Parcheando unos componentes

Fecha	
Victima	Packetes de componentes de *Utilmind Solutions*
URL de descarga	http://www.appcontrols.com/download.html
Protección	Trials sin posibilidad de registrarlos
Herramientas	C++Builder 6, OllyDbg 1.10, WinHex, Total Commander Ultimate Prime (TCUP)
Objetivo	Conseguir una versión no Shareware o al menos algo que se le parezca
Dificultad	Fácil - Media
Cracker	Aguml

Indice

1. Disk Controls – Presentación

Descripción: *Disk Controls es un conjunto de 22 componentes que pueden hacer su vida mucho más fácil a la hora de desarrollar software que trabaje con discos (duro / blando / CD / RAM / Red), directorios y sistema de archivos.*

Puedo ver el contenido de la pestaña del paquete de componentes Disk Controls:

Si miro la compatibilidad veo esto:
Disk Controls compatible with Delphi 3/4/5/6/7 and BCB 3/4/5/6, and has been tested on Win95, Win95OSR2, Win98, WinME, NT4, Win2000 and WinXP.

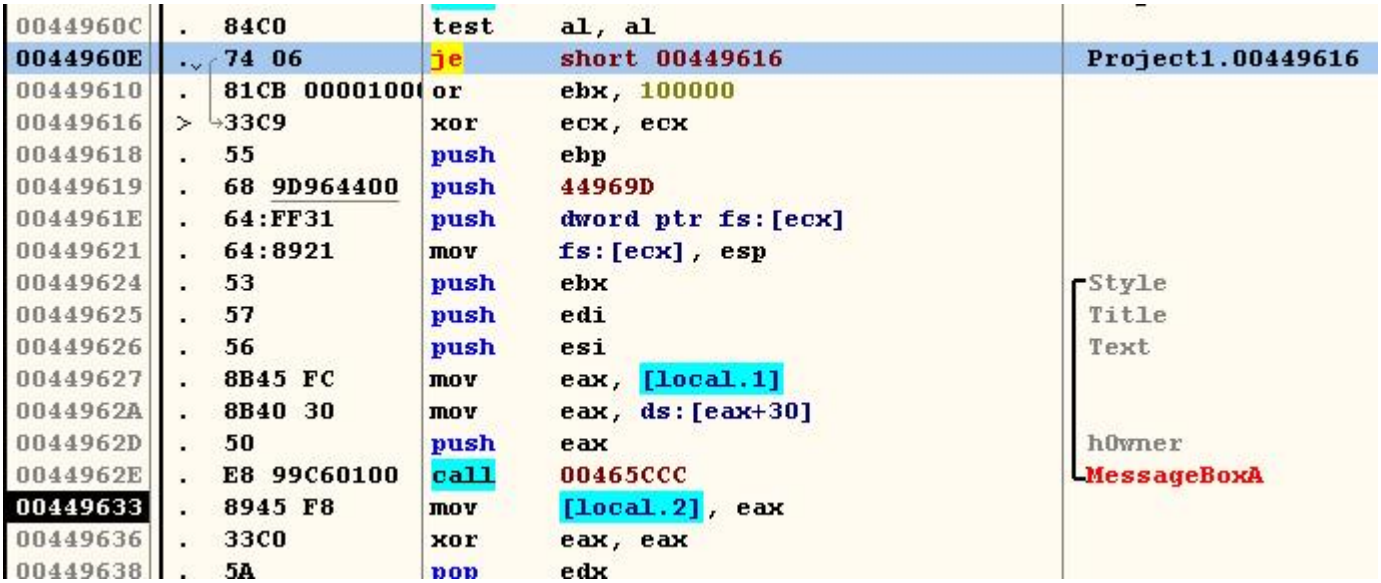
Puedo ver que le queda poca vida ya que no vale para versiones modernas de Windows y tampoco para versiones modernas de C++Builder o Delphi pero la motivación no es sacar provecho económico sino aprender y mejorar.

2. Disk Controls – Analisis de la protección

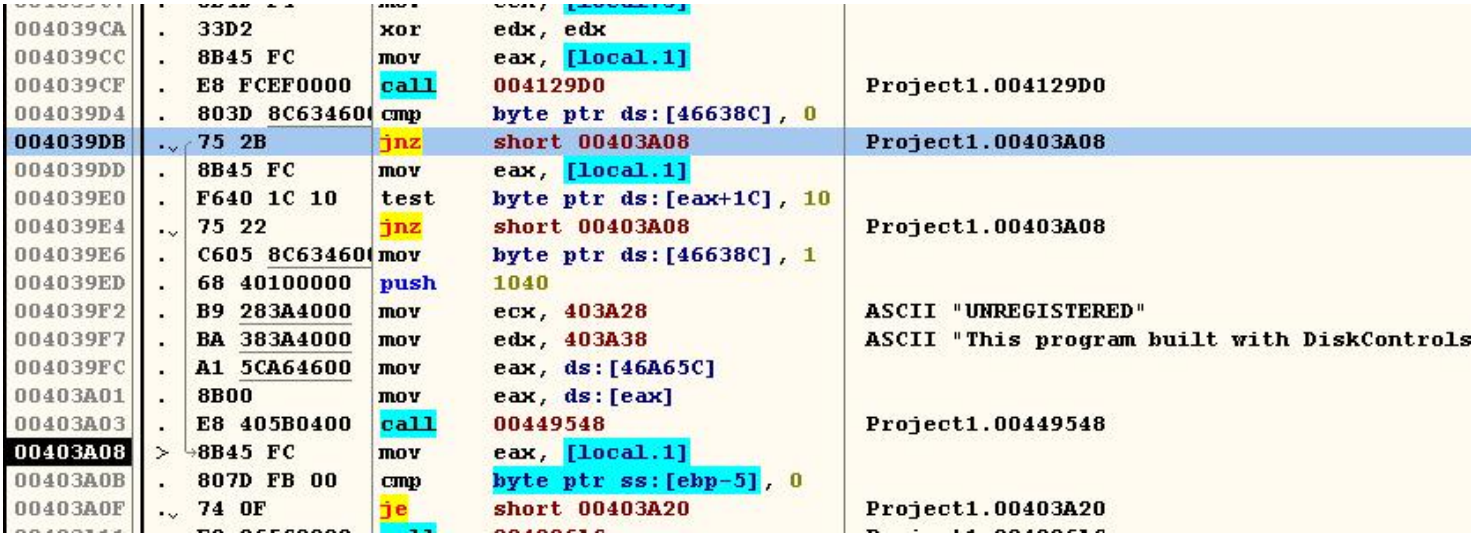
Si creo un proyecto, añado alguno de sus componentes, compilo y ejecuto veo esto:

Esa es la única protección que trae así que usaré ese ejecutable para analizarlo en Olly y ver como poder quitar esa NAG.

Cargo el ejecutable creado por C++Builder en el Olly y doy a F9 para iniciar su ejecución y nos saldrá la NAG. Acto seguido doy a F12 para pausar la depuración y voy con F7 y Ctrl + F9 hasta que el Olly se quede esperando una acción y entonces doy al botón “Aceptar” de la NAG. La NAG se cerrará y en Olly caigo en un RETN. Sigo con F7 y Ctrl + F9 hasta llegar a la zona de código del ejecutable:



Estoy dentro de un CALL y no hay nada interesante así que sigo con Ctrl + F9 y F7 hasta que llego aquí:



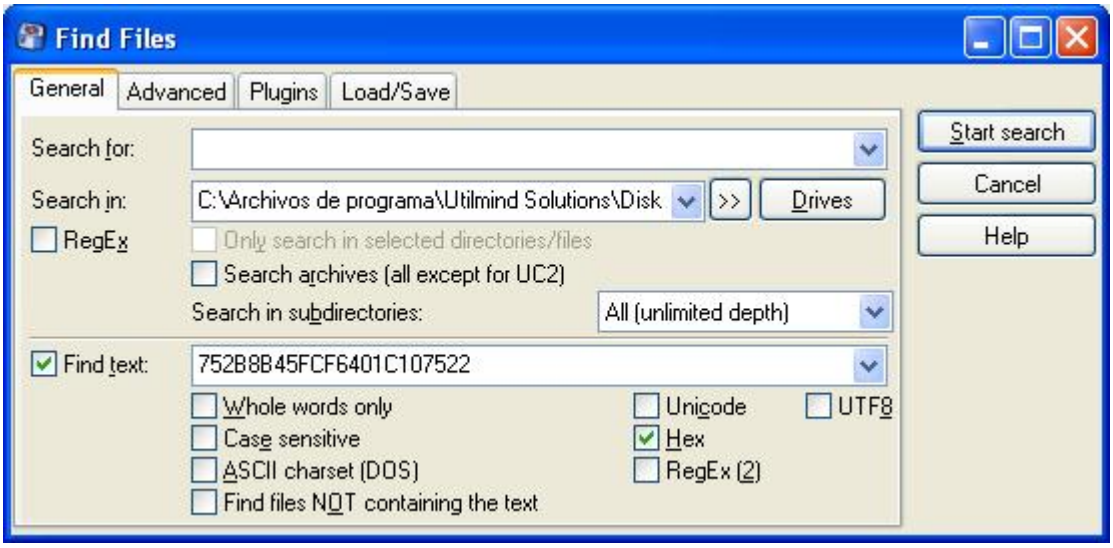
Los dos saltos de arriba parecen prometedores así que pongo un BP en el primero, reinicio Olly y, una vez pare, fuerzo el salto y ya no me sale la NAG. ¡Adiós NAG! jejeje.
Esto está muy bien pero no podemos andar parcheando cada ejecutable que creemos así que toca parchear el paquete de componentes para que salgan los ejecutables ya parcheados.
Lo primero que necesito es obtener una firma binaria a buscar y yo me he decantado por esta:

```
75 2B 8B 45 FC F6 40 1C 10 75 22
```

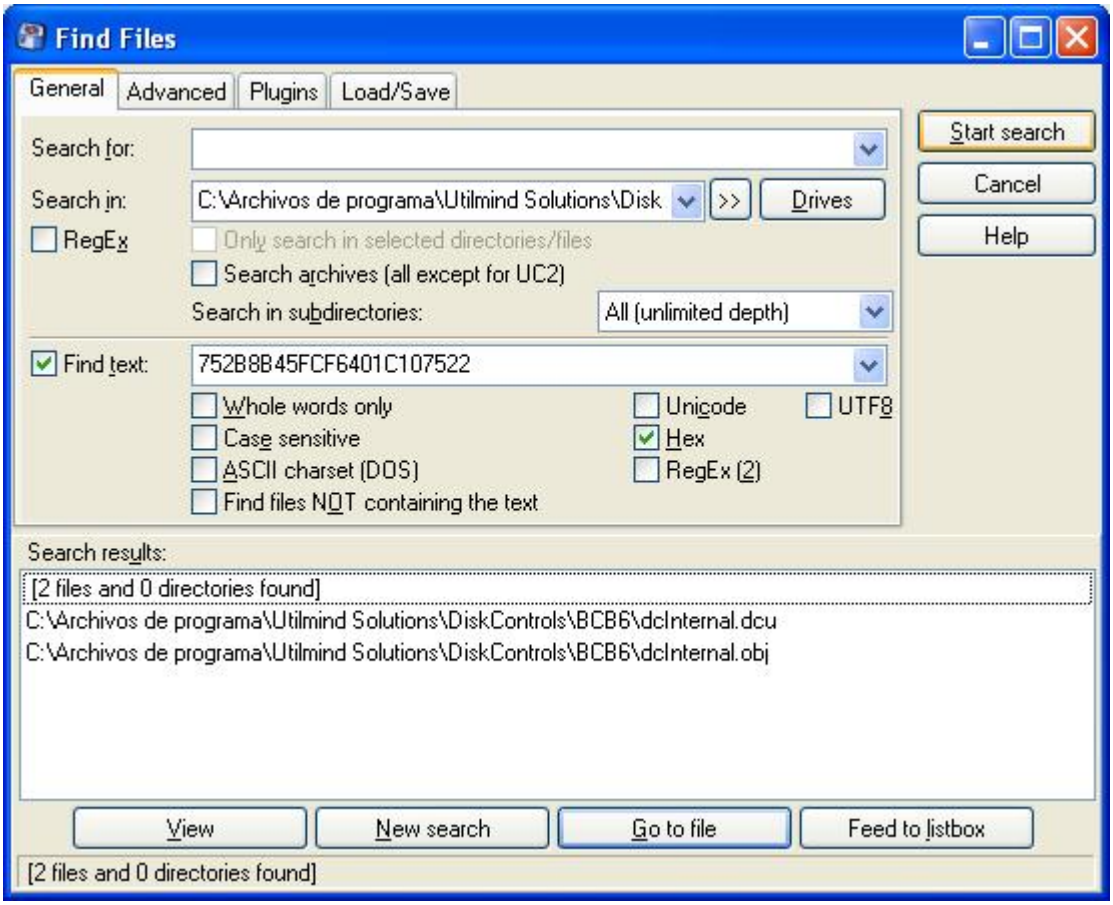
Es el binario desde el primer salto al Segundo (ambos incluidos).

3. Disk Controls – Parcheando el componente

Lo primero es buscar la firma binaria en todos los archivos del paquete de componentes y para eso usaré TCUP. Abro TCUP y doy a “Commands→Search…” y me sale esta ventana:



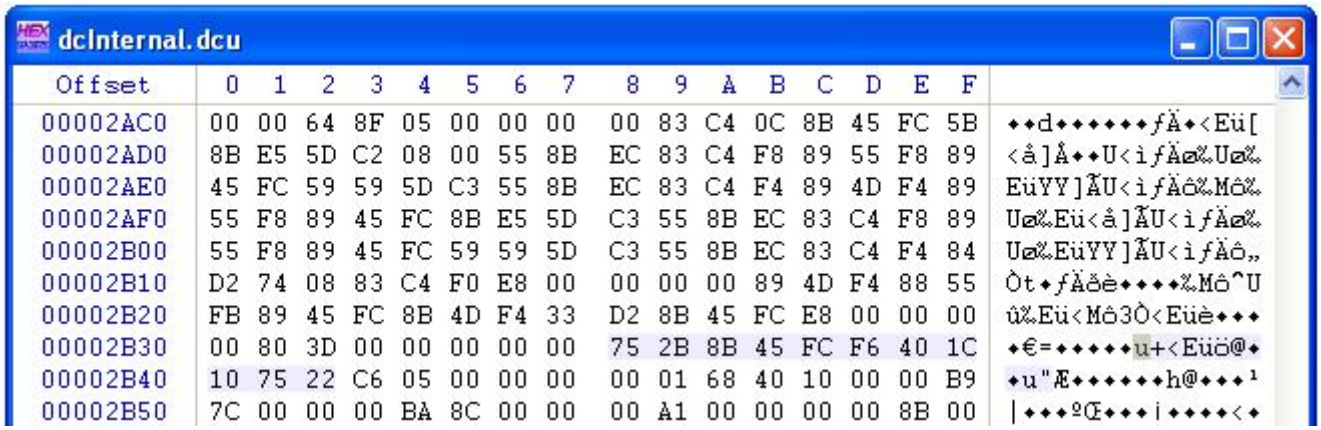
Coloco la ruta en “Search in”, marco la opción “Find text”, meto la firma binaria, marco la opción “Hex” y doy al botón “Start search” y me muestra esto:



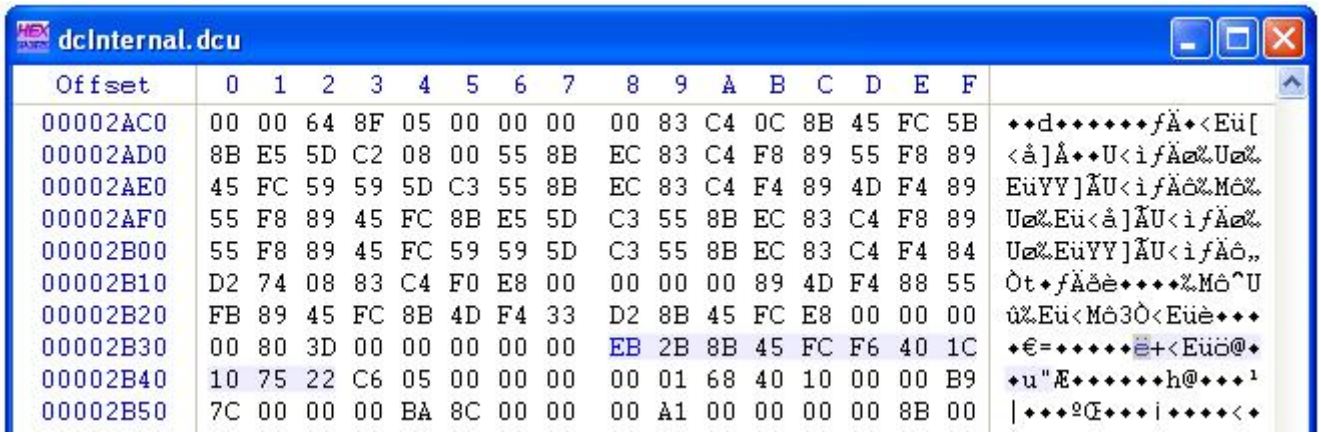
Pues ya sé cuáles parchear y para ello usaré WinHex.

Abro el archivo dcInternal.dcu en WinHex y doy al botón de “Busqueda binaria”  y mostrará esta ventana a la cual le coloco la firma binaria que tengo y doy en “Aceptar”:

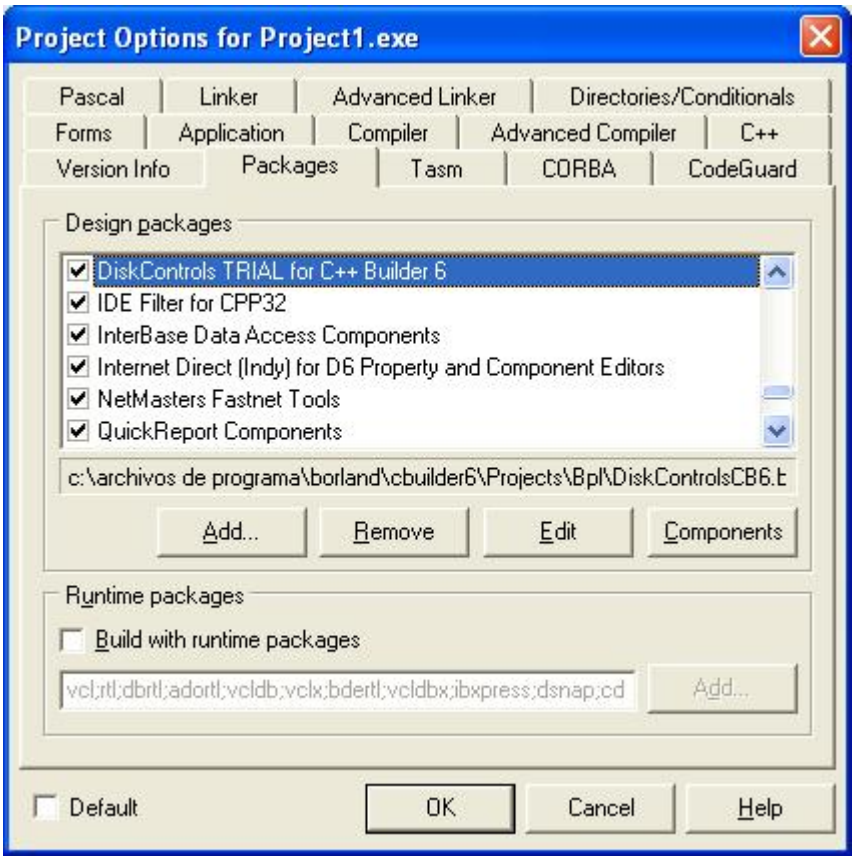
Y me manda aquí:



No hay más coincidencias con lo que lo modifico sin miedo a equivocarme de lugar:



Hago la misma operación con el archivo dcInternal.obj y ya tengo el componente parcheado.
Para probarlo tendré que abrir primero C++Builder, ir a "Project→Options", dar a la pestaña "Packages", buscar el paquete y eliminarlo dando al botón "Remove":



Luego voy al directorio donde se creó el archivo .lib. En mi caso se creó en la siguiente ruta:
C:\Archivos de programa\Borland\CBuilder6\Projects\Lib

Y veo que hay dos archivos (DiskControlsCB6.bpi y DiskControlsCB6.lib) los cuales elimino.
Luego voy al directorio donde se creó el .bpl. En mi caso se creó en la siguiente ruta:
C:\Archivos de programa\Borland\CBuilder6\Projects\Bpl

Y veo que también hay dos archivos (DiskControlsCB6.bpl y DiskControlsCB6.tds) los cuales elimino también.

Una vez hecho esto vuelvo a instalar el paquete de componentes y ya no me vuelve a salir nunca más la NAG.

1. Advanced Application Controls – Presentación

Bueno, como el DiskControl fue facilón, decido mirar el AppControls que es del mismo fabricante para ver si el modo de parchearlo es algo estándar o no.

Descripción: *Advanced Application Controls (AppControls) es un conjunto de más de 92 componentes de usos múltiples de primera calidad para todas las versiones de Delphi y C ++ Builder en Win32. El paquete contiene todo lo necesario para agregar y, más importante, dar apariencia realmente profesional para su software, por lo que el desarrollo de grandes interfaces y servicios internos serán rápidos. Todos los componentes están muy integrados entre sí y extremadamente fáciles de usar.*

Podemos ver la paleta de los componentes del Advanced Application Controls:

Si miramos la compatibilidad vemos esto:
AppControls compatible with Delphi 2/3/4/5/6/7/2005/2007, Borland Developer Studio 2006, and BCB 3/4/5/6, and has been tested on Win95, Win95OSR2, Win98, WinME, NT4, Win2K, WinXP and Win2003 Server.

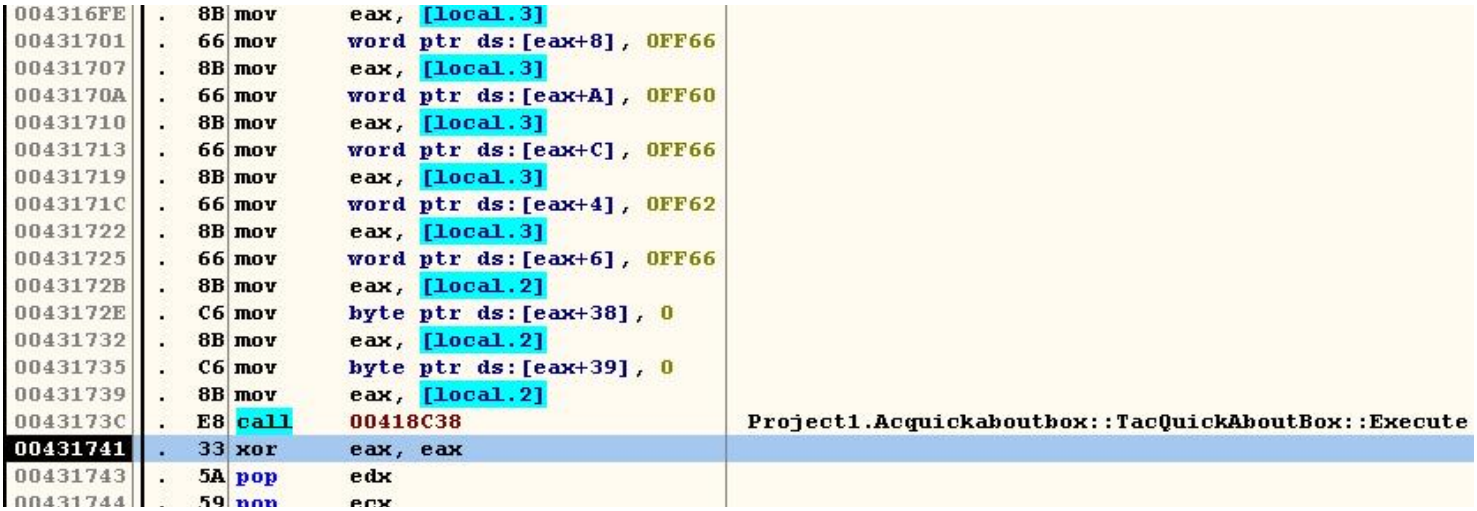
Puedo ver que también le queda poca vida ya que tampoco vale para versiones modernas de Windows y tampoco para versiones modernas de C++Builder o Delphi pero, al igual que antes, la motivación no es sacar provecho económico sino aprender y mejorar.

2. Advanced Application Controls – Analisis de la protección

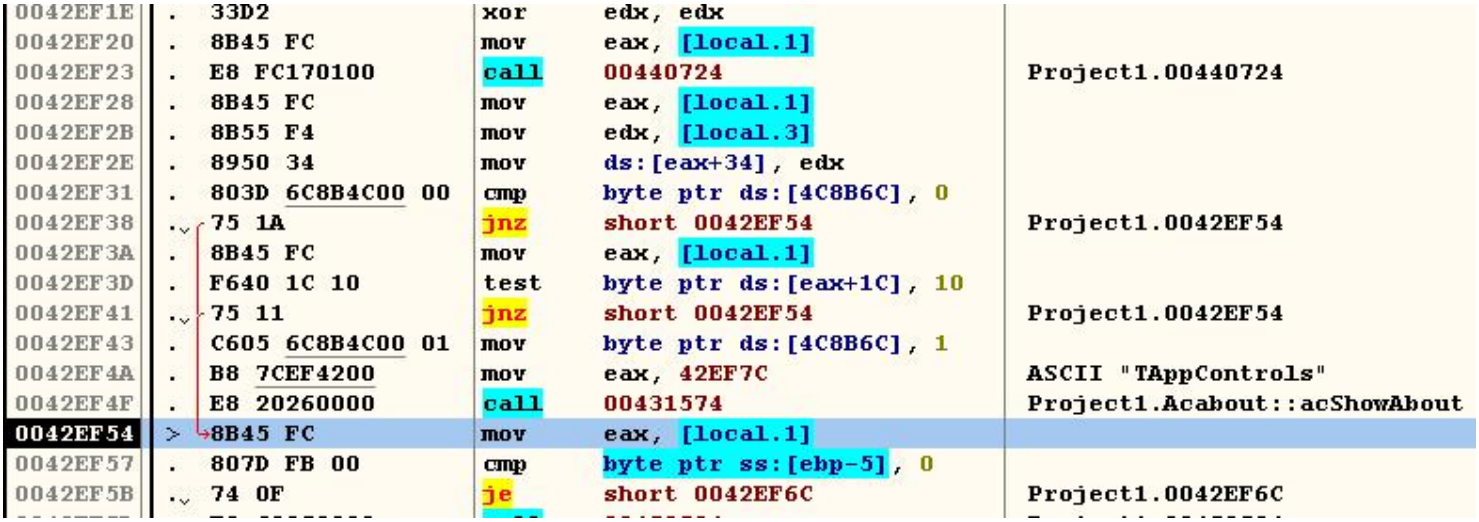
Lo instalo en C++Builder, creo un nuevo proyecto y le añado algún componente de este pack para ver que hace, lo compilo y ejecuto y veo esto:

Si doy en los puntos suspensivos de la propiedad “About” del componente insertado veo esto:

Las limitaciones son las mismas que en el pack que vi anteriormente así que analizo en Olly el binario creado por C++Builder.
Doy a F9 para iniciar la depuración y, una vez salga la NAG, doy a F12 para pausar. Voy con F7 y Ctrl + F9 hasta que la NAG responde y me deja pulsar el botón “Aceptar” y le doy a dicho botón. Una vez le haya dado, voy a Olly y continúo con Ctrl + F9 y F7 observando adónde voy retornando para ver si veo algo interesante hasta que llego aquí:



Si impido que se ejecute ese Call ya no saldría ninguna NAG pero lo extraño es que no hay ningún salto que la evite así que sigo con Ctrl + F9 y F7 y llego hasta aquí:



Veo que salgo debajo de un Call muy sospechoso también y esta vez sí que tengo 2 saltos que lo evitan.
Si parcheo uno de esos saltos no mostraría la NAG pero tiene un gran fallo, si lo hiciéra en el pack de componentes, en este hay un componente que se llama acQuickAboutBox y a esa zona llega siempre que crea un componente de ese tipo y, como para la NAG usa ese componente, pues solucionaríamos eso pero también perderíamos el componente acQuickAboutBox y, además, tampoco mostraría los Abouts de los componentes del pack ya que también lo usa para eso. ¿Cómo lo sé? Pues lo sé porque ya lo probé jejeje. Hay que buscar otra solución.
Pongo un BP en el Call de arriba, reinicio Olly y cuando pare entro con F7 y voy traceando hasta que llego aquí:

```
00431649  .  E8 DA760200      call    00458D28                 Project1.00458D28
0043164E  .  8B45 F8         mov     eax, [local.2]
00431651  .  C640 68 01      mov     byte ptr ds:[eax+68], 1
00431655  .  8B45 FC         mov     eax, [local.1]
00431658  .  BA 94174300     mov     edx, 431794              ASCII "AppControls"
0043165D  .  E8 6E7A0200     call    004590D0                 Project1.004590D0
00431662  .v 75 19           jnz     short 0043167D           Project1.0043167D
00431664  .  8B45 F8         mov     eax, [local.2]
00431667  .  83C0 40         add     eax, 40
0043166A  .  BA A8174300     mov     edx, 4317A8              ASCII "Software built with Advanced
0043166F  .  E8 B4760200     call    00458D28                 Project1.00458D28
00431674  .  A1 A4D84C00     mov     eax, ds:[4CD8A4]
00431679  .  FF00            inc     dword ptr ds:[eax]
0043167B  .v EB 10           jmp     short 0043168D           Project1.0043168D
0043167D  >  8B45 F8         mov     eax, [local.2]
00431680  .  83C0 40         add     eax, 40
00431683  .  BA DC174300     mov     edx, 4317DC              ASCII "Part of Advanced Application
00431688  .  E8 9B760200     call    00458D28                 Project1.00458D28
0043168D  >  E8 465B0900     call    004C71D8                 GetDesktopWindow
```

¿Qué tiene de curioso ese lugar? Pues que si no se produce el salto coge el texto mostrado en la NAG pero si se produce coge el texto mostrado en los Abouts de los componentes. Además vemos que arriba coloca en EAX un puntero a una variable que contiene la cadena “AppControls” y en EDX coloca otro puntero a otra cadena que también es “AppControls”.
Si entramos en el Call veremos cómo compara byte a byte las cadenas así que si son iguales no saltará. Apuntemos esa zona.
Abro el C++Builder en Olly, creo un nuevo proyecto, añado un componente de tipo acQuickAboutBox, doy a la propiedad “About” del componente y me sale la NAG. Pauso en Olly con F12. Continúo con F7 y Ctrl + F9 hasta que llego aquí:

```
0605AEFF  .  8B mov     eax, [local.2]
0605AF02  .  C6 mov     byte ptr ds:[eax+38], 0
0605AF06  .  8B mov     eax, [local.2]
0605AF09  .  C6 mov     byte ptr ds:[eax+39], 0
0605AF0D  .  8B mov     eax, [local.2]
0605AF10  .  E8 call    0600A15C                 AppContr.Acquickaboutbox::TacQuickAboutBox::Execute
0605AF15  .  33 xor     eax, eax
0605AF17  .  5A pop     edx
```

Si seguimos con Ctrl+F9 y F7 llegamos aquí:

```
060586EE  .  8950 34         mov     ds:[eax+34], edx
060586F1  .  803D 50500706 00 cmp     byte ptr ds:[6075050], 0
060586F8  .v 75 1A           jnz     short 06058714           AppContr.06058714
060586FA  .  8B45 FC         mov     eax, [local.Self]
060586FD  .  F640 1C 10      test    byte ptr ds:[eax+1C], 10
06058701  .v 75 11           jnz     short 06058714           AppContr.06058714
06058703  .  C605 50500706 01 mov    byte ptr ds:[6075050], 1
0605870A  .  B8 3C870506     mov     eax, 605873C             ASCII "TAppControls"
0605870F  .  E8 34260000     call    0605AD48                 AppContr.Acabout::acShowAbout
06058714  >  8B45 FC         mov     eax, [local.Self]
06058717  .  807D FB 00      cmp     byte ptr ss:[ebp-5], 0
0605871B  .v 74 0F           je      short 0605872C           AppContr.0605872C
```

Pongo un BP en ese Call, doy a F9 y vuelvo a dar al About del componente y esta vez entro con F7, voy traceando y llego aquí:

```
0605AE25  .  C640 68 01      mov   byte ptr ds:[eax+68], 1
0605AE29  .  8B45 FC         mov   eax, [local.1]
0605AE2C  .  BA 68AF0506     mov   edx, 605AF68          ASCII "AppControls"
0605AE31  .  E8 F6540100     call  0607032C              <jmp.&rtl60.System::LStrCmp>
0605AE36  .v 75 19           jnz   short 0605AE51        AppContr.0605AE51
0605AE38  .  8B45 F8         mov   eax, [local.2]
0605AE3B  .  83C0 40         add   eax, 40
0605AE3E  .  BA 7CAF0506     mov   edx, 605AF7C          ASCII "Software built with Ad
0605AE43  .  E8 32550100     call  0607037A              <jmp.&rtl60.System::LStrAsg>
0605AE48  .  A1 4C630706     mov   eax, ds:[607634C]
0605AE4D  .  FF00            inc   dword ptr ds:[eax]
0605AE4F  .v EB 10           jmp   short 0605AE61        AppContr.0605AE61
0605AE51  >  8B45 F8         mov   eax, [local.2]
0605AE54  .  83C0 40         add   eax, 40
0605AE57  .  BA B0AF0506     mov   edx, 605AFB0          ASCII "Part of Advanced Appli
0605AE5C  .  E8 19550100     call  0607037A              <jmp.&rtl60.System::LStrAsg>
0605AE61  >  E8 88800100     call  06072EEE              [GetDesktopWindow
```

Ya estoy donde quería. Ahora, si miro que hay en EAX y EDX vemos esto:

```
Registers (FPU)
EAX 07127974 ASCII "acQuickAboutBox"
ECX 00000000
EDX 0605AF68 ASCII "AppControls"
EBX 040616D4
ESP 0012F300
EBP 0012FB2C
ESI 01290FD8
EDI 0012FEA0
```

O sea que compara el nombre de la clase del componente con la cadena “AppControls”. Si sigo traceando veo que salta en el condicional y nos muestra el About del componente pero no la NAG.
Pongo un BP en ese Call, reinicio Olly, doy a F9, cuando arranca C++Builder, vuelvo a crear un nuevo proyecto, añado un componente acQuickAboutBox, doy a su propiedad “About” y miro lo que tengo en los registros:

```
Registers (FPU)
EAX 07111B88 ASCII "AppControls"
ECX 00000000
EDX 0605AF68 ASCII "AppControls"
EBX 0409E1F4
ESP 0012EAA0
EBP 0012F2CC
ESI 01290FD8
EDI 0012FEA0
```

Veo que esta vez EAX apunta a “AppControls” en vez de apuntar al nombre de la clase del componente.
¿Qué pasaría si después del salto condicional de la línea 0x0605AE36 colocase un salto incondicional que apunte justo debajo de la llamada al método Execute del acQuickAboutBox?

```
0605AE25  .  C640 68  mov   byte ptr ds:[eax+68]
0605AE29  .  8B45 FC  mov   eax, [local.1]
0605AE2C  .  BA 68AF0 mov   edx, 605AF68          ASCII "AppControls"
0605AE31  .  E8 F6540 call  0607032C              <jmp.&rtl60.System::LStrCmp>
0605AE36  .v 75 19    jnz   short 0605AE51        AppContr.0605AE51
0605AE38  .v E9 D8000 jmp   0605AF15              AppContr.0605AF15
0605AE3D     90       nop
0605AE3E  .  BA 7CAF0 mov   edx, 605AF7C          ASCII "Software built with Advanced App
0605AE43  .  E8 32550 call  0607037A              <jmp.&rtl60.System::LStrAsg>
0605AE48  .  A1 4C630 mov   eax, ds:[607634C]
0605AE4D  .  FF00     inc   dword ptr ds:[eax]
```

```
0605AF09  .  C640 39   mov     byte ptr ds:[eax+39]
0605AF0D  .  8B45 F8   mov     eax, [local.2]
0605AF10  .  E8 47F2F  call    0600A15C                AppContr.Acquickaboutbox::TacQuickAboutBox::Execute
0605AF15  .  33C0      xor     eax, eax
0605AF17  .  5A        pop     edx
0605AF18  .  59        pop     ecx
0605AF19  .  59        pop     ecx
0605AF1A  .  64:8910   mov     fs:[eax], edx
0605AF1D  .  68 32AF0  push    605AF32
0605AF22  >  8B45 F8   mov     eax, [local.2]
0605AF25  .  E8 0A550  call    06070434                <jmp.&rtl60.System::TObject::Free>
0605AF2A  L. C3        retn
```

Pues que no muestra la NAG pero el About no está redimensionado correctamente:

¿Por qué pasa esto?
Si reinicio y repito todo pero sin colocar el salto incondicional y traceo veo esto:

```
0605AE3E  .  BA 7CAF0  mov     edx, 605AF7C            ASCII "Software built with Advanced Ap
0605AE43  .  E8 32550  call    0607037A                <jmp.&rtl60.System::LStrAsg>
0605AE48  .  A1 4C630  mov     eax, ds:[607634C]
0605AE4D  .  FF00      inc     dword ptr ds:[eax]
0605AE4F  .v EB 10     jmp     short 0605AE61          AppContr.0605AE61
0605AE51  >  8B45 F8   mov     eax, [local.2]
```

¿Adónde apunta EAX?

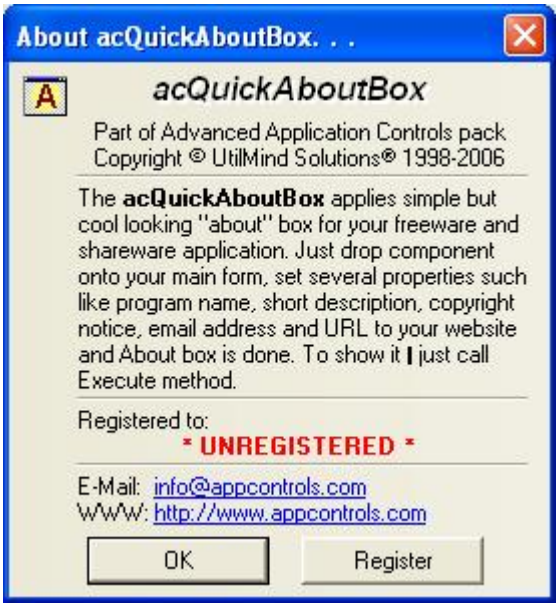
```
ds:[0607504C]=0000052D

:170.
```

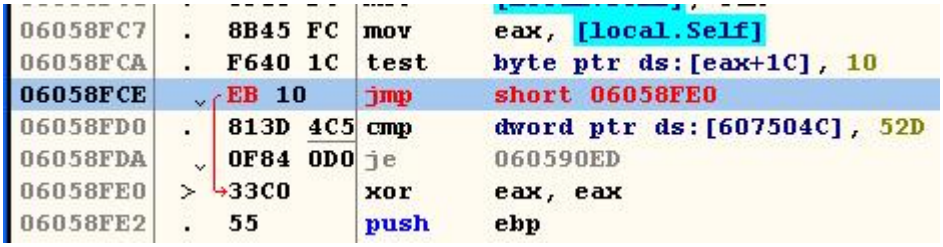
Pongo un HBP on Access de un byte en esa dirección, doy a F9 y para aquí:

```
06058FC1  .  8955 F8   mov     ss:[ebp-8], edx
06058FC4  .  8945 FC   mov     [local.Self], eax
06058FC7  .  8B45 FC   mov     eax, [local.Self]
06058FCA  .  F640 1C   test    byte ptr ds:[eax+1C], 10
06058FCE  .v 75 10     jnz     short 06058FE0
06058FD0  .  813D 4C5  cmp     dword ptr ds:[607504C], 52D
06058FDA  .v 0F84 0D0  je      060590ED
06058FE0  >  33C0      xor     eax, eax
06058FE2  .  55        push    ebp
```

Vuelvo a reiniciar y esta vez coloco el salto incondicional, nopeo el salto que se ve arriba resaltado y quito el HBP ya que para continuamente y al mostrar el About:



En vez de nopear ese salto condicional, se podría también convertir el primer salto condicional que está encima de la comparación en un salto incondicional con lo que el resultado sería lo mismo que si nopeo el segundo:

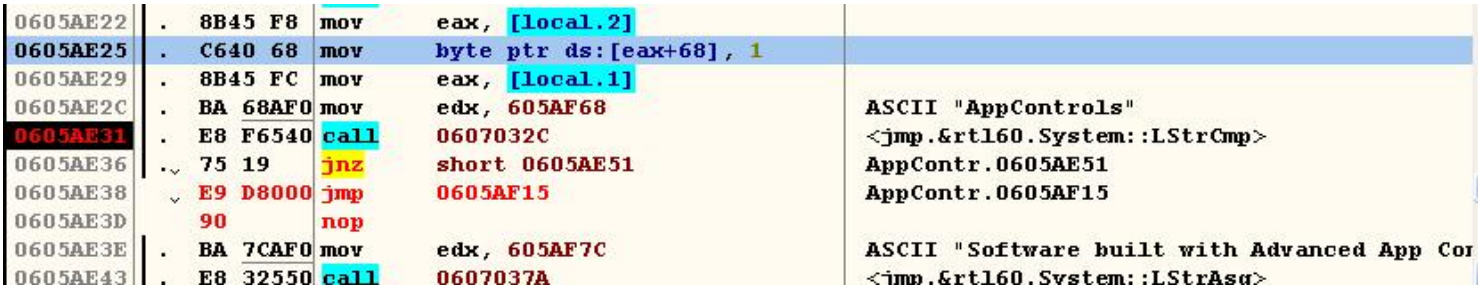


Ya no muestra la NAG y muestra el About de forma correcta. O sea, 0x52D es un identificador para saber si ya se mostró la NAG ya que si se entra en la zona de la NAG se incrementa con lo que quedaría como 0x52E. Al injertar el salto antes de que incremente ese identificador da este problema pero si cambio el salto de la imagen anterior ya siempre redimensionará la ventana.
Ahora queda algo más, la palabra “* UNREGISTERED *” queda feísima ahí así que habrá que hacer algo para solucionarlo.
Si hago doble clic sobre el componente acQuickAboutBox que añadí al proyecto me muestra esto:

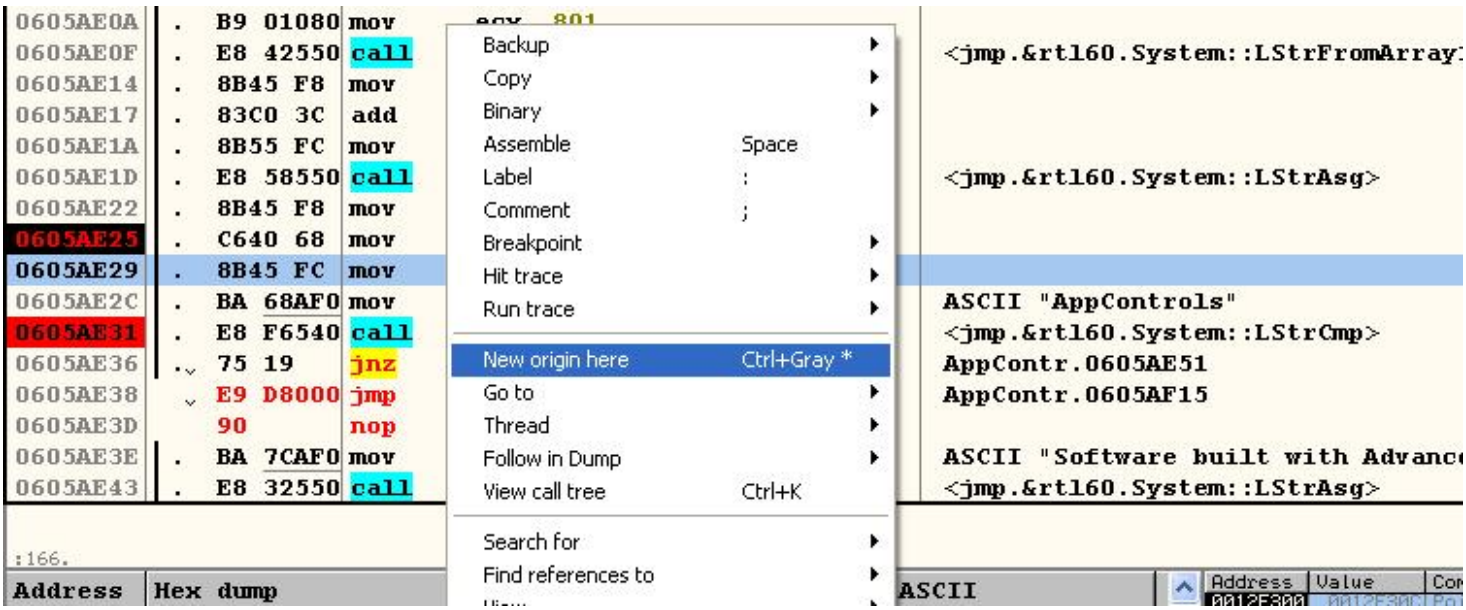
Si miro entre sus propiedades veo esto:

Yo no he cambiado absolutamente nada así que las propiedades están tal cual se crea el componente. Tanto la propiedad “Registered” como “Shareware” tienen asignadas el valor false por defecto. Si le asigno a “Shareware” el valor true veo esto:

Esto quiere decir que el componente pone la propiedad “Shareware” a true después de su creación pero antes de mostrar la ventana. Si vuelvo a dar a la propiedad “About” y miro un poco más arriba del BP veo esto:



Pongo un BP en esa línea y cuando pare doy clic derecho en la siguiente línea y elijo “New origin here”:



Doy a F9 2 veces para que continúe la ejecución y veo esto:

O sea que esa línea hay que nopearla o bien hacer que meta 0 en lugar de 1 y con eso ya no mostrará el “* UNREGISTERED *”.


0605AE1A . 8B55 FC mov edx, [local.1]
0605AE1D . E8 58550 call 0607037A <jmp.&rtl60.System::LStrAsg>
0605AE22 . 8B45 F8 mov eax, [local.2]
0605AE25 C640 68 mov byte ptr ds:[eax+68], 0
0605AE29 . 8B45 FC mov eax, [local.1]
0605AE2C . BA 68AF0 mov edx, 605AF68 ASCII "AppControls"
0605AE31 . E8 F6540 call 0607032C <jmp.&rtl60.System::LStrCmp>
0605AE36 . 75 19 jnz short 0605AE51 AppContr.0605AE51
0605AE38 E9 D8000 jmp 0605AF15 AppContr.0605AF15
0605AE3D 90 nop
0605AE3E . BA 7CAF0 mov edx, 605AF7C ASCII "Software built with Advanced A
0605AE43 . E8 32550 call 0607037A <jmp.&rtl60.System::LStrAsg>
0605AE48 . A1 4C630 mov eax, ds:[607634C]
0605AE4D . FF00 inc dword ptr ds:[eax]
0605AE4F . EB 10 jmp short 0605AE61 AppContr.0605AE61
0605AE51 > 8B45 F8 mov eax, [local.2]
0605AE54 . 83C0 40 add eax, 40
0605AE57 . BA B0AF0 mov edx, 605AFB0 ASCII "Part of Advanced Application (
0605AE5C . E8 19550 call 0607037A <jmp.&rtl60.System::LStrAsg>
0605AE61 > E8 88800 call 06072EEE GetDesktopWindow

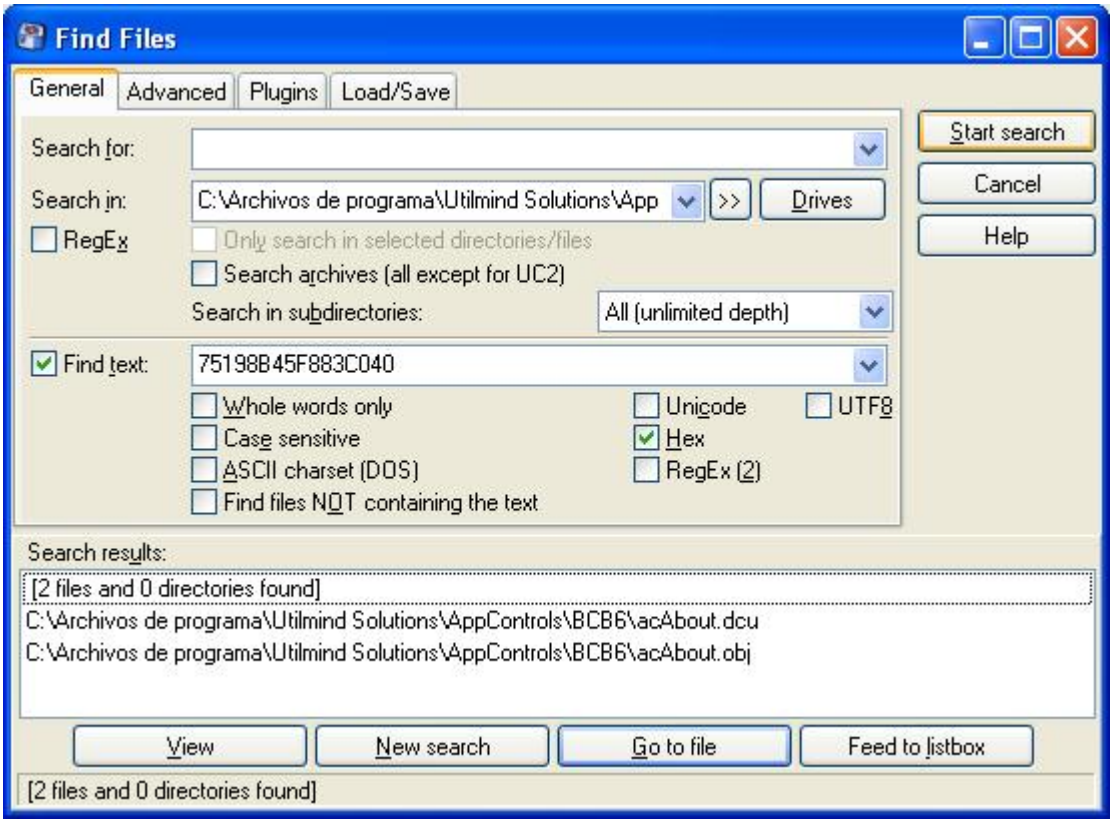
3. Advanced Application Controls – Parcheando el componente

Ahora lo que debo hacer es obtener las firmas binarias a buscar en los archivos del componente, buscarlas con TCUP en el directorio del paquete y modificarlos con WinHex. Yo usé estas:

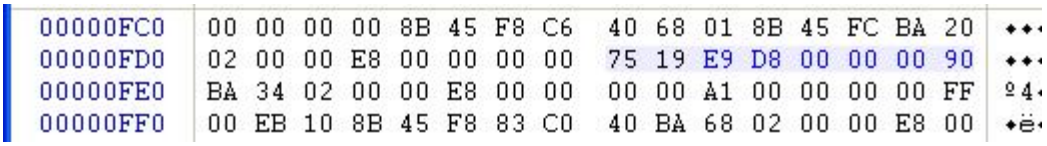
Para el primero busco esto:

```
75 19 8B 45 F8 83 C0 40
```

Y lo encuentra en:



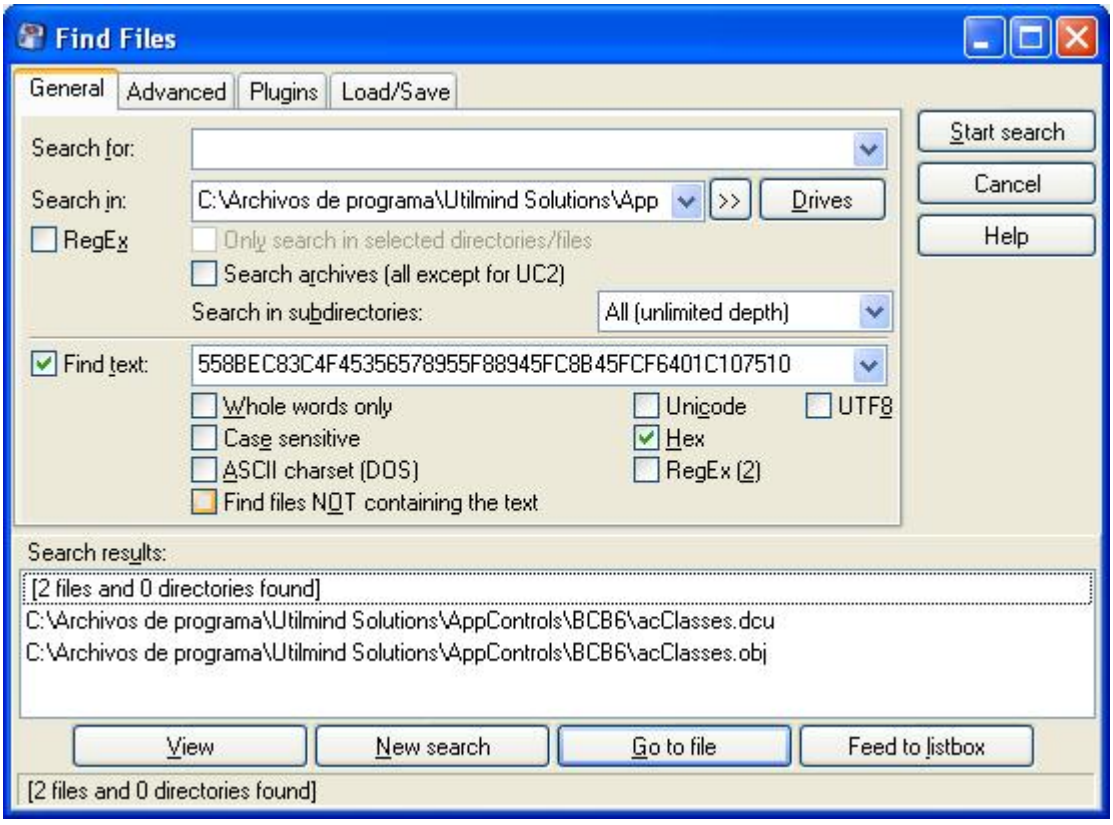
Y queda modificado así:



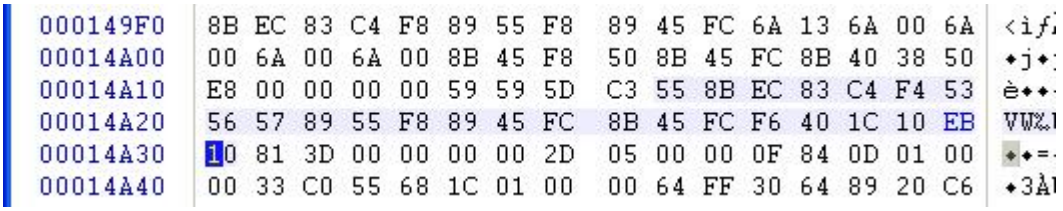
Para el segundo busco esto:

```
55 8B EC 83 C4 F4 53 56 57 89 55 F8 89 45 FC 8B 45 FC F6 40 1C 10 75 10
```

Y lo encuentra en:



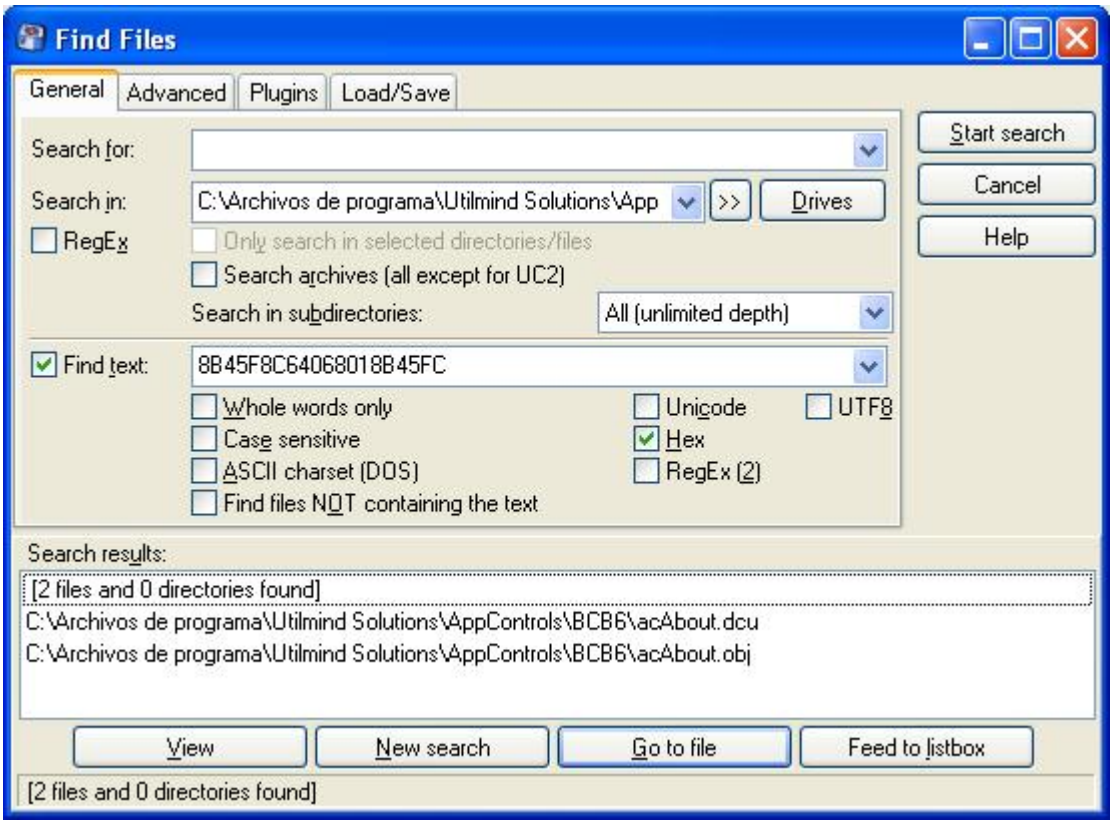
Y queda modificado así:



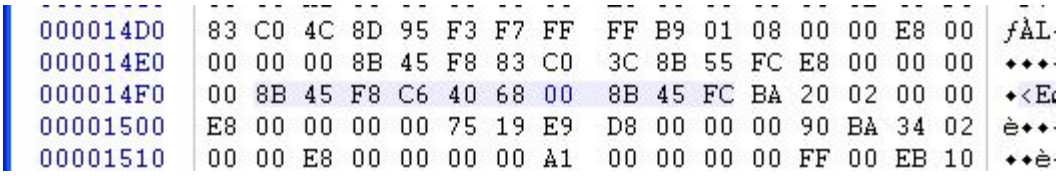
Para el tercero busco esto:

```
8B 45 F8 C6 40 68 01 8B 45 FC
```

Y lo encuentra en:



Y queda modificado así:



Desinstalo el paquete de componentes tal y como lo hice para el paquete anterior, voy al directorio donde se creó el archivo .lib. En mi caso se creó en la siguiente ruta:
C:\Archivos de programa\Borland\CBuilder6\Projects\Lib

Y veo que hay dos archivos (AppControlsCB6.bpi y AppControlsCB6.lib) los cuales elimino.
Luego voy al directorio donde se creó el .bpl. En mi caso se creó en la siguiente ruta:
C:\Archivos de programa\Borland\CBuilder6\Projects\Bpl

Y veo que también hay dos archivos (AppControlsCB6.bpl y AppControlsCB6.tds) los cuales elimino también.

Una vez hecho esto vuelvo a instalar el paquete de componentes y ya no me vuelve a salir nunca más la NAG y no aparece el About “* UNREGISTERED *”.

Decir que ambos paquetes de componentes no hay manera de registrarlos ni con un .reg, ni con formulario de registro, ni con nada por el estilo (o al menos yo no vi la manera). La única solución es el parcheo y ahí están, recién salidos del horno.
Espero que os haya gustado y que, sobre todo, sirva de ayuda para aquellos que quieran hacer algo similar ya que parchear un componente para Builder o Delphi es así para prácticamente todos los casos.

Agradecimientos

Gracias a todos los **CracksLatinoS** porque sin su ayuda no sería capaz de hacer nada de lo que ahora soy capaz y en especial para este proyecto a mi amigo Neutrino que me ayudó en un momento en que me quedé atascado y no sabía cómo seguir y que además creó el pack de **Olly portable All In One 1.10** que uso para mis proyectos y que es tan completo que, normalmente, no necesito nada más para ello.