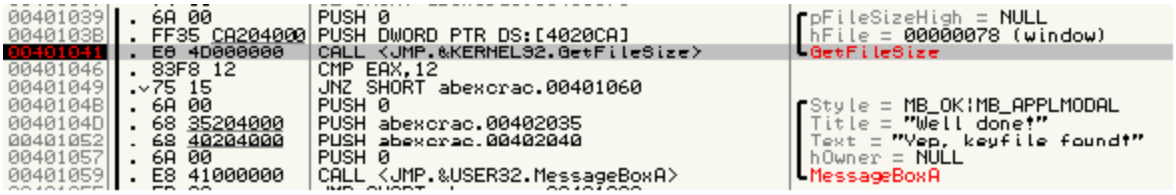
Que nos dice que aunque ha encontrado el archivo este no es valido, claro, esta vació no tiene nada escrito.
Vamos a averiguar que condiciones tiene que cumplir el keyfile que hemos creado para que el programa lo acepte.

Vemos que la String The found file is not a valid keyfile esta en la dirección 00401067 y el principio de la rutina esta en 00401060 nos ponemos encima y Control + R para ver las referencias y vemos que hay un salto condicional (00401049) que nos lleva hasta aquí.

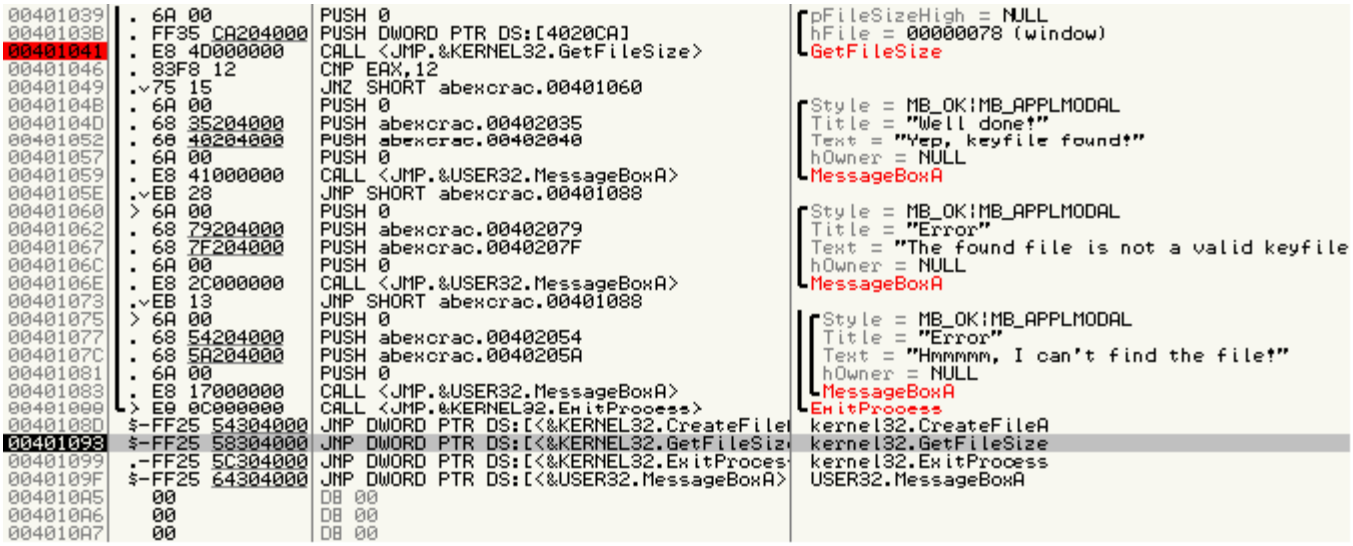
Encima de este salto en la dirección 00401041 hay un Call que llama a la API GetFileSize y luego

un CMP (EAX,12)

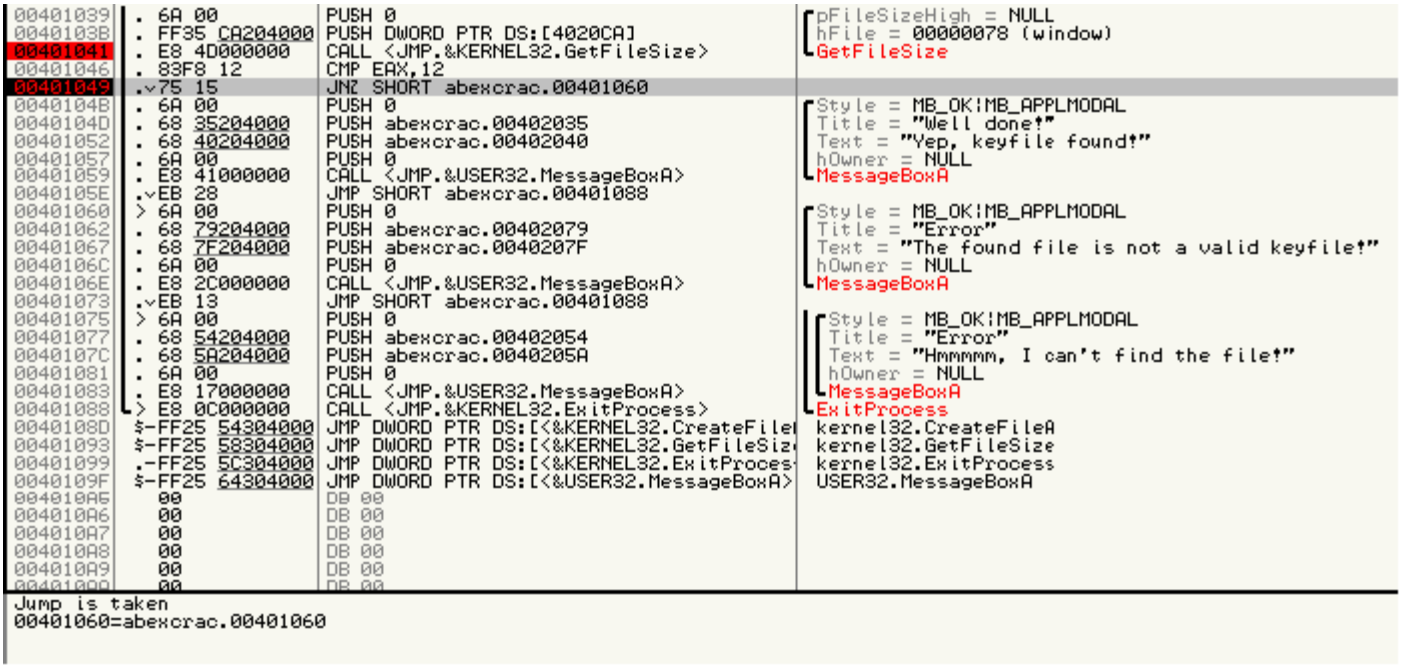
Ponemos un BP en el Call 00401041 y ejecutamos el programa aceptando ventanas hasta que Olly nos pare en el Call



Pulsamos F7 para entrar en el Call



Y seguimos con F8 instrucción a instrucción mirando el apartado registers, pero salimos del Call en la dirección 00401046 sin ver nada dentro del Call y el resultado del salto no nos conviene para nuestros fines pues saltaría y veríamos el cartel The found file is not a valid keyfile



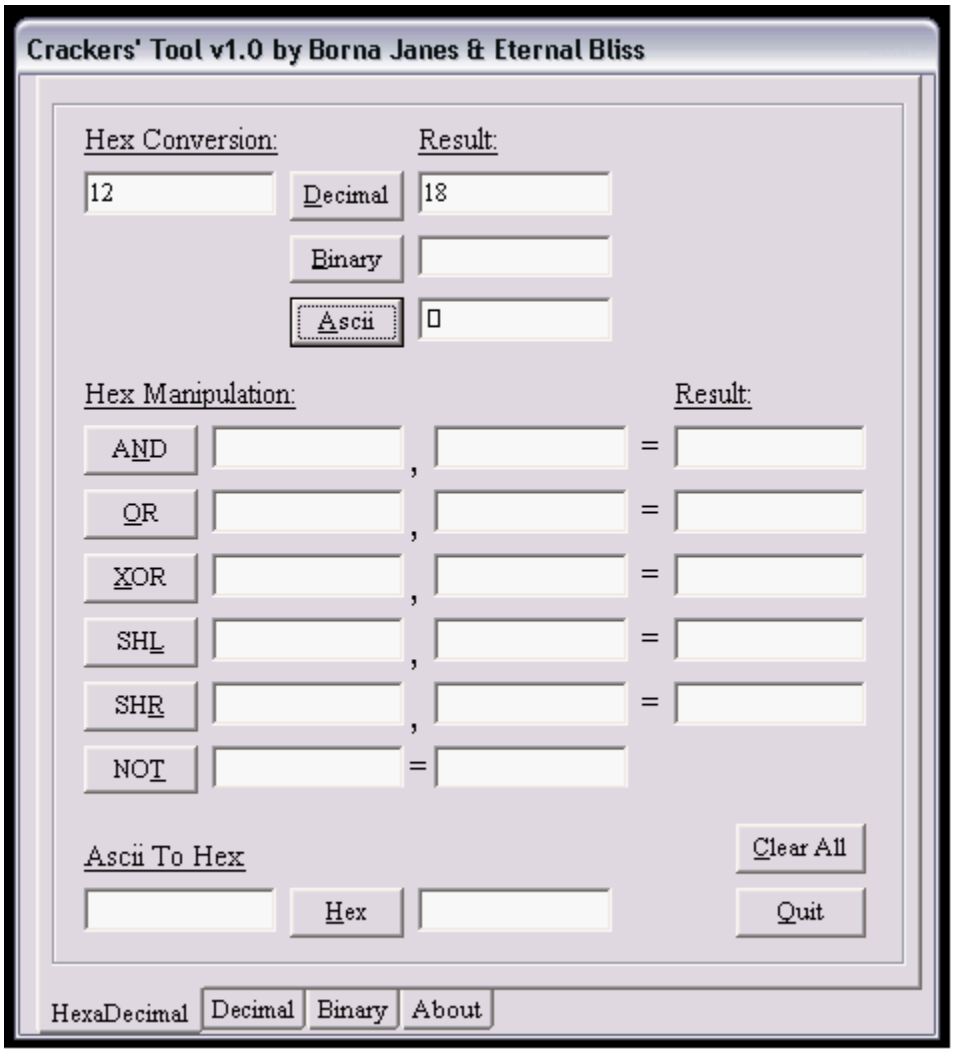
Si dentro del Call no hemos visto nada la solución debe de estar en 00401046 CMP EAX,12
Esta instrucción compara el valor de EAX con 12.
Con 12 que ¿?
Fijémonos en el nombre de la API que hay en el Call antes de la comparación: GetFileSize significa algo así como coge el tamaño (talla) del archivo

Si tenemos claro que los archivos están compuestos por bytes lo que hace esta API es pedir el tamaño en bytes del archivo.
Luego lo compara con 12 y si es igual el salto no se ejecuta y pasamos directamente a una parte del código en la que vemos que se genera la MessageBox Well done! Yep keyfile found

Tenemos que rellenar nuestro archivo abex.l2c con 12 bytes ¿?

Si miramos un poco la tabla ASCII del capitulo 0 vemos que existen mas tipos de numeración que el decimal, el hexadecimal por ejemplo.
Los números que vemos en el código están en base hexadecimal excepto que haya una etiqueta ASCII o unicode que los identifique como hemos visto al buscar seriales en la ventana registers en anteriores capítulos que siempre llevan ASCII y el número entre comillas.

Si miramos la tabla ASCII del capitulo 0 o utilizamos un convertidor como el crackers tool o abrimos la calculadora científica de Windows podremos hacer la conversión de 12 hexadecimal a numero decimal



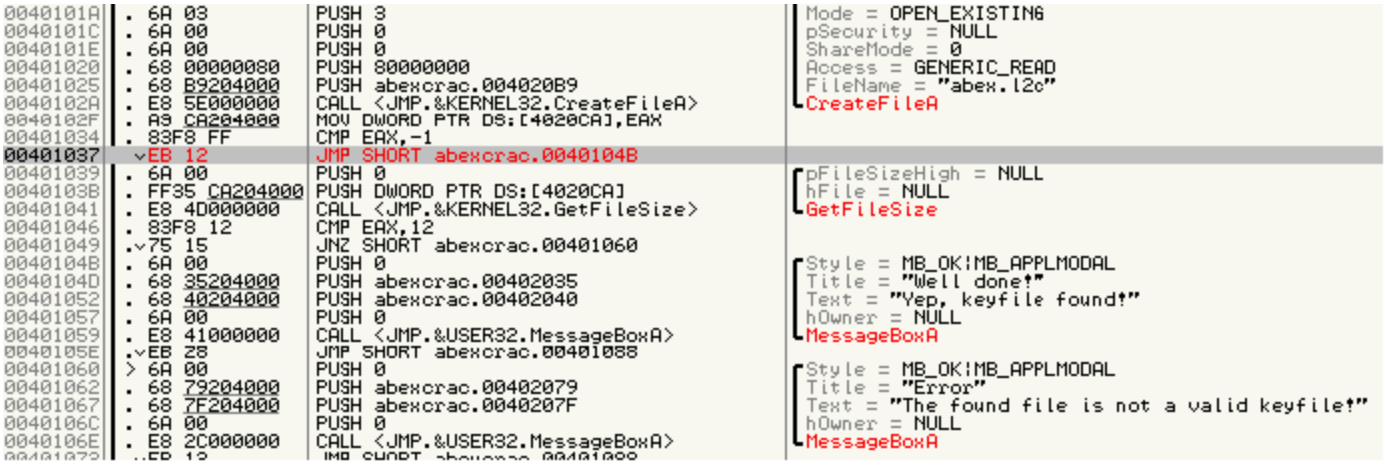
12 hexadecimal = 18 decimal

Debemos llenar nuestro archivo abex.l2c hasta que pese 18 bytes.
Ahora al estar vació “pesa” 0 bytes
Lo abrimos con el bloc de notas y escribimos el numero 1 por ejemplo cerramos el bloc guardando los cambios
Examinamos cuanto pesa ahora el archivo: un miserable byte
Guiados por nuestra lógica vamos a probar a escribir 18 unos (111111111111111111) a ver si así alcanzamos el peso o “damos la talla”
Cerramos el bloc guardando los cambios y examinamos el peso del archivo
Hemos conseguido el peso ideal sin necesidad de dietas ni otras coñas marineras.

Cerramos Olly y probamos el crackme y vemos que encontramos la solución.

Otra manera de solucionarlo pero sin aprender nada nuevo

Espero que a estas alturas del curso os hayáis dado cuenta, fijaros que solo necesito cambiar el valor de uno de los saltos y el programa aparece como registrado.



Otra manera Las llamadas a la APIs

Creo que es el momento de entrar en este tema.

Para resolver este crackme hemos puesto dos BP en dos Call:

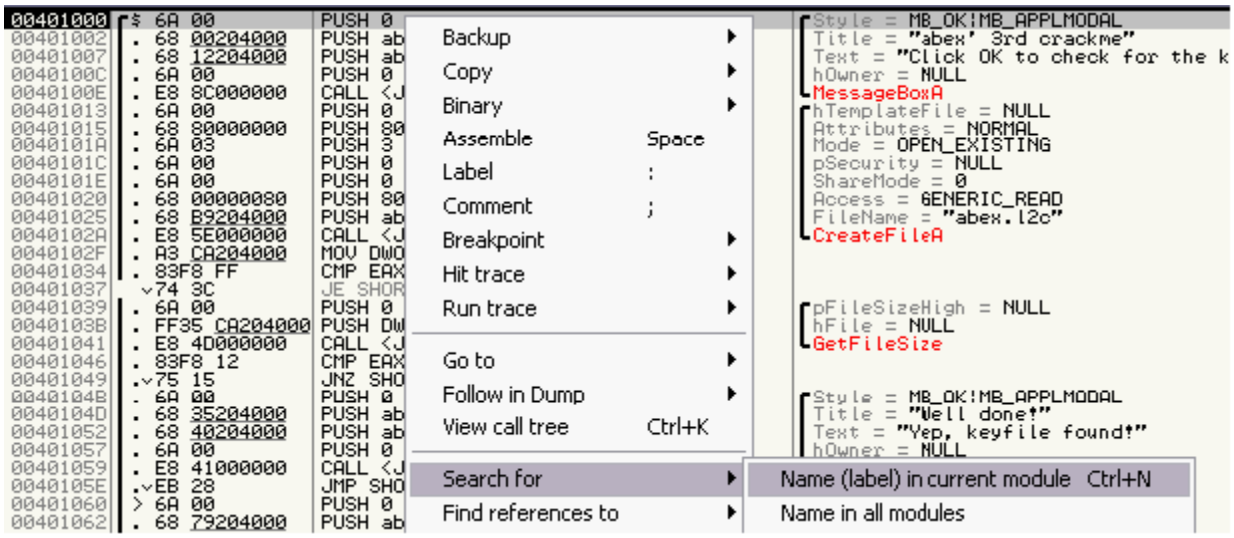
Primero para ver donde nos pedían el keyfile o archivo de registro en 0040102A que llamaba a la API CreateFileA y donde nos dimos cuenta que nos hacia falta un archivo llamado abex.l2c

Segundo para ver donde se comparaba “la talla” o “peso en bytes” del archivo keyfile en 00401041 y llamaba a la API GetFileSize y vimos que el peso en bytes del archivo abex.l2c era 18 (12 en hexadecimal)

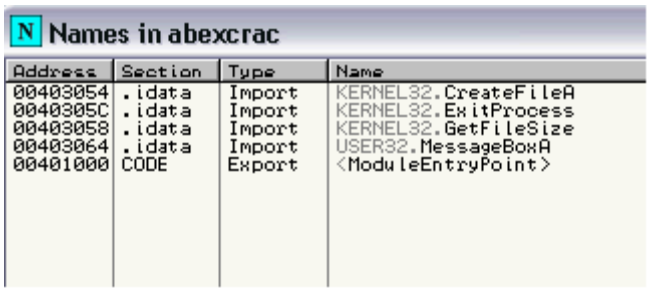
Vamos a utilizar las APIs para registrar el programa, pero donde podemos ver la APIs ¿?

Con click derecho en cualquier sitio de la ventana de ensamblado Search for – Name (label) in

current module o Control + N



Aparece esta ventana que nos muestra las APIs del crackme de abex y su dirección

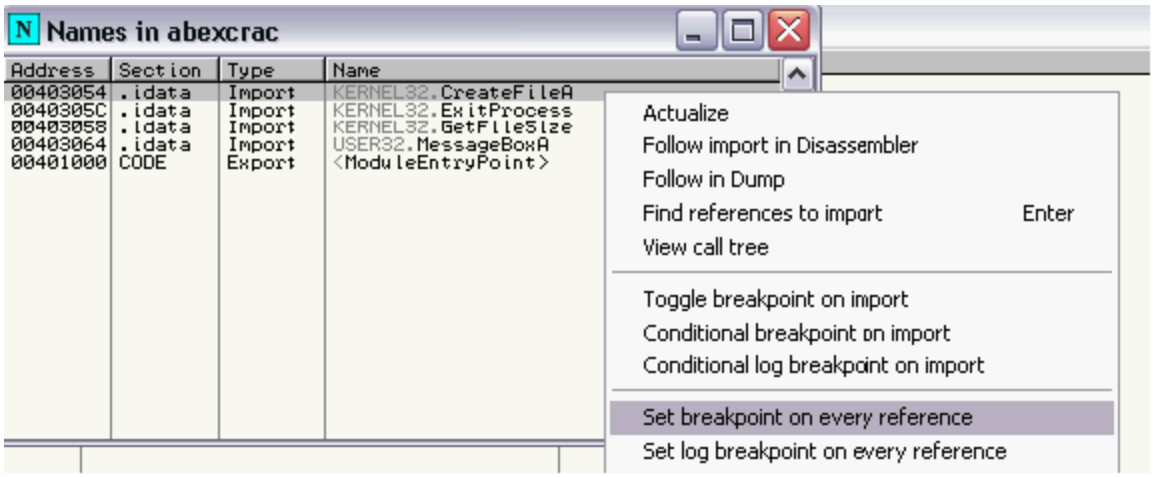


Ahora que sabemos que la API CreateFileA nos sirve para ver donde se compara si existe o no el keyfile seria sencillo al abrir un crackme de este tipo ir directamente a Olly y decirle que se detenga cada vez que encuentre uno

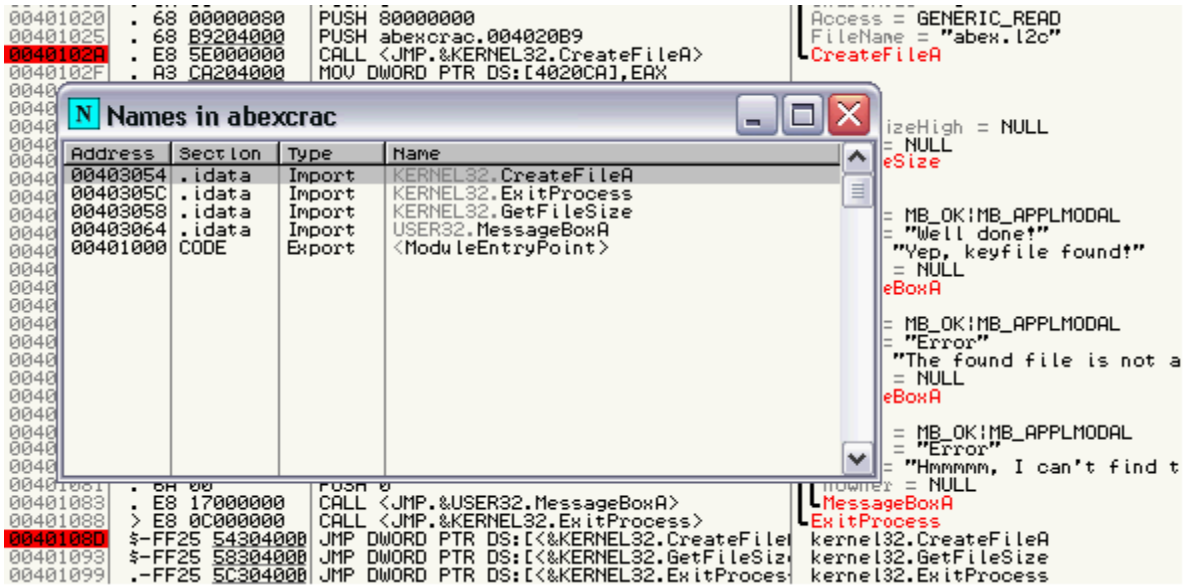
Lo hacemos como se ve en la imagen: nos colocamos encima de la API que nos interese y click derecho Set Breakpoint on every reference

La traducción real seria algo así como pon un BP en cada referencia (a esta API)

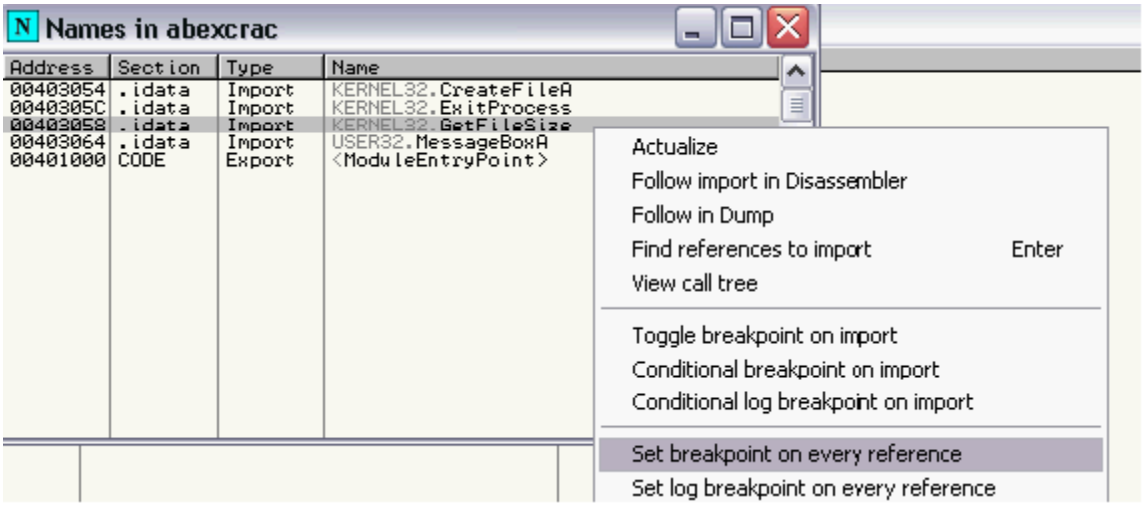
Traducción ratonil = cuando encuentres la API ya te estas parando en ella que quiero analizarla, Olly majete



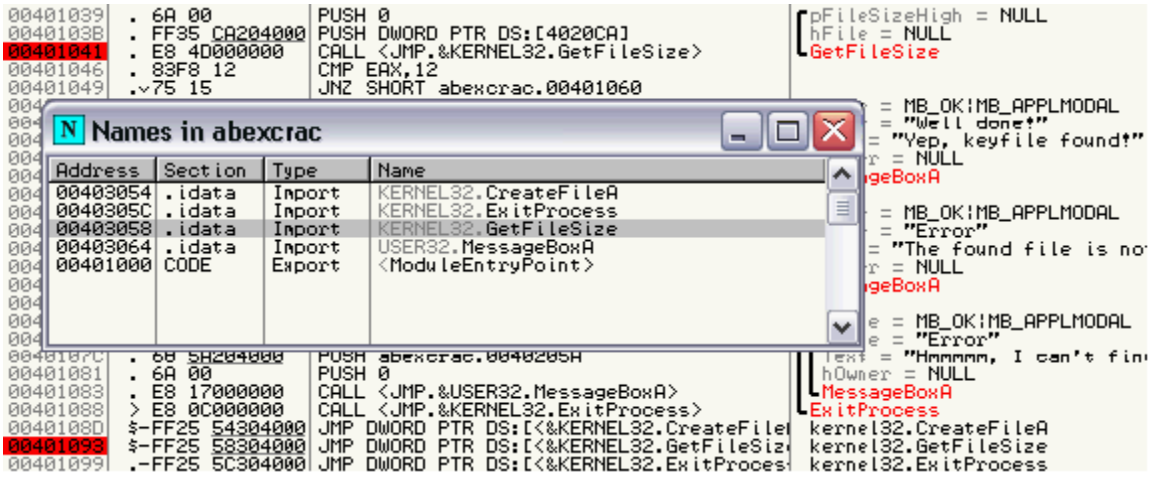
Vemos en el código que la primera API marcada corresponde al primer Call que analizamos para ver si había o no keyfile y tuvimos que crear el archivo abex.l2c



Una vez analizado este Call y resuelto el problema de la existencia del keyfile haríamos lo mismo para encontrar la talla utilizando la llamada a la API GetFileSize



Aquí se puede ver el resultado de marcar los BP de esta API, nos pone el BP en el segundo Call donde miramos los bytes que debía pesar el archivo abex.l2c



Podíamos haber cambiado el valor del JMP para registrarnos pero en este crackme tan sencillo hubiera sido un remedio pobre, es mejor habernos creado nuestro propio keyfile con el tamaño adecuado, no creéis ¿?
Entre los caminos que llevan a Roma a veces merece la pena coger el más largo pues a lo mejor el paisaje nos depara alguna sorpresa agradable

Interesa que vayáis quedándoos con la idea y busquéis información sobre las APIs pues son fundamentales, ya veréis de aquí en adelante.
Podéis empezar por cracker notes y seguir por Google.

Probad a crackearlo utilizando las llamadas a las APIs y por supuesto sin el tutorial delante.

Gracias

A todas las personas que colaboran desde el foro de HackxCrack para llevar adelante el curso, tanto los que colaboran aportando sus conocimientos como complemento al curso como a los que postean sus dudas para que aprendamos todos y por supuesto a los moderadores del mismo
A todos los crackers y programadores de los cuales he aprendido y sigo aprendiendo.
A los creadores de crackmes
En especial y sin menospreciar a nadie a Ricardo Narvaja por su aportación y su trabajo sobre el estudio de las protecciones y sus tutoriales en castellano y a Makkakko por sus tutoriales con Olly Debugger (Recomendados 100%) y por supuesto a Shoulck por la ayuda desinteresada que me esta prestando a costa de algo tan preciado como su tiempo.
A ti que me estas leyendo.

Ratón Enero 2004