AQ Video Converter Platinum 1.00.118	
Fecha	**1 de mayo de 2009**
URL de descarga	**http://www.aqripper.com/download/AQ_Video_Converter_Platinum.exe**
MD5 del instalador	**f39b70284fb47086af8c3b75a199aa22**
Protección	**Serial**
Herramientas	**OllyDbg 1.10, C++Builder 6 con componente TDbgCLS**
Objetivo	**Quitar las limitaciones**
Dificultad	**Media**
Cracker	**Aguml**

INDICE

DESCRIPCION DEL PROGRAMA

"Edita y convierte vídeos para cualquier dispositivo o formato"

La mayoría de conversores de vídeo son poco intuitivos y carecen de funciones de edición. Para algo tan simple como un recorte hace falta usar un segundo programa.

AQ Video Converter Platinum realiza la conversión a partir de una gran selección de perfiles predefinidos, organizados según el dispositivo o típo de vídeo. Así pues, además de convertir a un fichero estándar, puedes optimizar el vídeo final para un iPod, un móvil o una consola.

Con AQ Video Converter Platinum puedes editar los vídeos antes de convertirlos. Para ello deberás añadirlos primero a una tabla de ficheros. Tras elegir uno y pulsar Edit, AQ Video Converter Platinum mostrará sus propiedades y varias pestañas de edición.

En Trim puedes recortar el vídeo eligiendo un punto de inicio y otro final, mientras que en Norm hay selectores de proporción y ajuste de volumen. Crop & Pad posibilita insertar márgenes o recortar la imagen. Cuando hayas terminado, el botón Start hará que AQ Video Converter Platinum inicie el proceso de conversión, bastante rápido y controlable mediante prácticos botones.

AQ Video Converter Platinum es un conversor potente, de aspecto agradable y con útiles opciones de edición. Si bien no superan la calidad de utilidades profesionales, son más que suficientes para el día a día.

AQ Video Converter Platinum soporta los siguientes formatos:

Vídeo: AVI, MP4, DivX, XviD, MPEG, WMV, MOV, ASF, 3GP, MOD, FLV, HD
Sonido: MP3, WAV,AC3, AAC, M4A, OGG
Dispositivos: iPod, iPhone, Zune, PSP, Móviles, Apple TV, Zen, PS3, Xbox, Archos, etcétera

Limitaciones de la versión de prueba:

- Cada vídeo muestra una marca de agua con la URL del programa
- Sólo se ejecuta un proceso de conversión
- Duración limitada a tres minutos

Para utilizar AQ Video Converter Platinum necesitas:

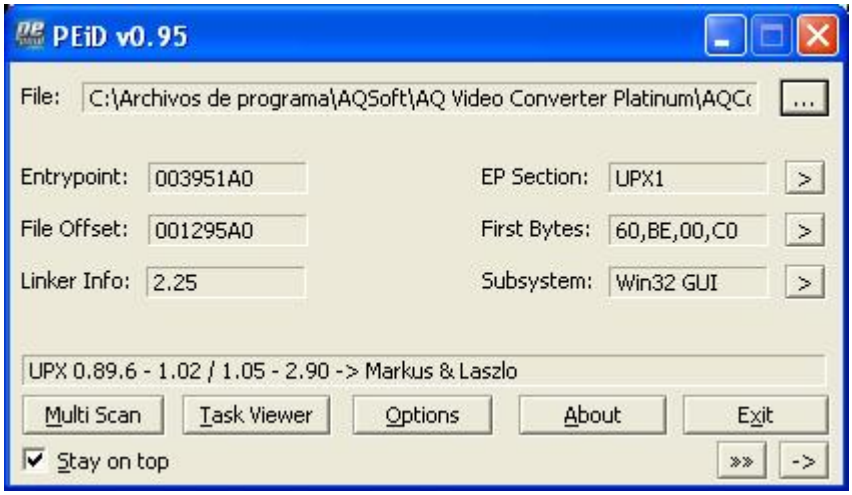
- Sistema operativo: Win2000/XP/2003/Vista

Requisitos mínimos:

- Procesador: 800 MHz
- Memoria: 512 MB
- Vídeo: 32 MB
- Espacio libre en disco: 20 MB
- Resolución de pantalla: 800x600

ESTUDIANDO AL ENEMIGO

Bueno, ellos mismos nos dicen ya las limitaciones que trae así que lo pasaré por algun analizador:



Pues parece que nos toco una protección sencilla así que creo que es la victima ideal para probar lo que he aprendido sobre Loaders-Debuggers. Abramoslo en Olly:

Address	Hex dump	Disassembly
007951A0	60	pushad
007951A1	BE 00C06600	mov esi, 66C000
007951A6	8DBE 0050D9FF	lea edi, ds:[esi+FFD95000]
007951AC	57	push edi
007951AD	83CD FF	or ebp, FFFFFFFF
007951B0	EB 10	jmp short 007951C2
007951B2	90	nop
007951B3	90	nop
007951B4	90	nop
007951B5	90	nop
007951B6	90	nop
007951B7	90	nop
007951B8	8A06	mov al, ds:[esi]

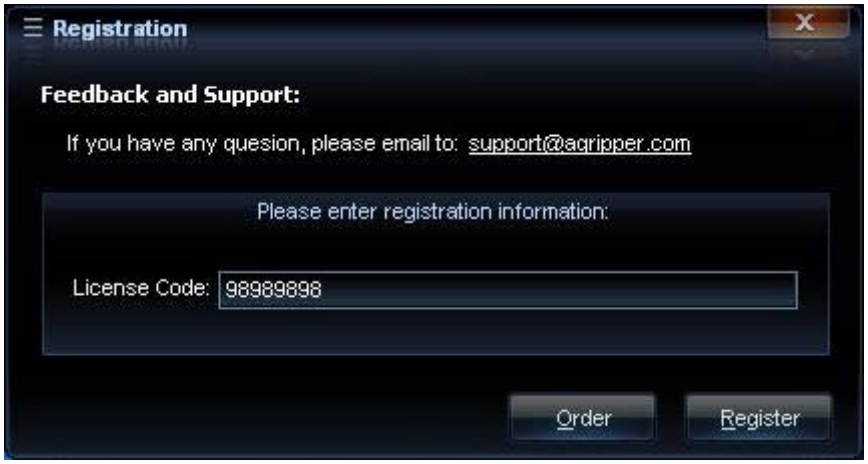
Provamos con el método del PUSHAD-POPAD y llegamos aquí:

Address	Hex dump	Disassembly
00795308	FF96 70DB3900	call ds:[esi+39DB70]
0079530E	61	popad
0079530F	E9 B0C6E5FF	jmp 005F19C4
00795314	2C 53	sub al, 53
00795316	79 00	jns short 00795318
00795318	50	push eax
00795319	53	push ebx
0079531A	79 00	jns short 0079531C
0079531C	C420	les esp, ds:[eax]
0079531E	5F	pop edi
0079531F	0000	add ds:[eax], al
00795321	0000	add ds:[eax], al
00795323	0000	add ds:[eax], al

Ponemos un BP en ese JMP que nos lleva al OEP para no tener que repetir el método del PUSHAD-POPAD y damos a F7 y ya estamos en el OEP:

Address	Hex dump	Disassembly		Comment
005F19C4	55	push	ebp	
005F19C5	8BEC	mov	ebp, esp	
005F19C7	83C4 F0	add	esp, -10	
005F19CA	53	push	ebx	
005F19CB	56	push	esi	
005F19CC	57	push	edi	
005F19CD	B8 94145F00	mov	eax, 5F1494	
005F19D2	E8 C958E1FF	call	004072A0	AQConver.004072A0
005F19D7	8B3D B0585F00	mov	edi, ds:[5F58B0]	AQConver.005F6BC8
005F19DD	8B07	mov	eax, ds:[edi]	
005F19DF	E8 2C94E9FF	call	0048AE10	AQConver.0048AE10
005F19E4	68 A81A5F00	push	5F1AA8	ASCII "ICOimage"
005F19E9	A1 68665F00	mov	eax, ds:[5F6668]	
005F19EE	50	push	eax	
005F19EF	E8 E465E1FF	call	00407FD8	jmp to USER32.LoadIconA

Pues ya tengo algunos datos necesarios por si queremos desempacarlo pero de momento voy a seguir a ver si soy capaz de dar con un serial bueno o puedo crear un keygen. Damos a F9 para que arranque el programa ye intento registrarme:



Al dar al botón Register no sale ninguna ventana de chico malo asi que lo tendremos mas difícil para dar con la zona que nos interesa. Lo primero que se me ocurrió fue mirar en las strings a ver si veía el Unregistered de la ventana principal:

```
005E72B7 mov     ecx, 5E7538    ASCII "AQ Video Converter Platinum"
005E72BC mov     edx, 5E755C    ASCII "The current version of DirectX installed on your computer is too low,"
005E734E mov     ecx, 5E75E4    ASCII "AQVCP.aqv"
005E738C mov     ecx, 5E7538    ASCII "AQ Video Converter Platinum"
005E7391 mov     edx, 5E75F8    ASCII "Open the default file error! you need to reinstall the software. "
005E7410 mov     edx, 5E7538    ASCII "AQ Video Converter Platinum"
005E7423 mov     edx, 5E7644    ASCII "AQ Video Converter Platinum (Unregistered)"
005E743C mov     edx, 5E7538    ASCII "AQ Video Converter Platinum"
005E74B2 mov     edx, 5E7678    ASCII "39EA968465F26B6CDCA1E51EC8FAC6392100A838813BD0E0F5575E31AC38AEE434E3AF
```

Pues parece que podríamos tener suerte asi que vayamos a esa dirección a ver que nos encontramos:

```
005E740E  .  8B00          mov     eax, ds:[eax]
005E7410  .  BA 38755E00   mov     edx, 5E7538                 ASCII "AQ Video Converter Platinum"
005E7415  .  E8 DE35EAFF   call    0048A9F8                    AQConver.0048A9F8
005E741A  .  833D 408A5F0( cmp     dword ptr ds:[5F8A40], 0
005E7421  .v 74 19         je      short 005E743C              AQConver.005E743C
005E7423  .  BA 44765E00   mov     edx, 5E7644                 ASCII "AQ Video Converter Platinum (Unregistered)"
005E7428  .  8B45 FC       mov     eax, ss:[ebp-4]
005E742B  .  E8 E830E8FF   call    0046A518                    AQConver.0046A518
005E7430  .  8B45 FC       mov     eax, ss:[ebp-4]
005E7433  .  C680 2805000( mov     byte ptr ds:[eax+528], 1
005E743A  .v EB 0D         jmp     short 005E7449              AQConver.005E7449
005E743C  >  BA 38755E00   mov     edx, 5E7538                 ASCII "AQ Video Converter Platinum"
005E7441  .  8B45 FC       mov     eax, ss:[ebp-4]
```

Pues tenemos algo muy interesante, si nos fijamos, vemos que esta la cadena con el (Unregistered) pero justo arriba tenemos un salto condicional que, si se cumple, metemos la cadena en EDX pero sin el (Unregistered), o sea, que es la comprobación que hace para mostrar la barra de titulo de la ventana principal como registrada o como no registrada. Justo encima del salto condicional tenemos una comparación entre una dirección fija y 0 y justo encima tenemos un CALL.
Pues bien, quitemos ese BP y pongamoslo en la comparación que está en 5E741A, vayamos a esa dirección fija en el dump y pongamos un HBP on Access y reiniciemos el programa a ver si para al arrancar. Para varias veces sin importancia aparente y luego para aquí:

```
005E6F28   E8 FFDDE1FF   call    00404D2C
005E6F2D   8BC6          mov     eax, esi
005E6F2F   E8 18CFE1FF   call    00403E4C
005E6F34   C705 408A5F00 mov     dword ptr ds:[5F8A40], -1
005E6F3E   33C0          xor     eax, eax
005E6F40   5A            pop     edx
005E6F41   59            pop     ecx
```

Podemos observar como metió 0xFFFFFFFF en la dirección que nos interesa. Pues bien, en el salto que vimos antes, al comparar el contenido de esa dirección con 0, en esa dirección había 0xFFFFFFFF. Hagamos una prueba, pongamos a 0 ese contenido y a ver donde vuelve a parar. Para ello vamos a la misma en el dump y la editamos y y damos a F9 para continuar.

```
005E73BE   8B45 FC       mov     eax, ss:[ebp-4]
005E73C1   E8 36960000   call    005F09FC                    AQConver.005F09FC
005E73C6   A1 448A5F00   mov     eax, ds:[5F8A44]
005E73CB   E8 C0F6FFFF   call    005E6A90                    Aqui comprueba nuestro archivo de registro
005E73D0   A3 408A5F00   mov     ds:[5F8A40], eax
005E73D5   8B45 FC       mov     eax, ss:[ebp-4]
005E73D8   33D2          xor     edx, edx
```

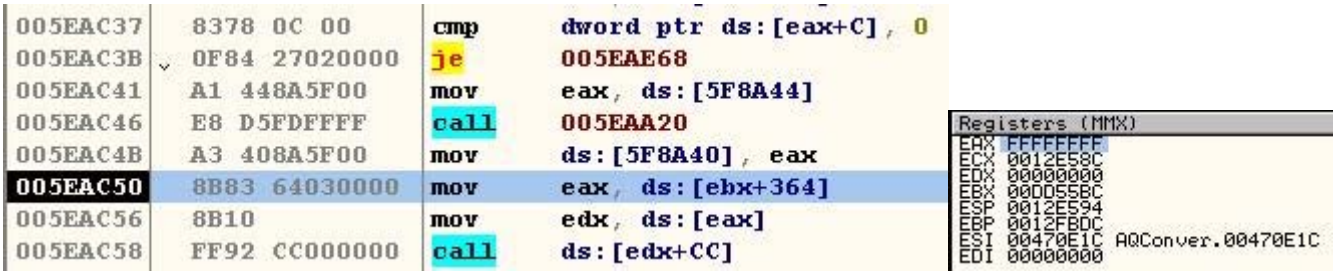
```
Registers (FPU)
EAX FFFFFFFF
ECX 0012FD98
EDX 0012FDE4
EBX 00DD55BC
ESP 0012FDE4
EBP 0012FE10
ESI 00DD16D8
EDI 005E4F2C AQConver.
```

Podemos observar que movió el valor de EAX a la dirección que nos interesa y que EAX vale 0xFFFFFFFF, mal asunto para nuestros intereses jejeje. Pongamos de nuevo a 0 el contenido de dicha dirección y demos a F9 para continuar. Ahora paró en la comparación para sacar el titulo de la ventana:

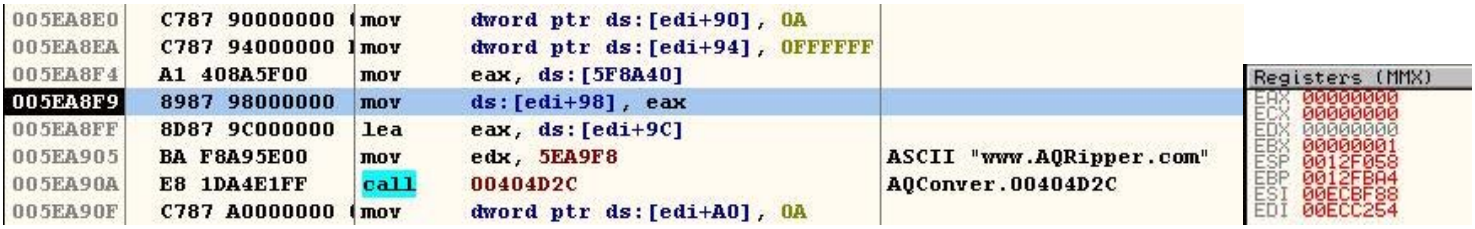
```
005E7410   BA 38755E00   mov     edx, 5E7538                 ASCII "AQ Video Converter Platinum"
005E7415   E8 DE35EAFF   call    0048A9F8                    AQConver.0048A9F8
005E741A   833D 408A5F00 cmp     dword ptr ds:[5F8A40], 0
005E7421 v 74 19         je      short 005E743C              AQConver.005E743C
005E7423   BA 44765E00   mov     edx, 5E7644                 ASCII "AQ Video Converter Platinum (Unregistered)"
005E7428   8B45 FC       mov     eax, ss:[ebp-4]
005E742B   E8 E830E8FF   call    0046A518                    AQConver.0046A518
005E7430   8B45 FC       mov     eax, ss:[ebp-4]
005E7433   C680 28050000 mov     byte ptr ds:[eax+528], 1
005E743A v EB 0D         jmp     short 005E7449              AQConver.005E7449
005E743C   BA 38755E00   mov     edx, 5E7538                 ASCII "AQ Video Converter Platinum"
005E7441   8B45 FC       mov     eax, ss:[ebp-4]
005E7444   E8 CF30E8FF   call    0046A518                    AQConver.0046A518
```

Podemos observar como esta vez ha llegado con el valor 0 en la dirección fija y la condición se cumple asi que demos a F9 para continuar.

Parece ser que ya estamos registrados pero por si acaso probemos a cambiar el tipo de archivo cambiando el cuadro de Profile a ver qué pasa:



Paró aquí al escribir en nuestra dirección fija y vemos que escribe el valor de EAX que el 0xFFFFFFFF. Apunto también esta dirección y pongo de nuevo el valor de la dirección fija a 0 y doy a F9.



Paró de nuevo pero esta vez lo que hace es meter el valor de la dirección fija en EAX y luego lo copia a otro sitio asi que no es problemática esa operación pero ya vemos algo que me interesa pues una de las limitaciones es una marca de agua con la URL de la pagina del creador del software y justo debajo ya vemos la URL. Demos de nuevo a F9 y a ver donde para:

```
005E200A   8B08           mov     ecx, ds:[eax]
005E200C   FF91 CC000000  call    ds:[ecx+CC]
005E2012   A1 68595F00    mov     eax, ds:[5F5968]
005E2017   8338 00        cmp     dword ptr ds:[eax], 0      responsable de poner marca de agua
005E201A   0F84 B7000000  je      005E20D7                   AQConver.005E20D7
005E2020   BA 88245E00    mov     edx, 5E2488                ASCII "www.AQRipper.com"
005E2025   8B83 F8040000  mov     eax, ds:[ebx+4F8]
005E202B   E8 E884E8FF    call    0046A518                   AQConver.0046A518
005E2030   8B83 F8040000  mov     eax, ds:[ebx+4F8]
005E2036   8B40 68        mov     eax, ds:[eax+68]
005E2039   BA A4245E00    mov     edx, 5E24A4                ASCII "Tahoma"
005E203E   E8 C168E4FF    call    00428904                   AQConver.00428904
005E2043   8B83 F8040000  mov     eax, ds:[ebx+4F8]
```

Pues ahí estamos en una zona interesante, pues esta es la encargada de poner la marca de agua pero, como podemos observar, pasamos la condición puesto que pusimos el valor de la dirección fija a 0 y en este caso EAX apunta a la dirección fija con lo cual ya no nos saldrá la marca de agua. Volvemos a dar a F9 y ahora le damos al botón Start para que comience a trabajar y ahora para aquí:

```
005EF63E   8BD8           mov     ebx, eax
005EF640   833D 408A5F00  cmp     dword ptr ds:[5F8A40], 0
005EF647   74 31          je      short 005EF67A
005EF649   8BCB           mov     ecx, ebx
005EF64B   B2 01          mov     dl, 1
005EF64D   A1 6C405E00    mov     eax, ds:[5E406C]
005EF652   E8 FD3EE9FF    call    00483554
005EF657   8BF0           mov     esi, eax
005EF659   8BC6           mov     eax, esi
005EF65B   8B10           mov     edx, ds:[eax]
005EF65D   FF92 EC000000  call    ds:[edx+EC]
005EF663   8BC6           mov     eax, esi
005EF665   E8 E247E1FF    call    00403E4C
005EF66A   837E 0C 01     cmp     dword ptr ds:[esi+C], 1
005EF66E   75 11          jnz     short 005EF681
005EF670   8BC3           mov     eax, ebx
005EF672   E8 B58EFFFF    call    005E852C
005EF677   5E             pop     esi
005EF678   5B             pop     ebx
005EF679   C3             retn
005EF67A   8BC3           mov     eax, ebx
```

Otra comprobación y también la pasamos igual que antes porque aun no ha intentado cambiar el valor del contenido dicha dirección. Damos de nuevo a F9 y comienza la conversión sin problemas pero nos queda aun algo muy importante y es que, según el autor, está limitado también esto y solo convertirá 3 minutos. Este punto está más complicado ya que, al llegar a los 3 minutos, la conversión se corta y no para en nuestro HBP para nada así que usa alguna otra cosa para comprobar. No se apuren, intentaremos otra cosa. Como se puede ver, el programa va monitoreando el tiempo real de película convertida, el tanto por ciento del proceso que se ha realizado, y los Mb que están ocupando por ahora el archivo de salida. Pues bien, este último tiene una cadena de texto la cual tendrá que unir una y otra vez con el número de Mb que tiene el archivo de salida así que intentaremos entrar por ahí. Para ello buscaremos la cadena MB en las strings del programa y pondremos un BP a todas las que encontremos de ese tipo mientras el programa está corriendo y, cuando pare, estaremos en la correcta.

```
005E8BC4  .  50            push    eax                              Arg1
005E8BC5  .  DF6D 0C       fild    qword ptr ss:[ebp+C]
005E8BC8  .  D835 848C5E0( fdiv    dword ptr ds:[5E8C84]
005E8BCE  .  D835 848C5E0( fdiv    dword ptr ds:[5E8C84]
005E8BD4  .  DB7D DC       fstp    tbyte ptr ss:[ebp-24]
005E8BD7  .  9B            wait
005E8BD8  .  8D45 DC       lea     eax, [local.9]
005E8BDB  .  8945 E8       mov     [local.6], eax
005E8BDE  .  C645 EC 03    mov     byte ptr ss:[ebp-14], 3
005E8BE2  .  8D55 E8       lea     edx, [local.6]
005E8BE5  .  33C9          xor     ecx, ecx
005E8BE7  .  B8 908C5E00   mov     eax, 5E8C90                      ASCII "%.1fMB"
005E8BEC  .  E8 FF25E2FF   call    0040B1F0                         De aqui sale con el tamaño en MB que lleva convertido
```

Y si pasamos el CALL con F8 vemos esto:

```
005E8BE7  .  B8 908C5E00   mov     eax, 5E8C90                      ASCII "%.1fMB"
005E8BEC  .  E8 FF25E2FF   call    0040B1F0                         De aqui sale con el tamaño en MB que lleva convertido
005E8BF1  .  8B4D F0       mov     ecx, [local.4]
005E8BF4  .  BA 05000000   mov     edx, 5
005E8BF9  .  8BC6          mov     eax, esi
Stack ss:[0298EC14]=00E5E8D0, (ASCII "4,9MB")
ecx=00000000
```

Pues bien, bajemos un poquito y veremos esto:

```
005E8BF1  .  8B4D F0       mov     ecx, [local.4]
005E8BF4  .  BA 05000000   mov     edx, 5
005E8BF9  .  8BC6          mov     eax, esi
005E8BFB  .  8B30          mov     esi, ds:[eax]
005E8BFD  .  FF56 20       call    ds:[esi+20]
005E8C00  .  817D 1C 38BB( cmp     [arg.6], 2BB38
005E8C07  .v 7E 21         jle     short 005E8C2A                   salto limitador de tiempo
005E8C09  .  A1 448A5F00   mov     eax, ds:[5F8A44]
005E8C0E  .  E8 7DDEFFFF   call    005E6A90                         AQConver.005E6A90
005E8C13  .  85C0          test    eax, eax
```

¿Por qué esa comparación y ese salto me resultaron tan sospechosos? Lo explicaré rápidamente.
Si cogemos 2BB38 y lo pasamos a decimal, nos dará 179000. Bien, ya se va atisbando algo ya que el tiempo se mide en milisegundos, si lo dividimos entre 1000 nos da 179 ¿Cuántos segundos tienen 3 minutos? 180.
Pongamos un BP en la comparación y reiniciemos y volvamos a realizar todas las operaciones anteriores para pasar todos los controles e intentamos convertir un archivo y vamos viendo con que compará 2BB38.
Podrán observar que el valor es un valor hexadecimal que va incrementando y, si quitan este BP y ponen uno en 5E8C09, verán que caerá justo cuando llevamos 3 minutos de conversión.
Pues simplemente con cambiar ese salto por un JMP ya no estaremos limitados a 3 minutos.
Con esto último liquidamos todas las limitaciones que tiene.
Ahora la idea es usar un loader-debugger para que nos ejecute el programa pero sin modificar el ejecutable físicamente, solo en memoria y asi, aunque en este caso no hay detecciones de integridad, poder pasar las detecciones de integridad sobre el archivo.

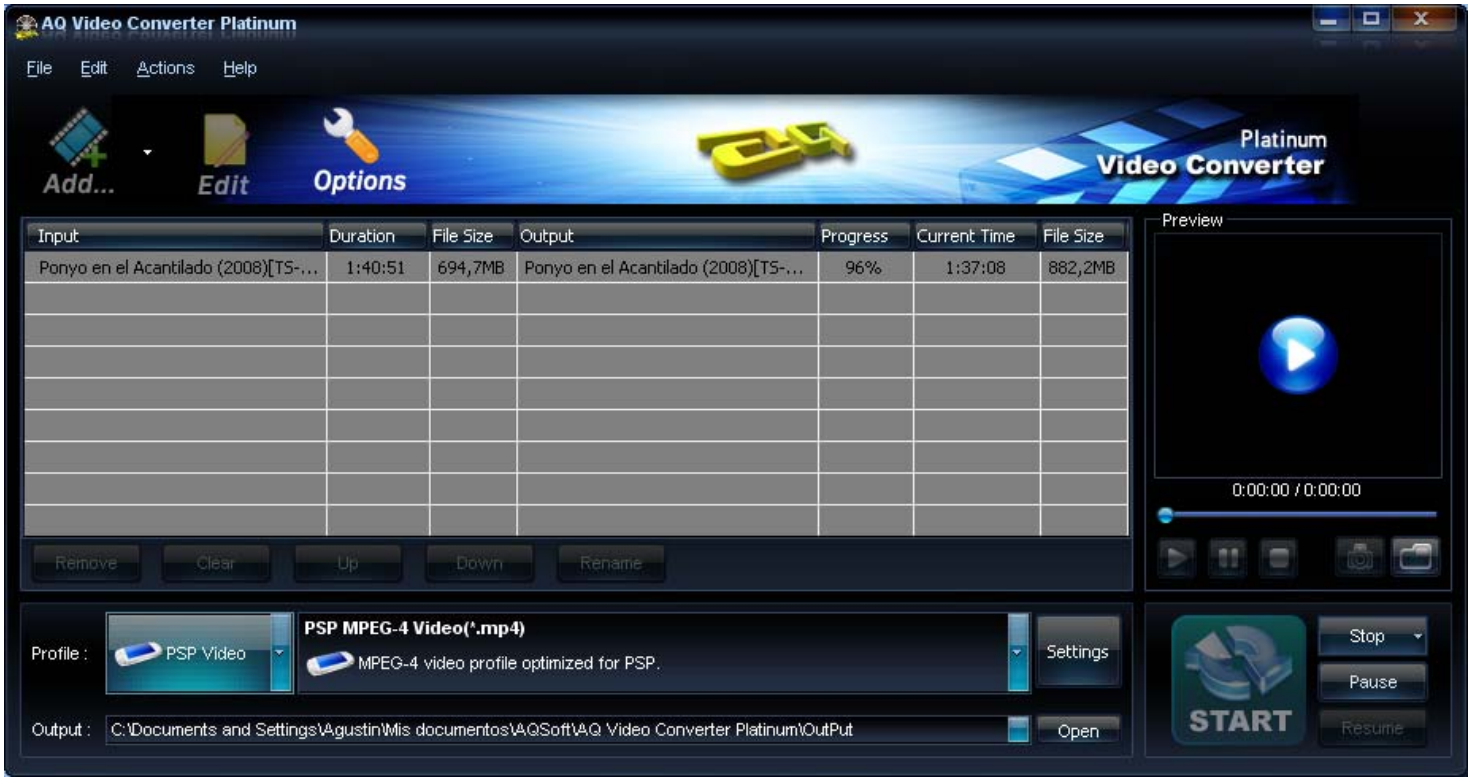
CODIGO DE CREACION DEL LOADER-DEBUGGER

Como estoy aprendiendo a programar Loaders-Debuggers, cogeré a esta victima para practicar.
Lo crearé con la ayuda del componente de stzwei para que vean lo fácil y lo bueno que es dicho componente.
Este es el código del Loader-Debugger usando dicho componente:

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "DbgCLSXv106_OCX"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------

void __fastcall TForm1::DbgCLSX1BreakBP(TObject *Sender)
{
    switch (DbgCLSX1->LastAddr)
    {
        case 0x079530F:  //BP del salto al OEP
            // Con estas dos modificaciones eliminamos el Unregistered, la marca de agua, y alguna que otra
            // comprobación mas
            DbgCLSX1->WriteMemoryDW(0x05E6F3A,0);
            DbgCLSX1->WriteMemoryDW(0x05F8A40,0);

            // Con esta modificación eliminamos límite de 3 minutos
            DbgCLSX1->WriteMemoryB(0x05E8C07,0x0EB);

            // Ponemos dos BPs en estas direcciones para salvar 2 comprobaciones más
            DbgCLSX1->SetBP(0x05E73D0);
            DbgCLSX1->SetBP(0x05EAC4B);

            // Quitamos el BP del salto al OEP
            DbgCLSX1->ClearBP(0x079530F);
            break;

        case 0x05E73D0:
            // Cuando pare en el BP de 5E73D0 ponemos EAX a 0 para salvar una comprobación
            DbgCLSX1->SetEAX(0);
            break;
```

```
            case 0x05EAC4B:
                // Cuando pare en el BP de 5EAC4B ponemos EAX a 0 para salvar una comprobación
                DbgCLSX1->SetEAX(0);
                break;
        }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::DbgCLSX1CreateProcEvent(TObject *Sender)
{
    // Ponemos un BP en el salto al OEP al parar en el Punto de Entrada del programa
    DbgCLSX1->SetBP(0x079530F);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Form1->Hide();  // Ocultamos el Form
    DbgCLSX1->InitDebug();  // Creamos el proceso
    DbgCLSX1->DebugLoop();  // Ejecutamos el Loop para controlar las excepciones
    Application->Terminate();  // Cuando salgamos del programa víctima, cerramos el loader
}
//---------------------------------------------------------------------------
```

Con ese código, simplemente ejecutamos nuestro loader y la aplicación se ejecuta automáticamente y sin limitaciones de ningún tipo y cuando la cerremos el loader se cerrará también.
Miren el resultado:



AGRADECIMIENTOS

Pues igual que en mi tuto anterior, quería agradecer a stzwei por haber hecho algo tan bueno como este componente para crear Loaders-Debuggers y a toda la lista de CracksLatinos en general.