Estudio completo de aplicaciones Visual C++ 6.0 (3 parte)

por

AkirA

Información

Programa:	VCCrackMe10
Tamaño:	El tamaño no importa XDDDDD
Herramientas:	1. Ollydbg 1.09c
Dificultad:	**NewBie avanzado**

Introducción

Hola amigos!!!! Bienvenidos a la 43 entrega del curso de AkirA sobre protecciones comerciales.

Hola, en el tuto anterior dije que haría una tercera parte explicando mejor como saber para que sirven las funciones del MFC que utilizan los crackmes de Visual C++ hechos con MFC.
La idea original no era hacerlo con este crackme, pero cuando aplique la técnica sobre el me pareció lo suficientemente bueno para utilizarlo y explicarlo lo mejor posible

Espero que lo aprovechéis XD

Cualquier duda escribirme a mi email atalasa@hotmail.com

Nota : si necesitáis desinformación sobre Ollydbg buscar en la pagina de Joe Cracker : www.iespana.es/ollydbg esta es la mejor pagina que hay sobre el tema, de hecho gracias ha ella yo hago todos es tos proyectos en olly

Comentario del Programa

Por supuesto el disclaimer de turno. Vamos a ver, no es que no me haga responsable de la utilización de esta información, es que directamente paso del tema, el único propósito de todo esto es de carácter educativo.
A fin de cuentas, si lo que quieres es crackear un programa pues te bajas el crack y punto, pero si vas a leer este tutorial es porque tu objetivo es aprender. Eso es lo que nos motiva, el comprender como funcionan las cosas o como están hechas por dentro, y por supuesto el subidón de haberle ganado a un equipo de ingenieros diseccionando un objeto que ellos habían diseñado y del cual no sabemos nada.

Manos a la Obra
Comienza el asedio....

Hola y bienvenidos a la 43 entrega del curso de AkirA sobre protecciones comerciales.

Vamos a ver como saber para que sirve cada llamada a cada función de las MFC con un caso practico que me ha gustado, porque pienso que es muy muy claro.

Para este ejemplo vamos a utilizar el VCCrackMe10.exe del tuto anterior.

En el tuto anterior pudimos encontrar el punto en el que comenzaba la función que gestiona el evento de "se ha presionado el botón Check".

Pero luego nos encontramos con que no utiliza apis, entonces es muy difícil saber lo que hace porque simplemente llama a funciones de la MFC42.dll y solo vemos CALL #438 , por ejemplo o cosas así.

Bien, haced esto. Cargar el IDA y cargar abrir el con el , el VCCrackMe10.

Marcad Load Recurses:

Bien , dad al botón de Ok y esperad a que el botón verde de arriba a la derecha este en verde:

Una vez aquí, pinchad en File – Produce –Create MAP file y poned de nombre al archivo Crackme.map

En el cuadro de dialogo marcar todas las opciones.

Y cerráis el IDA.

Ahora cargáis el Olly y lógicamente el crackme también.(sabemos del tuto anterior que la función que comprueba el serial empieza en 401526

Debereis tener un plugin llamado MAPCONV.

Dais a plugins – MapConv – replace label y pincháis en el crackme.map:
Mirad XDDD ahora todas esas llamadas a tipo CALL [MFC42.DLL#389] ahora tienen un significado.
Esos son los nombres de las funciones de verdad. Claro que para sacarle el máximo partido a esto no estaría demás que miraseis manuales de programación para windows con MFC.

Podéis ver que ahora con el nombre de las funciones todo es mucho mas fácil.

Vamos a verlo paso a paso

Primero, vemos como crea varios Objetos Cstring (que no son mas que "variables" que contienen una cadena de texto)

```
00401543 . 68 58304000   PUSH VCCrackM.00403058                    ASCII "Wrong! Try Again"
00401548 . E8 79030000   CALL <VCCrackM.CString::CString(char const  JMP to MFC40.#483
0040154D . 68 50304000   PUSH VCCrackM.00403050                    ASCII "Error"
00401552 . 8D4D EC       LEA ECX,DWORD PTR SS:[EBP-14]
00401555 . C745 FC 00000( MOV DWORD PTR SS:[EBP-4],0
0040155C . E8 65030000   CALL <VCCrackM.CString::CString(char const  JMP to MFC40.#483
00401561 . 68 44304000   PUSH VCCrackM.00403044                    ASCII "Correct..."
00401566 . 8D4D E8       LEA ECX,DWORD PTR SS:[EBP-18]
00401569 . C645 FC 01    MOV BYTE PTR SS:[EBP-4],1
0040156D . E8 54030000   CALL <VCCrackM.CString::CString(char const  JMP to MFC40.#483
00401572 . 68 38304000   PUSH VCCrackM.00403038                    ASCII "Good Work!!"
00401577 . 8D4D DC       LEA ECX,DWORD PTR SS:[EBP-24]
0040157A . C645 FC 02    MOV BYTE PTR SS:[EBP-4],2
0040157E . E8 43030000   CALL <VCCrackM.CString::CString(char const  JMP to MFC40.#483
00401583 . 68 24304000   PUSH VCCrackM.00403024                    ASCII "Inform me via Email"
00401588 . 8D4D D8       LEA ECX,DWORD PTR SS:[EBP-28]
0040158B . C645 FC 03    MOV BYTE PTR SS:[EBP-4],3
0040158F . E8 32030000   CALL <VCCrackM.CString::CString(char const  JMP to MFC40.#483
```

Luego vemos mas abajo como como mete en la pila Push 3E8 y luego llama Cwnd::GetDlgItemTextA

```
004015A2 . 68 E8030000   PUSH 3E8
004015A7 . E8 14030000   CALL <VCCrackM.CWnd::GetDlgItemTextA(int,CS  JMP to MFC40.#2712
004015AC . 8D4D EC       LEA ECX,DWORD PTR SS:[EBP-14]
```

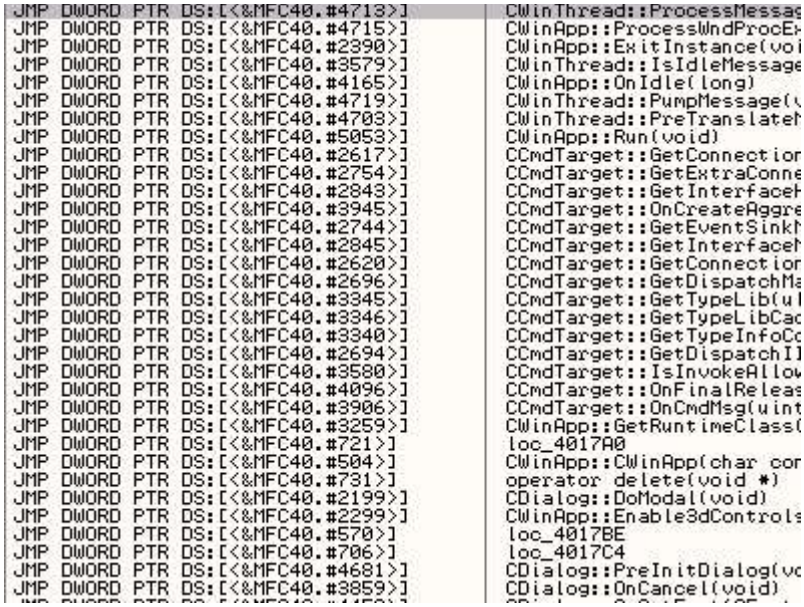
CWnd no es mas que un objeto ventana, que posee una función GetDlgItemTextA (os podéis imaginar que esta función es la que se encarga de coger nuestro serial, y que 3E8 sera el ID de la caja de texto)

Y luego vemos que también utiliza la función Cstring::Operator(+) (y que pone Push eax y Push edx, si hacemos follow in dump eax y edx veremos las cadenas Correct... y Serial . Ok conclusión, esta creando una cadena que sera la sumas de esas dos cadenas , es casi intuitivo)

Y mas abajo vemos que con la función _mbsStrc compara la cadena que acaba de formar con nuestro serial, bueno ya sabéis cual es el serial XDD

Bueno, pues después de estos tres tutos espero haber explicado bien como funciona Visual C++ y que hay que hacer para analizarlo.

Pero las posibilidades de saber las funciones no se limitan a los seriales, ya se sabe para que funciona cualquier trozo del programa, echad un vistazo a esto XD



Hombre, si no sabes programar con las MFC, quizás, esto no te diga mucha, ya sabes XD aprende las MFC que lo vas a agradecer XD

Por curiosidad, abrid el crackme.map con el notepad y mirad lo que hay

```
0001:000006CE       CWinApp::OnHelp(void)
0001:000006D4       CWinApp::WinHelpA(ulong,uint)
0001:000006DA       CWinApp::OnDDECommand(char *)
0001:000006E0       CWinApp::DoWaitCursor(int)
0001:000006E6       CWinApp::DoMessageBox(char const *,uint,uint)
0001:000006EC       CWinApp::SaveAllModified(void)
0001:000006F2       CWinApp::InitApplication(void)
0001:000006F8       ??ddToRecentFileList@CWinApp@@UAEXPBD@Z
0001:000006FE       CWinApp::OpenDocumentFile(char const *)
0001:00000704       CWinThread::Delete(void)
0001:0000070A       CWinThread::GetMainWnd(void)
0001:00000710       CWinThread::ProcessMessageFilter(int,tagMSG *)
```

Bueno, espero que os hayais divertido y que hayais aprendido mucho. Hasta pronto!!!

Nota:

Bueno espero que te hayas divertido y sobre todo que hayas aprendido mucho que es de lo que se trata , que te sirva de ejemplo para que tu tambien puedas hacerlo, desde luego no hay comparado como coger un programa, abrirlo, ver un monton de codigo por todas partes y manejar lo que otros ingenieros han hecho ,tu solo, por ti mismo.

Bueno , si quieres comentarme algo escribeme a atalasa@hotmail.com

Chao!!

Espero que hayan disfrutado leyendo este tutorial y que les sirva para incrementar sus habilidades, pero recuerden, lean muchos tutoriales, practiquen, estudien y CRACKEAR será mucho más