EXCEPCIONES, SEH Y TRAZADORES 7

DISCALIMER

Aquí no encontraras ni seriales, ni cracks, ni codigos de acceso. EL unico proposito de este texto es de carácter educativo, se pretende discutir nuevos tipos de ataques que harian vulnerables la seguridad de los sistemas de proteccion actuales y por lo tanto dejarian desprotegidos nuestros sistemas, nuestros productos o incluso nuestros datos sensibles

INTRODUCCION

DIRECCION WEB :

AkirA : www.iespana.es/ollydbg/akira.html
Joe: www.iespana.es/ollydbg
Crackslatinos: www.**crackslatinos**.hispadominio.net

Hola y bienvenidos a la 64 entrega del curso de AkirA sobre protecciones comerciales

Hace mucho del ultimo tuto que escribi ,bueno, creo que ha merecido la pena la espera, creo que he creado mi mejor trazador hasta la fecha

Bueno, quiero dedicarle el tuto a varias personas. Entre ellas a eSn-mIn , ya hace algun tiempo del triste fallecimiento de este grandisimo cracker y grandisimo amigo, a pesar del tiempo se te sigue echando de menos, para ti va todo mi reconocimiento

Tambien quiero mandar un saludo a Ricardo , a Kort, a +ncr ... a todos los miembros de la lista de crackslatinos , gente admirable de verdad, y por ultimo otra dedicatoria para kaos_xls que me ha mandado la solucion del trazador4b para kernel32, muchas felicidades amigo yo no pude dar con el problema , espero que pronto hagas el tuto.

COMIENZA EL ASEDIO

Hoy vamos a ver mi ultima creación, el trazador de la serie 7.

Este trazador especial es muy importante ya que nos va a dar todas las posibilidades que tiene ollydbg pero sin ser un debugger ni utlizar los mecanismo de comunicación de un debugger ni nada por el estilo, por lo que es indetectable

Las posibilidades que nos brinda es trazador son impresionantes y solo estan limitadas por la imaginación del autor ☺ En este tutorial vamos aver algunos ejemplos e ideas en desarrollo, haremos cosas como sacar el EOP de un armadillo sin trazar y en menos de 3 segundos, veremos un buscador de buffer overflow para servidores, o veremos un programa automatizado que trazara linea a linea un programa/packer haciendo lo que nosotros queramos

Este va a ser un tutorial algo lioso pero muy completo, tomate tu tiempo para leerlo y disfruta.

Ya te adelante que el trazador 7 tiene su base en un gancho especial en la api KiUserExceptionDispatcher de la ntdll.dll

NOTA: como ya recordareis un trazador tiene dos ficheros, el inyector1.exe y la dll que es el nucleo del trazador, como el inyector1.exe es siempre el mismo no voy a poner el codigo aquí, solo hablare de la DLL.

Pero vamos por partes, lo primero es comprender el sistema de interrupciones y de gestion de excepciones en una arquitectura NT .

Cuando se esta ejecutando un programa cualquiera pueden ocurrir situaciones imprevistas : division por cero, interrupción 3, violación de acceso, etc, etc…

Estas situaciones hacen que el micro salte y busque en una tabla llamada IDT la direccion de la rutina que debe atender el tipo de interrupción que se produjo.

Por ejemplo, cuando se produce una INT3 , se coge la entrada 3 de la IDT .

Esa direccion apunta a una rutina que puso el SSOO al arrancar y que se ejecuta en ring0.

En cierto momento esa rutina le pasa el control a la funcion KiDispatchException (todavía estamos en ring0)

PseudoCodigo:

```
VOID KiDispatchException(PEXCEPTION_RECORD Er, ULONG Reserved,
                         PKTRAP_FRAME Tf, MODE PreviousMode,
                         BOOLEAN SearchFrames)
{
    PCR->KeExceptionDispatchCount++;

    CONTEXT Context
        = {CONTEXT_FULL | (PreviousMode == UserMode ? CONTEXT_DEBUG : 0)};

    KeContextFromKframes(Tf, Reserved, &Context);

    if (Er->ExceptionCode == STATUS_BREAKPOINT) Context.Eip--;

    do {
        if (PreviousMode == KernelMode) {
            if (SearchFrames) {
                if (KiDebugRoutine &&
                    KiDebugRoutine(Tf, Reserved, Er, &Context,
                                   PreviousMode, FirstChance) != 0) break;

                if (RtlDispatchException(Er, &Context) == 1) break;
            }
            if (KiDebugRoutine &&
                KiDebugRoutine(Tf, Reserved, Er, &Context,
                               PreviousMode, LastChance) != 0) break;
        }
        else {
            if (SearchFrames) {
                if (PsGetCurrentProcess()->DebugPort == 0
                    || KdIsThisAKdTrap(Tf, &Context)) {

                    if (KiDebugRoutine &&
                        KiDebugRoutine(Tf, Reserved, Er, &Context,
                                       PreviousMode, FirstChance) != 0) b
                }
                if (DbgkForwardException(Tf, DebugEvent,
                                         FirstChance) != 0) return;

                if (valid_user_mode_stack_with_enough_space) {

                    // copy EXCEPTION_RECORD and CONTEXT to user mode sta

                    // push addresses of EXCEPTION_RECORD and CONTEXT
                    // on user mode stack;

                    Tf->Eip = KeUserExceptionDispatcher;

                    return;
                }
            }

            if (DbgkForwardException(Tf, DebugEvent,
                                     LastChance) != 0) return;

            if (DbgkForwardException(Tf, ExceptionEvent,
                                     LastChance) != 0) return;

            ZwTerminateThread(NtCurrentThread(), Er->ExceptionCode);
        }

        KeBugCheckEx(KMODE_EXCEPTION_NOT_HANDLED, Er->ExceptionCode,
                     Er->ExceptionAddress, Er->ExceptionInformation[0],
                     Er->ExceptionInformation[1]);
    } while (false);

    KeContextToKframes(Tf, Reserved, &Context,
                       Context.ContextFlags, PreviousMode);
}
```

Aquí podemos seguir dos caminos :

1)
Si la interrupción se produjo en un driver o cualquier cosa que se ejecute en ring0 , lo primero que se hace en esta rutina es comprobar que la excepcion es manejable.

Si es manejable, primero le da la posibilidad a un depurador de kernel (tipo softice) para que maneje la excepcion.

Si no hay depurador o no la ha manejado llama a la funcion RtlDispatchException pero del modulo ntoskernel.exe para que busque el manejardor de excepciones oportuno
Si no hay manejador instala o no pudo arrglarlo KiDispatcherException indica que la excepcion no es manejable y vuelve a llamar a un depurador de kernel para ver si la maneja esta vez
Si tampoco pudo manejarla pues simplemente invoca a KiBugCheckEx con el mensaje KMODE_EXCEPCION_NOT_HANDLE

2)
Si la excepcion se produjo en un programa que se ejecutaba en modo usuario (que son realmente los programas que a nosotros nos interesa) el camino es otro

Lo primero que hace es mira si la excepcion es manejable
Si es manejable, pira dentro de la TIB a ver si hay depurador de ring3 (tipo Olly). Eso se hace gracias a DebugPort debtro de esa estructura
Si el depurador no esta o no la quiere manejar , entonces se llama a KiUserExceptionDispatcher.

Entonces KiUserExceptionDispatcher (ahora ya en ring3 !!!!!!!!) llama a RtlDispatchException para buscar un manejador de la excepcion, osea una SEH que pueda manejarla
(si quieres saber mas sobre como se almacenan estas estructuras SEH te recomiendo que leas el tuto que hice hace tiempo)

Si ninguna SEH puede manejarla , se indica que la excepcion no es manejable y se le da la posibilidad de nuevo al debugger a que la maneje

Si tampoco el debugger lo maneja entonces se le pasa la excepcion a UnhandlerExcepcion (osea el manejador por defecto del proceso)para ver si el la puede manejar

Si tampoco el pudo manejarla se llama a ZwterminateThread y si este tampoco la puede manejar pues se llama a keBugCheckEx

Bueno, a nosotros lo que mas nos importa es el segundo caso. Osea, que si no hay depurador por medio la primera api que sera informada de la excepcion en nivel de usuario es KiUserExceptionDispatcher , pues a por ella

PseudoCodigo :

```
KiUserExceptionDispatcher( PEXCEPTION_RECORD pExcptRec, CONTEXT *
pContext )
 {
     DWORD retValue;

     // Note: If the exception is handled, RtlDispatchException()
never returns
     if ( RtlDispatchException( pExceptRec, pContext ) )
         retValue = NtContinue( pContext, 0 );
     else
         retValue = NtRaiseException( pExceptRec, pContext, 0 );

     EXCEPTION_RECORD excptRec2;

     excptRec2.ExceptionCode = retValue;
     excptRec2.ExceptionFlags = EXCEPTION_NONCONTINUABLE;
     excptRec2.ExceptionRecord = pExcptRec;
```

```
     excptRec2.NumberParameters = 0;

     RtlRaiseException( &excptRec2 );
 }
```

Bueno como veis reciben dos estructuras como parámetros :

```
typedef struct _EXCEPTION_RECORD
{
     DWORD ExceptionCode;
     DWORD ExceptionFlags;
       struct _EXCEPTION_RECORD *ExceptionRecord;
     PVOID ExceptionAddress;
     DWORD NumberParameters;
     DWORD ExceptionInformation[EXCEPTION_MAXIMUM_PARAMETERS];
 }  EXCEPTION_RECORD;


typedef struct _CONTEXT
{
     DWORD ContextFlags;
     DWORD   Dr0;
     DWORD   Dr1;
     DWORD   Dr2;
     DWORD   Dr3;
     DWORD   Dr6;
     DWORD   Dr7;
     FLOATING_SAVE_AREA FloatSave;
     DWORD   SegGs;
     DWORD   SegFs;
     DWORD   SegEs;
     DWORD   SegDs;
     DWORD   Edi;
     DWORD   Esi;
     DWORD   Ebx;
     DWORD   Edx;
     DWORD   Ecx;
     DWORD   Eax;
     DWORD   Ebp;
     DWORD   Eip;
     DWORD   SegCs;
     DWORD   EFlags;
     DWORD   Esp;
     DWORD   SegSs;
} CONTEXT;
```

En la primera veis que nos poddemos informar de los de cosas generales, el codigo de la excepcion , la direccion desde la que se produjo etc y en la segunda estan los registros intactos de cuando se prdujo la excepcion

Ah , una ultima cosa estas son las posibles excepciones:

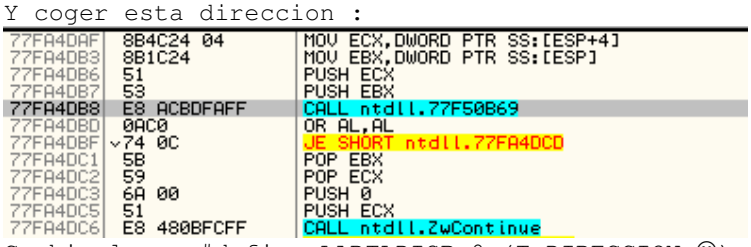
```
#define WAIT_TIMEOUT                        STATUS_TIMEOUT
#define WAIT_IO_COMPLETION                  STATUS_USER_APC
#define STILL_ACTIVE                        STATUS_PENDING
#define EXCEPTION_ACCESS_VIOLATION          STATUS_ACCESS_VIOLATION
```

```
#define EXCEPTION_DATATYPE_MISALIGNMENT   STATUS_DATATYPE_MISALIGNMENT
#define EXCEPTION_BREAKPOINT                STATUS_BREAKPOINT
#define EXCEPTION_SINGLE_STEP               STATUS_SINGLE_STEP
#define EXCEPTION_ARRAY_BOUNDS_EXCEEDED   STATUS_ARRAY_BOUNDS_EXCEEDED
#define EXCEPTION_FLT_DENORMAL_OPERAND   STATUS_FLOAT_DENORMAL_OPERAND
#define EXCEPTION_FLT_DIVIDE_BY_ZERO       STATUS_FLOAT_DIVIDE_BY_ZERO
#define EXCEPTION_FLT_INEXACT_RESULT       STATUS_FLOAT_INEXACT_RESULT
#define EXCEPTION_FLT_INVALID_OPERATION STATUS_FLOAT_INVALID_OPERATION
#define EXCEPTION_FLT_OVERFLOW              STATUS_FLOAT_OVERFLOW
#define EXCEPTION_FLT_STACK_CHECK           STATUS_FLOAT_STACK_CHECK
#define EXCEPTION_FLT_UNDERFLOW             STATUS_FLOAT_UNDERFLOW
#define EXCEPTION_INT_DIVIDE_BY_ZERO     STATUS_INTEGER_DIVIDE_BY_ZERO
#define EXCEPTION_INT_OVERFLOW              STATUS_INTEGER_OVERFLOW
#define EXCEPTION_PRIV_INSTRUCTION       STATUS_PRIVILEGED_INSTRUCTION
#define EXCEPTION_IN_PAGE_ERROR             STATUS_IN_PAGE_ERROR
#define EXCEPTION_ILLEGAL_INSTRUCTION       STATUS_ILLEGAL_INSTRUCTION
#define EXCEPTION_NONCONTINUABLE_EXCEPTION
STATUS_NONCONTINUABLE_EXCEPTION
#define EXCEPTION_STACK_OVERFLOW            STATUS_STACK_OVERFLOW
#define EXCEPTION_INVALID_DISPOSITION       STATUS_INVALID_DISPOSITION
#define EXCEPTION_GUARD_PAGE               STATUS_GUARD_PAGE_VIOLATION
#define EXCEPTION_INVALID_HANDLE            STATUS_INVALID_HANDLE
#define CONTROL_C_EXIT                      STATUS_CONTROL_C_EXIT
```

Bueno, pues vamos a empezar a ver el codigo del trazador7.cpp basico : (NOTA : el como hice el gancho lo explicare al final debido a que KiUserExceptionDispatcher es una funcion especial que es llamada de forma especial y tuve que hacer trucos de magia para poder engancharlo, pero eso solo te interesa si eres programador, aquí vamos a lo nuestro)

Bueno , lo primero que tenemos que tener en cuenta es que necesitamos una cosa previa antes de compilar :
#define LARTLDISP 0x77f50b69
Esta direccion es RtlDispatchException y es distinta en cada ordenador.
Lo primero que debes hacer es ir al Olly, abrir cualquier programa o crackme y dar a GO TO->EXPRESSION y poner KiUserExceptionDispatcher

Y coger esta direccion :



Cambiarla en #define LARTLDISP 0x(TuDIRECCION ☺) y ya lo pudes compilar)

En la funcion void __declspec(naked) MyKiUserExceptionDispatcher es donde vamos a tener la oportunidad de informarnos de cualquier excepcion que ocurra en el programa y actuar antes de que nadie la toque o el programa sea informado del asunto , jejeje

Primero vamos a hacer un simple LOG de las excepciones :

```
void __declspec(naked) MyKiUserExceptionDispatcher( EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      _asm
```

```
	{
		mov ecx,[esp+4];
		mov pContext2,ecx;
		mov ebx,[esp];
		mov pExcptRec2,ebx;
	}

	//MessageBox(0,0,0,0);
	DirRetorno=(DWORD)pExcptRec2->ExceptionAddress;
	VolcarInfo2();
	DirRetorno=pExcptRec2->ExceptionCode;
	VolcarInfo3();
	HookOffOne(&KiUser);
	HookOnOne(&RtlDisp);
	_asm jmp callKiUserExceptionDispatcher;
}
```

Se lo aplicacamos a Pelock :

```
1512F7      EXCEPTION_INT_DIVIDE_BY_ZERO
151AD1      EXCEPTION_ACCESS_VIOLATION
151AD4      EXCEPTION_INT_DIVIDE_BY_ZERO
151C7E      EXCEPTION_ACCESS_VIOLATION
151C81      EXCEPTION_INT_DIVIDE_BY_ZERO
151E1B      EXCEPTION_ACCESS_VIOLATION
151E1E      EXCEPTION_INT_DIVIDE_BY_ZERO
152A0F      EXCEPTION_ILLEGAL_INSTRUCTION
152A0F      EXCEPTION_ILLEGAL_INSTRUCTION
152A0F      EXCEPTION_ILLEGAL_INSTRUCTION
152A0F      EXCEPTION_ILLEGAL_INSTRUCTION
152C8F      EXCEPTION_ACCESS_VIOLATION
152C92      EXCEPTION_INT_DIVIDE_BY_ZERO
152E27      EXCEPTION_ACCESS_VIOLATION
152E2A      EXCEPTION_INT_DIVIDE_BY_ZERO
152EC2      EXCEPTION_ACCESS_VIOLATION
152EC5      EXCEPTION_INT_DIVIDE_BY_ZERO
154986      EXCEPTION_ACCESS_VIOLATION
```

Se lo aplicacamos a Petite :

```
408217      EXCEPTION_ACCESS_VIOLATION
40229E      EXCEPTION_SINGLE_STEP
```

Se lo aplicacamos a Pespin :

```
40851E      EXCEPTION_ACCESS_VIOLATION
4086CA      EXCEPTION_INT_DIVIDE_BY_ZERO
40AD91      EXCEPTION_PRIV_INSTRUCTION
40AD91      EXCEPTION_PRIV_INSTRUCTION
40AD55      EXCEPTION_ACCESS_VIOLATION
40A8F8      EXCEPTION_ACCESS_VIOLATION
408A80      EXCEPTION_ACCESS_VIOLATION
400201      EXCEPTION_ILLEGAL_INSTRUCTION
```

Se lo aplicacamos a Obsidium :

```
40EE7F      EXCEPTION_ACCESS_VIOLATION
150351      EXCEPTION_ACCESS_VIOLATION
77F667CD    EXCEPTION_BREAKPOINT
151479      EXCEPTION_ILLEGAL_INSTRUCTION
15159D      EXCEPTION_INT_DIVIDE_BY_ZERO
```

```
0     EXCEPTION_ACCESS_VIOLATION
0     EXCEPTION_ACCESS_VIOLATION
15477E      EXCEPTION_ACCESS_VIOLATION
15476F      EXCEPTION_ACCESS_VIOLATION
154DDE      EXCEPTION_INT_DIVIDE_BY_ZERO
154DE8      EXCEPTION_ACCESS_VIOLATION
154DE4      EXCEPTION_ACCESS_VIOLATION
154DED      EXCEPTION_ILLEGAL_INSTRUCTION
154F11      EXCEPTION_INT_DIVIDE_BY_ZERO
154DDE      EXCEPTION_INT_DIVIDE_BY_ZERO
154DE8      EXCEPTION_ACCESS_VIOLATION
154DF1      EXCEPTION_BREAKPOINT
155105      EXCEPTION_ACCESS_VIOLATION
152309      EXCEPTION_INT_DIVIDE_BY_ZERO
152309      EXCEPTION_INT_DIVIDE_BY_ZERO
152725      EXCEPTION_INT_DIVIDE_BY_ZERO
152309      EXCEPTION_INT_DIVIDE_BY_ZERO
152725      EXCEPTION_INT_DIVIDE_BY_ZERO
152725      EXCEPTION_INT_DIVIDE_BY_ZERO
405497      EXCEPTION_ILLEGAL_INSTRUCTION
405967      EXCEPTION_INT_DIVIDE_BY_ZERO
```

Se lo aplicacamos a Armadillo :

```
77E53887    77E53887    C61D63      EXCEPTION_PRIV_INSTRUCTION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7CBD3      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7E3E7      EXCEPTION_ILLEGAL_INSTRUCTION
C7E3EC      EXCEPTION_SINGLE_STEP
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7CE04      EXCEPTION_ACCESS_VIOLATION
C7DB2A      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7DF8C      EXCEPTION_ACCESS_VIOLATION
C7CBD3      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7D266      EXCEPTION_ACCESS_VIOLATION
77C13636    EXCEPTION_ACCESS_VIOLATION
```

Las posibilidades son infinitas, cuando se produzca la ultima excepcion se supone que el codigo y los datos estan desencriptados y se puede volcar. Si tenemos un packer que desencripta el codigo por trozos mientras se ejecuta, podemos informarnos de la direccion e ir volcando todo conforme se vayan produciendo y todo ello sin un debugger (lógicamente , la primera excepcion de esta naturaleza que este dentro de la seccoind e codigo sera el EOP , jejej)

Pero vamos a ver una posibilidad mejor. Recordais al principio que dije que este trazador nos daba todas las posibilidades de un debugger como Oly, pues vamos a ver un ejemplo. (vamos a sacar el EOP de

armadillo) Como podeis ver arriba en el log de armadillo se han producido 20 excepciones de tipo EXCEPTION_ACCESS_VIOLATION, lo que voy a hacer es que cuando se produzca la ultima y el packer este a punto de saltar al EOP, voy a cambiar los permisos de acceso a la seccion de la seccion de codigo a DWORD NewP=PAGE_NOACCESS;

Y el nuevo codigo sera :

```
void __declspec(naked) MyKiUserExceptionDispatcher( EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      _asm
      {
            mov ecx,[esp+4];
            mov pContext2,ecx;
            mov ebx,[esp];
            mov pExcptRec2,ebx;
      }

      //MessageBox(0,0,0,0);
      DirRetorno=(DWORD)pExcptRec2->ExceptionAddress;
      VolcarInfo2();
      DirRetorno=pExcptRec2->ExceptionCode;
      VolcarInfo3();
      if(pExcptRec2->ExceptionCode==EXCEPTION_ACCESS_VIOLATION)
      {
            if(Contador ==20 )
            {
                  VirtualProtect((void *)0x401000,0x1000,NewP,&OldP);
                  flag=true;

            }

            Contador++;
      }

      HookOffOne(&KiUser);
      HookOnOne(&RtlDisp);
      _asm jmp callKiUserExceptionDispatcher;
}
```

lo que va a ocurrir aquí es que cuando el packer quiera saltar al EOP , se producira otra excepcion EXCEPTION_ACCESS_VIOLATION que lógicamente, yo sere el primero en coger, jeje, antes que el programa.

Veamos:

```
77E53887    77E53887    C61D63      EXCEPTION_PRIV_INSTRUCTION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7CBD3      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7E3E7      EXCEPTION_ILLEGAL_INSTRUCTION
C7E3EC      EXCEPTION_SINGLE_STEP
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7CE04      EXCEPTION_ACCESS_VIOLATION
C7DB2A      EXCEPTION_ACCESS_VIOLATION
```

```
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7DF8C      EXCEPTION_ACCESS_VIOLATION
C7CBD3      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7C96E      EXCEPTION_ACCESS_VIOLATION
C7D266      EXCEPTION_ACCESS_VIOLATION
401099      EXCEPTION_ACCESS_VIOLATION

JEJEJEE , efectivamente, el EOP es 401099 ¡!!!!!!

Como ademas yo soy el primero en coger las excepciones, antes de que
el programa se entere podria poner esto :

void __declspec(naked) MyKiUserExceptionDispatcher( EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      _asm
      {
            mov ecx,[esp+4];
            mov pContext2,ecx;
            mov ebx,[esp];
            mov pExcptRec2,ebx;
      }

      //MessageBox(0,0,0,0);
      DirRetorno=(DWORD)pExcptRec2->ExceptionAddress;
      VolcarInfo2();
      DirRetorno=pExcptRec2->ExceptionCode;
      VolcarInfo3();
      if(pExcptRec2->ExceptionCode==EXCEPTION_ACCESS_VIOLATION)
      {
            if(Contador ==20 )
            {
                  VirtualProtect((void *)0x401000,0x1000,NewP,&OldP);
                  flag=true;

            }

            Contador++;
      }

      if(pExcptRec2->ExceptionCode==EXCEPTION_ACCESS_VIOLATION &&
flag==true)
      {
            VirtualProtect((void *)0x401000,0x1000,OldP,&NewP);
            flag=flase;
            HookOffOne(&KiUser);
            MessageBox(0,Atacame Con el Olly,EOP,0); //!!!!!!!!!!!!!!!
            callZwContinue( pContext2, 0 );

      }

      HookOffOne(&KiUser);
      HookOnOne(&RtlDisp);
      _asm jmp callKiUserExceptionDispatcher;
}
```

Osea, que devuelvo los permisos a su sitio y saco en MessageBox que me da tiempo a que lo ataque con el Olly cambiar poner un bpx en la siguiente intruccion y ya seria mio

Bueno, esto por un lado, la verdad es que las posibilidades son tantas , que no voy a llamarlos trazador7b otrazador7c etc, yo doy el trazador7 basico y que cada uno implemente esa idea genial que siempre queso hacer ☺

Aunque, tambien podemos hacer que el gancho saque en una ventanita el valor de los registros y el codigo y tal y hacernos un debugger alternativo “NO agresivo” y no detectable jeje

Otra posibilidad que puede revolucionar mucho el tema de la seguridad informatica es el buscador automatico de buffer overflow

Como sabeis los SEH son un sistema de recuperacion de posibles excepciones, como la del desbordamiento de buffer. Pero es lo mas importante de todo, es un sistema de recuperacion , no de prevencion, osea, que un programador de un servidor, ftp, o http, o mail, o lo que sea que se comunique con sockets o lpc o lo que sea, dice : voy a poner una SEH o un try/caght para que si se produce algun fallo de desbordamiento que no he podido prever, el programa sea capaz de gestionarlo y no se note, solo vuelve al estado original o no se procesa la petición y ya esta.

Eso esta bien, no se ven los efectos al exterior ni sale un mensaje diciendo que se produjo un error, pero no evita que el hecho de que la excepcion o el desbordamiento de buffer si que se ha producido, entonces si arrancamos un programa con el trazador basico , incluso aunque el Server este protegido por el mejor de los packers, podremos hacer un log de cuando se produce una excepcion y podremos volcar el contenido de los registros en ese preciso instante y ver justamente en donde se produjo el desbordamiento y saber cuanto espacio tenemos para el payload

Lo unico que hay que hacer es escribir por otra parte un programa que mediante sockets automatice el proceso de comunicarse con un Server ftp o de lo que sea y mandar todos los posibles comandos seguidos cde cadenas larguisimas ¡!!!!! Para ir comprobado en nuestro log si se van produciendo excepciones, si se produce alguna, es porque ese comando es vulnerable y tendremos toda la información de en que direccion y que tenian los registros en ese momento gracias al log

Este estudio esta en proceso de desarrollo aun, ya que necesito un programa libre para hacer las pruebas, que tampoco es cuestion de hacer mucho ruido , los resultados de este estudio se publicaran en un futuro

Asi , que pasaremos a ver otra posible opcion del trazador 7 que es muy interesante.

Muchas veces hemos intentado trazar un programa linea a linea y nos hemos dado cuenta una hora después que es estudido porque un programa tiene millones de instrucciones y que para hacer esto habria que ser una maquina.

Bueno, pues gracias a esto haremos que efectivamente, sea una maquina la que haga el trabajo de trazar linea a linea un programa.

Ademas haremos que cada vez que vaya a ejecutarse un JMP o un RETN , nos haga un log, para saber a donde va a saltar (puede ser un EOP) (NOTA: Mientras escribo me estoy dando cuenta de que esto no es necesario , que cuando se produzca un EXCEPTION_STEP , puedo mirar la direccion EIP y si esta dentro de los limites de la seccion de codigo es que es el EOP ☺ , jajajaja)
bueno, pero ya que lo he hecho veamos un ejemplo para poder aplicarlo en un futuro a nuevas cosas por ejemplopara atacar a los RDTSC que estan en todos los virus buenos a la hora de no dejar que veamos como se desencriptan, y no los que hacen ahora algunos adolescentes que escriben un programa simple sin protecciones ni nada, que te estropea el disco duro y lo llaman virus ☹ el arte en un virus esta en las tecnicas que usa no en el daño irreparable que pueda causar ...)

Lo que vamos a hacer es coger por ejemplo a Petite y esperar a que se ejecute la ultima excepcion :

```
408217      EXCEPTION_ACCESS_VIOLATION
40229E      EXCEPTION_SINGLE_STEP
```

Cuando se produzca la ultima excepcion lo que haremos es activar el bit de traza y devolverle la posibilidad de que se ejecute.

Entonces, se iran produciendo excepciones de tipo EXCEPTION_SINGLE_STEP que iremos trazapando y analizando y registrando si fuera oportuno y sin que se entere el packer jejejej

```
void __declspec(naked) MyKiUserExceptionDispatcher( EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      _asm
      {
            mov ecx,[esp+4];
            mov pContext2,ecx;
            mov ebx,[esp];
            mov pExcptRec2,ebx;
      }

      //MessageBox(0,0,0,0);
      DirRetorno=(DWORD)pExcptRec2->ExceptionAddress;
      DirRetorno=pExcptRec2->ExceptionCode;

      if(pExcptRec2->ExceptionCode==EXCEPTION_SINGLE_STEP)
      {
            TrazarJMP(pExcptRec2, pContext2);
            pContext2->EFlags = (DWORD)pContext2->EFlags | 0x100;
            callZwContinue( pContext2, 0 );
      }

      HookOffOne(&KiUser);
      HookOnOne(&RtlDisp);

      _asm jmp callKiUserExceptionDispatcher;
}
```

Veis que hay una funcion llamada TrazarJMP sirve para ver si la instrucción que debe ejecutarse ahora es un JMP su codigo es :

```
void TrazarJMP(EXCEPTION_RECORD * ExceptionRecord, CONTEXT *
ContextRecord)
{

      DWORD DireccionExp=(DWORD)ExceptionRecord->ExceptionAddress;
      char * CharDireccionExp=(char *)ExceptionRecord-
>ExceptionAddress;
      //printf("Direccion %x\n",DireccionExp);
      char Opcode1=*CharDireccionExp;
      //printf("Opcode1 %x\n",(BYTE)Opcode1);
      char Opcode2=*(CharDireccionExp+1);
      //printf("Opcode2 %x\n",(BYTE)Opcode2);


      FILE * archivo=fopen("log.txt","a+b");

      if(archivo==NULL)
      {
            return;
      }
      fseek(archivo,0,SEEK_END);

      if (0xE9==(BYTE)Opcode1)
      {
            DWORD DireccionDestino=0;
            memcpy((void *)&DireccionDestino,(void *)&Opcode2,4);
            DireccionDestino+=DireccionExp;
            DireccionDestino+=5;
            fprintf(archivo,"%X",DireccionExp);
            fprintf(archivo,"\t");
            fprintf(archivo,"JMP E9");
            fprintf(archivo,"\t");
            fprintf(archivo,"%X",DireccionDestino);
            fprintf(archivo,"\n");
      }

      if(0xEB== (BYTE)Opcode1)
      {
            DWORD DireccionDestino=0;
            memcpy((void *)&DireccionDestino,(void *)&Opcode2,2);
            DireccionDestino+=DireccionExp;
            DireccionDestino+=2;
            fprintf(archivo,"%X",DireccionExp);
            fprintf(archivo,"\t");
            fprintf(archivo,"JMP E9");
            fprintf(archivo,"\t");
            fprintf(archivo,"%X",DireccionDestino);
            fprintf(archivo,"\n");
      }

      fclose(archivo);
}
```

Aunque tambien podemos hacer un TrazarRETN , etc

```
void TrazarRetn(EXCEPTION_RECORD * ExceptionRecord, CONTEXT *
ContextRecord)
{

      DWORD DireccionExp=(DWORD)ExceptionRecord->ExceptionAddress;
```

```
      char * CharDireccionExp=(char *)ExceptionRecord-
>ExceptionAddress;
      //printf("Direccion %x\n",DireccionExp);
      char Opcode1=*CharDireccionExp;
      //printf("Opcode1 %x\n",(BYTE)Opcode1);
      char Opcode2=*(CharDireccionExp+1);
      //printf("Opcode2 %x\n",(BYTE)Opcode2);


      FILE * archivo=fopen("log.txt","a+b");

      if(archivo==NULL)
      {
            return;
      }
      fseek(archivo,0,SEEK_END);

      if (0xC3==(BYTE)Opcode1)
      {
            DWORD DireccionDestino=0;
            DWORD Pila=ContextRecord->Esp;
            memcpy((void *)&DireccionDestino,(void *)Pila,4);

            fprintf(archivo,"%X",DireccionExp);
            fprintf(archivo,"\t");
            fprintf(archivo,"RETN C3");
            fprintf(archivo,"\t");
            fprintf(archivo,"%X",DireccionDestino);
            fprintf(archivo,"\n");
      }

      fclose(archivo);
}
```

Total que vamos a probarlo a ver que pasa :

```
408217      EXCEPTION_ACCESS_VIOLATION
40229E      EXCEPTION_SINGLE_STEP
40229F      EXCEPTION_SINGLE_STEP
4022A6      EXCEPTION_SINGLE_STEP
4022A7      EXCEPTION_SINGLE_STEP
4022A9      EXCEPTION_SINGLE_STEP
4022AA      EXCEPTION_SINGLE_STEP
4022AC      EXCEPTION_SINGLE_STEP
4022B1      EXCEPTION_SINGLE_STEP
4022B4      EXCEPTION_SINGLE_STEP
4022B8      EXCEPTION_SINGLE_STEP
4022BA      EXCEPTION_SINGLE_STEP
4022C1      EXCEPTION_SINGLE_STEP
4022C5      EXCEPTION_SINGLE_STEP
4022CC      EXCEPTION_SINGLE_STEP
4022CD      EXCEPTION_SINGLE_STEP
4022D0      EXCEPTION_SINGLE_STEP
4022D3      EXCEPTION_SINGLE_STEP
4022D6      EXCEPTION_SINGLE_STEP
4022D7      EXCEPTION_SINGLE_STEP
4022B4      EXCEPTION_SINGLE_STEP
4022B8      EXCEPTION_SINGLE_STEP
4022BA      EXCEPTION_SINGLE_STEP
4022C1      EXCEPTION_SINGLE_STEP
```

```
4022C5      EXCEPTION_SINGLE_STEP
4022CC      EXCEPTION_SINGLE_STEP

... etc
```

Bueno, esta tecnica todavia es muy lenta y genera una cantidad de informacion excesiva . Enun futuro seguire trabajando con ello para alcanzar mejores prestaciones.

Bueno , una ultima cosa, podemos hacer un TrazarRDSTC y si comprobamos que la instrucción que se va a ejecutar es 0x0f31 le sumamos a EIP 2 bytes y en EAX, y en EDX le ponemos el TIMESTAMP que queramos para que no detecte retardos ☺

Bueno, pues con esto y un bizcocho hemos hecho otro trazador con multiples posibilidades.

Espero que te hayas divertido y que hayas aprendido mucho

NOTA FINAL : SOLO PARA PROGRAMADORES

Bueno, como podreis observar en el codigo de abajo he tenido que hookear no solo la KiUserExceptionDisptach , sino tambien RtlDispatchException , la razon es que KiUserExceptionDisptach es llamada de forma especial, no tiene retn , por lo que no podia quitar el gancho llamarla y luego volver a poner el gancho, asi que tuve que inventarme este truco.

Como lo que hace KiUserExceptionDisptach es llamar a RtlDispatchException lo que hago es lo siguiente

Pongo el gancho en KiUserExceptionDisptach
Cuando se ejecuta KiUserExceptionDisptach mi gancho hace el LOG , pone otro gancho en , RtlDispatchException quita el gancho de KiUserExceptionDisptach y deja que se ejecute normal

(ahora mismo, KiUserExceptionDisptach esta normal)

Entonces, cuando salte el gancho de RtlDispatchException ,volvera a poner el gancho en KiUserExceptionDisptach , RtlDispatchException quitara el suyo en y asi consigo que poner otra vez el gancho en KiUserExceptionDisptach y repetir de nuevo el bucle

El unico problema es que RtlDispatchException no es una funcion exportable, entonces no puedo usar GetProcAddress y por lo tanto hay que usar el Olly para sacar la direccion de RtlDispatchException antes de compilar el trazador 7

CODIGOS

Trazador 7 basico

```
#include "windows.h"
#include <stdio.h>
//#include "TrazarJMP.h"
//#include "TrazarRetn.h"

#define LARTLDISP 0x77f50b69
```

```
typedef struct
{
    FARPROC funcaddr;
    BYTE    olddata[5];
    BYTE    newdata[5];
}HOOKSTRUCT;

HHOOK       g_hHook;
HINSTANCE   g_hinstDll;
HMODULE     hModule ;
HANDLE      g_hForm;
DWORD       dwIdOld, dwIdNew;


HOOKSTRUCT  KiUser;
HOOKSTRUCT  RtlDisp;

DWORD contador=0;
DWORD DirRetorno=0;
DWORD Retornos[2000];


BOOL Init();
BOOL InstallHook();
BOOL UninstallHook();
void HookOnOne(HOOKSTRUCT *hookfunc);
void HookOffOne(HOOKSTRUCT *hookfunc);
BOOL hookapi(char *dllname, char *procname, DWORD myfuncaddr,
HOOKSTRUCT *hookfunc);
BOOL hookapi2(char *dllname, char *procname, DWORD myfuncaddr,
HOOKSTRUCT *hookfunc);
void VolcarInfo();
void VolcarInfo2();
void VolcarInfo3();

EXCEPTION_RECORD * pExcptRec2;
CONTEXT * pContext2;


void MyKiUserExceptionDispatcher(EXCEPTION_RECORD * pExcptRec, CONTEXT
* pContext );
void MyRtlDispatchException(EXCEPTION_RECORD * pExcptRec, CONTEXT *
pContext );

typedef void (WINAPI *ptrKiUserExceptionDispatcher)(EXCEPTION_RECORD
*, CONTEXT *);
ptrKiUserExceptionDispatcher callKiUserExceptionDispatcher;

typedef void (WINAPI *ptrRtlDispatchException)(EXCEPTION_RECORD *,
CONTEXT *);
ptrRtlDispatchException callRtlDispatchException;

typedef DWORD (WINAPI *ptrZwContinue)(CONTEXT *,int);
ptrZwContinue callZwContinue;

BOOL APIENTRY DllMain( HINSTANCE hInstance, DWORD  ul_reason_for_call,
LPVOID lpReserved)
{

   if(ul_reason_for_call == DLL_PROCESS_ATTACH)
```

```
   {
                Init();
   }

   if(ul_reason_for_call ==  DLL_THREAD_ATTACH)
   {
   }
   if(ul_reason_for_call ==  DLL_THREAD_DETACH)
   {
   }
   if(ul_reason_for_call ==  DLL_PROCESS_DETACH)
   {
          UninstallHook();
   }
    return TRUE;
}

//-----------------------------------------------------------------------
---
BOOL Init()
{

      callKiUserExceptionDispatcher=(ptrKiUserExceptionDispatcher)GetP
rocAddress(GetModuleHandle("ntdll.dll"),"KiUserExceptionDispatcher");
      //callRtlDispatchException=(ptrRtlDispatchException)GetProcAddre
ss(GetModuleHandle("ntdll.dll"),"RtlDispatchException");
      callRtlDispatchException=(ptrKiUserExceptionDispatcher)LARTLDISP
;
      callZwContinue=(ptrZwContinue)GetProcAddress(GetModuleHandle("nt
dll.dll"),"ZwContinue");

      if(callKiUserExceptionDispatcher==NULL) MessageBox(0,"NO1",0,0);
      if(callRtlDispatchException==NULL) MessageBox(0,"NO2",0,0);
      if(callZwContinue==NULL) MessageBox(0,"NO3",0,0);

      hookapi("ntdll.dll", "KiUserExceptionDispatcher",
(DWORD)MyKiUserExceptionDispatcher, &KiUser);
      hookapi2("ntdll.dll", "RtlDispatchException",
(DWORD)MyRtlDispatchException, &RtlDisp);
      dwIdNew = GetCurrentProcessId();
    dwIdOld = dwIdNew;
      HookOnOne(&KiUser);

    return(true);
}

BOOL hookapi(char *dllname, char *procname, DWORD myfuncaddr,
HOOKSTRUCT *hookfunc)
{
    hModule = LoadLibrary(dllname);
    hookfunc->funcaddr = GetProcAddress(hModule, procname);
    if(hookfunc->funcaddr == NULL)
        return false;

    memcpy(hookfunc->olddata, hookfunc->funcaddr, 6);
    hookfunc->newdata[0] = 0xe9;
    DWORD jmpaddr = myfuncaddr - (DWORD)hookfunc->funcaddr - 5;
    memcpy(&hookfunc->newdata[1], &jmpaddr, 5);
    return true;
}
```

```
BOOL hookapi2(char *dllname, char *procname, DWORD myfuncaddr,
HOOKSTRUCT *hookfunc)
{
    /*hModule = LoadLibrary(dllname);
    hookfunc->funcaddr = GetProcAddress(hModule, procname);
    if(hookfunc->funcaddr == NULL)
        return false;*/

      hookfunc->funcaddr=(FARPROC)LARTLDISP;
    memcpy(hookfunc->olddata, hookfunc->funcaddr, 6);
    hookfunc->newdata[0] = 0xe9;
    DWORD jmpaddr = myfuncaddr - (DWORD)hookfunc->funcaddr - 5;
    memcpy(&hookfunc->newdata[1], &jmpaddr, 5);
    return true;
}

//-----------------------------------------------------------------------
-------
void HookOnOne(HOOKSTRUCT *hookfunc)
{
    HANDLE hProc;
    dwIdOld = dwIdNew;
    hProc = OpenProcess(PROCESS_ALL_ACCESS, 0, dwIdOld);
    VirtualProtectEx(hProc, hookfunc->funcaddr, 5,
PAGE_READWRITE,&dwIdOld);
    WriteProcessMemory(hProc, hookfunc->funcaddr, hookfunc->newdata,
5, 0);
    VirtualProtectEx(hProc, hookfunc->funcaddr, 5, dwIdOld, &dwIdOld);
}


void HookOffOne(HOOKSTRUCT *hookfunc)
{
    HANDLE hProc;
    dwIdOld = dwIdNew;
    hProc = OpenProcess(PROCESS_ALL_ACCESS, 0, dwIdOld);
    VirtualProtectEx(hProc, hookfunc->funcaddr,5, PAGE_READWRITE,
&dwIdOld);
    WriteProcessMemory(hProc, hookfunc->funcaddr, hookfunc->olddata,
5, 0);
    VirtualProtectEx(hProc, hookfunc->funcaddr, 5, dwIdOld, &dwIdOld);
}

BOOL UninstallHook()
{
    HookOffOne(&KiUser);
      HookOffOne(&RtlDisp);
      //VolcarInfo();
      return true;

}

//-----------------------------------------------------------------------
----------------------
void __declspec(naked) MyKiUserExceptionDispatcher( EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      _asm
      {
            mov ecx,[esp+4];
            mov pContext2,ecx;
```

```
            mov ebx,[esp];
            mov pExcptRec2,ebx;
      }

      //MessageBox(0,0,0,0);
      DirRetorno=(DWORD)pExcptRec2->ExceptionAddress;
      VolcarInfo2();
      DirRetorno=pExcptRec2->ExceptionCode;
      VolcarInfo3();
      HookOffOne(&KiUser);
      HookOnOne(&RtlDisp);
      _asm jmp callKiUserExceptionDispatcher;
}
//-----------------------------------------------------------------------
----------------
void __declspec(naked) MyRtlDispatchException(EXCEPTION_RECORD *
pExcptRec, CONTEXT * pContext )
{
      HookOffOne(&RtlDisp);
      HookOnOne(&KiUser);
      _asm jmp callRtlDispatchException;
}

//-----------------------------------------------------------------------
----------------
void VolcarInfo2()
{


      FILE * archivo=fopen("log.txt","a+b");

      if(archivo==NULL)
      {
            return;
      }

      fseek(archivo,0,SEEK_END);
      fprintf(archivo,"%X \t",DirRetorno);
      fclose(archivo);
}


//-----------------------------------------------------------------------
----------------

void VolcarInfo3()
{


      FILE * archivo=fopen("log.txt","a+b");

      if(archivo==NULL)
      {
            return;
      }

      fseek(archivo,0,SEEK_END);

      if(DirRetorno==WAIT_TIMEOUT)
      {
            fprintf(archivo,"WAIT_TIMEOUT\n");
```

```
}
if(DirRetorno==WAIT_IO_COMPLETION)
{
      fprintf(archivo,"WAIT_IO_COMPLETION\n");
}
if(DirRetorno==STILL_ACTIVE)
{
      fprintf(archivo,"STILL_ACTIVE\n");
}
if(DirRetorno==EXCEPTION_ACCESS_VIOLATION)
{
      fprintf(archivo,"EXCEPTION_ACCESS_VIOLATION\n");
}
if(DirRetorno==EXCEPTION_DATATYPE_MISALIGNMENT)
{
      fprintf(archivo,"EXCEPTION_DATATYPE_MISALIGNMENT\n");
}
if(DirRetorno==EXCEPTION_BREAKPOINT)
{
      fprintf(archivo,"EXCEPTION_BREAKPOINT\n");
}
if(DirRetorno==EXCEPTION_SINGLE_STEP)
{
      fprintf(archivo,"EXCEPTION_SINGLE_STEP\n");
}
if(DirRetorno==EXCEPTION_ARRAY_BOUNDS_EXCEEDED)
{
      fprintf(archivo,"EXCEPTION_ARRAY_BOUNDS_EXCEEDED\n");
}
if(DirRetorno==EXCEPTION_FLT_DENORMAL_OPERAND)
{
      fprintf(archivo,"EXCEPTION_FLT_DENORMAL_OPERAND\n");
}
if(DirRetorno==EXCEPTION_FLT_DIVIDE_BY_ZERO)
{
      fprintf(archivo,"EXCEPTION_FLT_DIVIDE_BY_ZERO\n");
}
if(DirRetorno==EXCEPTION_FLT_INEXACT_RESULT)
{
      fprintf(archivo,"EXCEPTION_FLT_INEXACT_RESULT\n");
}
if(DirRetorno==EXCEPTION_FLT_INVALID_OPERATION)
{
      fprintf(archivo,"EXCEPTION_FLT_INVALID_OPERATION\n");
}
if(DirRetorno==EXCEPTION_FLT_OVERFLOW)
{
      fprintf(archivo,"EXCEPTION_FLT_OVERFLOW\n");
}
if(DirRetorno==EXCEPTION_FLT_UNDERFLOW)
{
      fprintf(archivo,"EXCEPTION_FLT_UNDERFLOW\n");
}
if(DirRetorno==EXCEPTION_INT_DIVIDE_BY_ZERO)
{
      fprintf(archivo,"EXCEPTION_INT_DIVIDE_BY_ZERO\n");
}
if(DirRetorno==EXCEPTION_INT_OVERFLOW)
{
      fprintf(archivo,"EXCEPTION_INT_OVERFLOW\n");
}
```

```
	if(DirRetorno==EXCEPTION_PRIV_INSTRUCTION)
	{
		fprintf(archivo,"EXCEPTION_PRIV_INSTRUCTION\n");
	}
	if(DirRetorno==EXCEPTION_IN_PAGE_ERROR)
	{
		fprintf(archivo,"EXCEPTION_IN_PAGE_ERROR\n");
	}
	if(DirRetorno==EXCEPTION_ILLEGAL_INSTRUCTION)
	{
		fprintf(archivo,"EXCEPTION_ILLEGAL_INSTRUCTION\n");
	}
	if(DirRetorno==EXCEPTION_NONCONTINUABLE_EXCEPTION)
	{
		fprintf(archivo,"EXCEPTION_NONCONTINUABLE_EXCEPTION\n");
	}
	if(DirRetorno==EXCEPTION_STACK_OVERFLOW)
	{
		fprintf(archivo,"EXCEPTION_STACK_OVERFLOW\n");
	}
	if(DirRetorno==EXCEPTION_INVALID_DISPOSITION)
	{
		fprintf(archivo,"EXCEPTION_INVALID_DISPOSITION\n");
	}
	if(DirRetorno==EXCEPTION_GUARD_PAGE)
	{
		fprintf(archivo,"EXCEPTION_GUARD_PAGE\n");
	}
	if(DirRetorno==EXCEPTION_INVALID_HANDLE)
	{
		fprintf(archivo,"EXCEPTION_INVALID_HANDLE\n");
	}
	if(DirRetorno==CONTROL_C_EXIT)
	{
		fprintf(archivo,"CONTROL_C_EXIT\n");
	}


	fclose(archivo);
}
```