En este escrito desarrollaremos las ventanas de IDA, este tiene tres estructuras de ventanas las principales, secundarias y terciarias. Por lo tanto tendremos tres partes sobre el tema ventanas

# 4.—Las vistas principales de IDA

En su configuración por defecto, IDA versión 5.2 crea ocho vistas de datos distintas en la fase inicial de carga y análisis de un binario nuevo. Cada una de esas ventanas de datos, son accesibles gracias a las solapas mostradas justo debajo de la banda de navegación, como ya vimos anteriormente. Las tres primeras ventanas visibles son la ventana **IDA-View**, la ventana **Names** y la ventana **mensajes**. La ventana **Strings** puede mostrarse por defecto o no. Suponiendo que no esté abierta por defecto, podemos abrirla realizando la acción **View > Open subview > Strings**. Esta acción hay que tenerla siempre en mente, ya que es muy fácil cerrar alguna vista inadvertidamente, precisamente no acordarte de como habilitar la vista cerrada “cabrea mucho”.

La tecla **ESC** es uno de los atajos más utilizado en IDA. Cuando la ventana de desensamblado está activa, la función **ESC** se utiliza de manera similar al botón página anterior de un navegador web, lo cual es muy interesante para navegar a través de la ventana de desensamblado, dicha navegación la trataremos más adelante. Por desgracia, cuando cualquier otra ventana está activa, la tecla **ESC** sirve para cerrar dicha ventana. Algunas veces será lo que desees realizar. En otras ocasiones sin embargo, querrías que no hubiera sucedido. Por lo tanto “ojo al parche”.

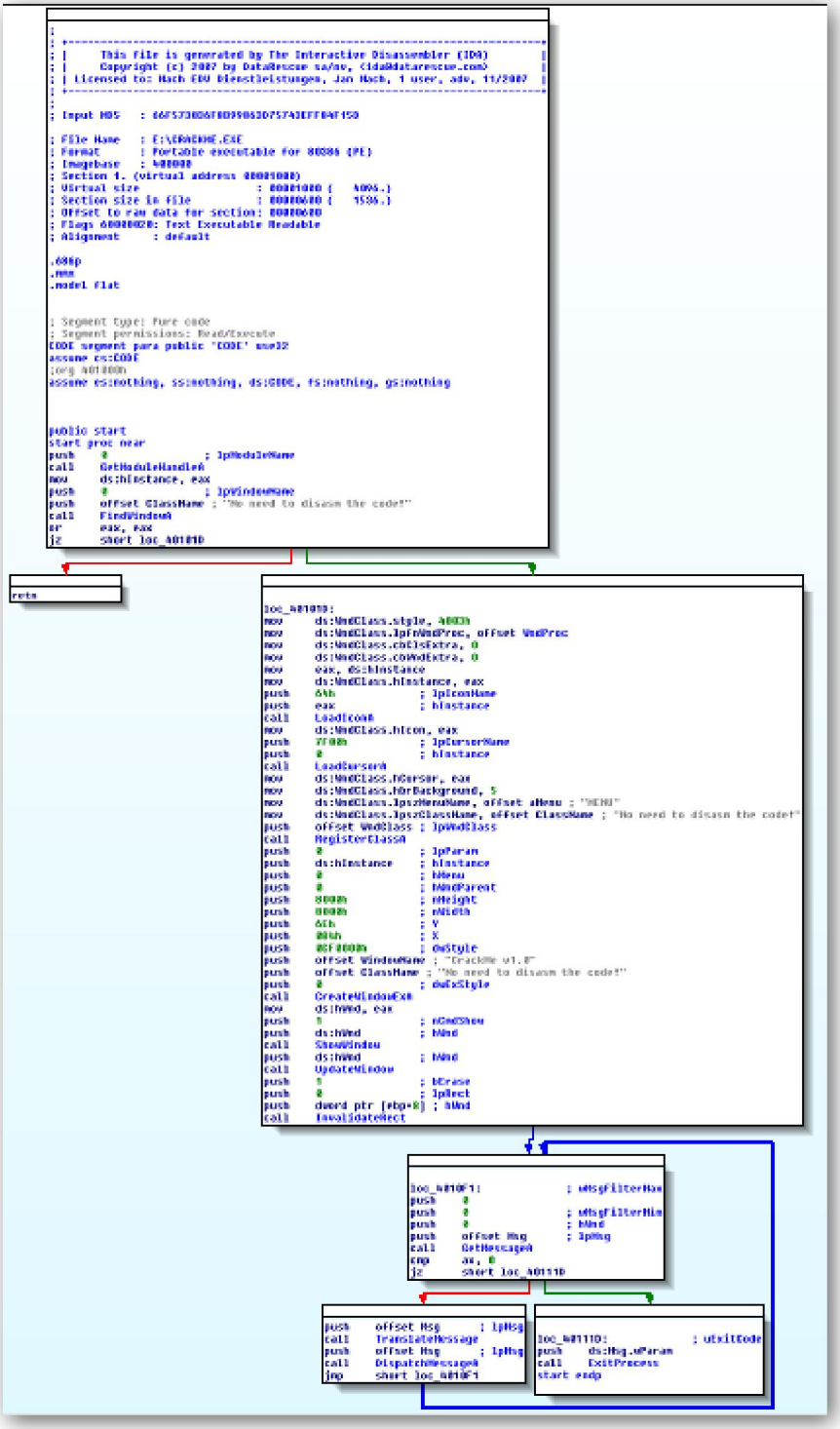
## 4.1.—La ventana de desensamblado

La ventana de desensamblado también es llamada **IDA-View**, dicha ventana será la principal herramienta para manipular y analizar los binarios. Por lo tanto es primordial que te familiarices íntimamente en la forma en que la información es mostrada en dicha ventana.

Como ya hemos explicado, existen dos formatos de vista en dicha ventana: una es un **listado de texto** guiado y el otro está basado en **gráficos**. El utilizar una forma u otra dependerá de uno mismo, según te convenga a tus necesidades o como prefieras visualizar el flujo de ejecución del programa. La vista gráfica es la mostrada por defecto en el desensamblado, anteriormente ya explicamos como cambiar la vista por defecto y también como cambiar de la vista gráfica a la vista listado, con la barra de espaciado.

### 4.1.1.—Vista gráfica de IDA (Graph View)

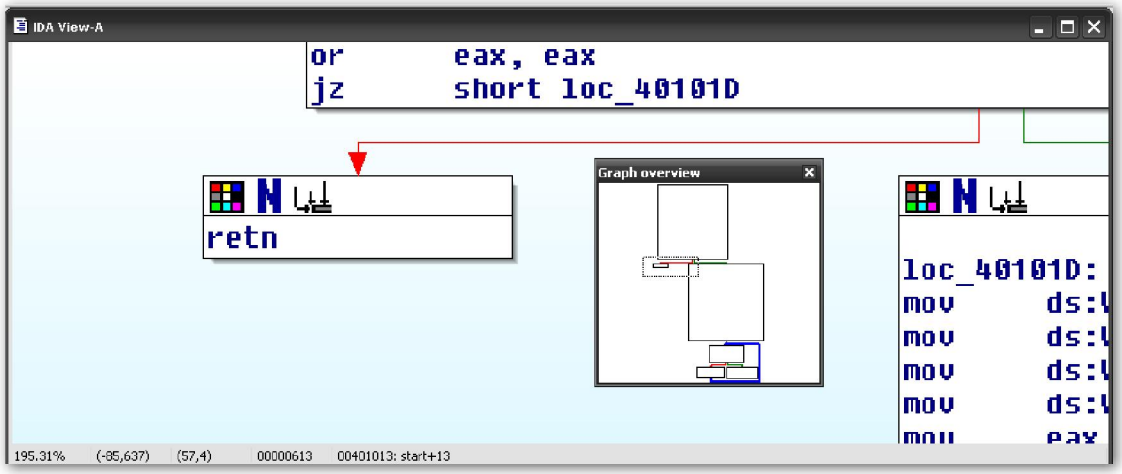
La vista gráfica nos recuerda a los ordinogramas de programas utilizados para visualizar el control del flujo de ejecución de un bloque a otro. Que entendemos por bloque, es la sucesión máxima de instrucciones que se ejecutan sin desviaciones, desde el punto de inicio hasta su punto final. Cada bloque básico por lo tanto tiene un punto de entrada, primera instrucción del bloque, y un punto de salida, última instrucción del bloque. Normalmente la primera instrucción del bloque tiene como objetivo una instrucción de desviación, mientras que normalmente la última instrucción es una desviación.
Veamos los bloques del CRACKME.EXE:



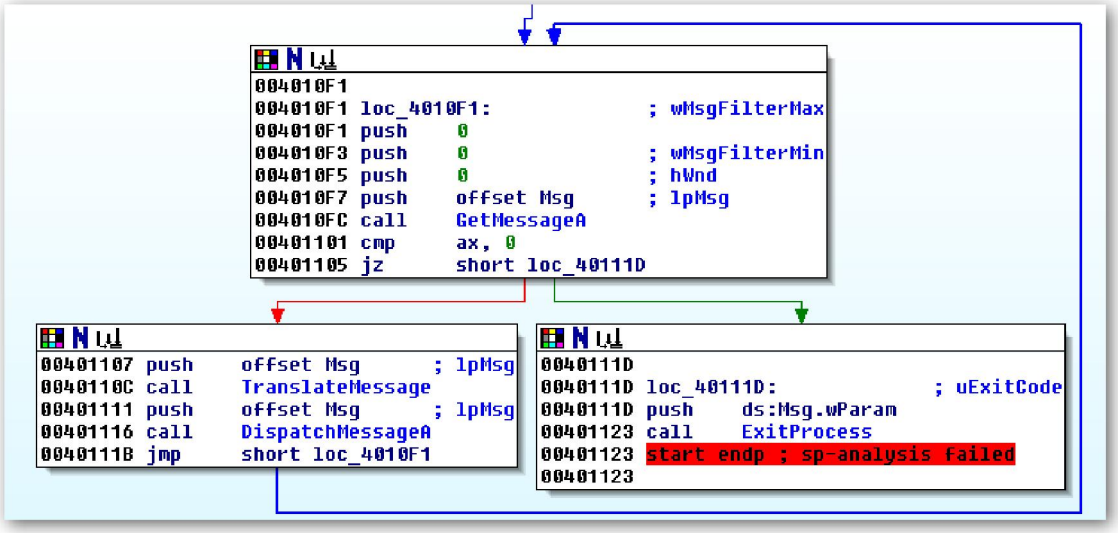
Fijándote en la pantalla, porque digo yo que tendréis el CRACKME.EXE cargado, bueno si no es así fijaros en la figura, observarás que IDA utiliza distintas líneas flechadas y coloreadas para distinguir los distintos flujos de ejecución entre los bloques de una función. IDA utiliza el término flujo para indicar cómo la ejecución puede continuar desde una instrucción dada hacia otra. Un flujo **normal**, también llamado ordinario, indica por defecto la ejecución secuencial de instrucciones. Un flujo **jump** o de salto indica que la instrucción actual desvía o puede desviar a una ubicación no secuencial. Por último un flujo **call** indica que la instrucción actual llama a una subrutina. Los bloques básicos que terminan con un salto condicional generan dos

posibles flujos dependiendo de la condición a verificarse; si la condición es **Si**, se realiza la desviación y hay que seguir la desviación de la línea flechada por defecto **verde**. Si la condición es **No**, hay que seguir la desviación de la línea flechada por defecto **roja**. Los bloques básicos que terminan con un solo flujo a seguir utilizan una línea flechada por defecto de color **azul** para indicar el bloque próximo a ejecutarse.

En modo gráfico, IDA muestra cada función de una vez. Indicaremos que para los que utilicen ratón con rueda, se puede ampliar el dibujo gráfico utilizando la combinación de **CTRL + rueda**. El control de zoom en el teclado se realiza efectuando **CTRL-+** para ampliar y **CTRL- -** para reducir, las teclas + y – son las del teclado numérico, zoom + en figura abajo. En las funciones grandes y complejas pueden causar, ver la vista gráfica extremadamente saturada haciendo difícil la navegación en ella. En estos casos, la ventana de visión gráfica general, figura abajo, está disponible para proporcionarnos cierta idea de situación en el desensamblado. La visión gráfica general también nos proporciona la vista completa de la estructura en bloques y el rectángulo con perímetro discontinuo nos indica la zona gráfica seleccionada la cual se nos está mostrando en la ventana de desensamblado. Dicho rectángulo puede ser arrastrado a lo largo de toda la región gráfica, reposicionando rápidamente la vista gráfica en la ubicación que deseemos.




Una cosa importante, ya se me pasaba. Cuando utilices la vista gráfica, tal como nos la muestra por defecto, a mí particularmente, me da la sensación de que me faltan cosas como la información del número de dirección virtual en el desensamblado. La razón de que IDA opte por ocultar muchas de estas informaciones tradicionales en otros desensambladores, es minimizar el espacio requerido para mostrar cada bloque. Si queremos que nos muestre dicha información sólo tenemos que realizar la acción siguiente **Options > General > Disassembly** y te proporcionará una serie de opciones para añadir o quitar. En nuestro caso habilitaremos la línea que pone **line prefixes**, transformándonos la vista gráfica de la siguiente forma, figuras abajo.

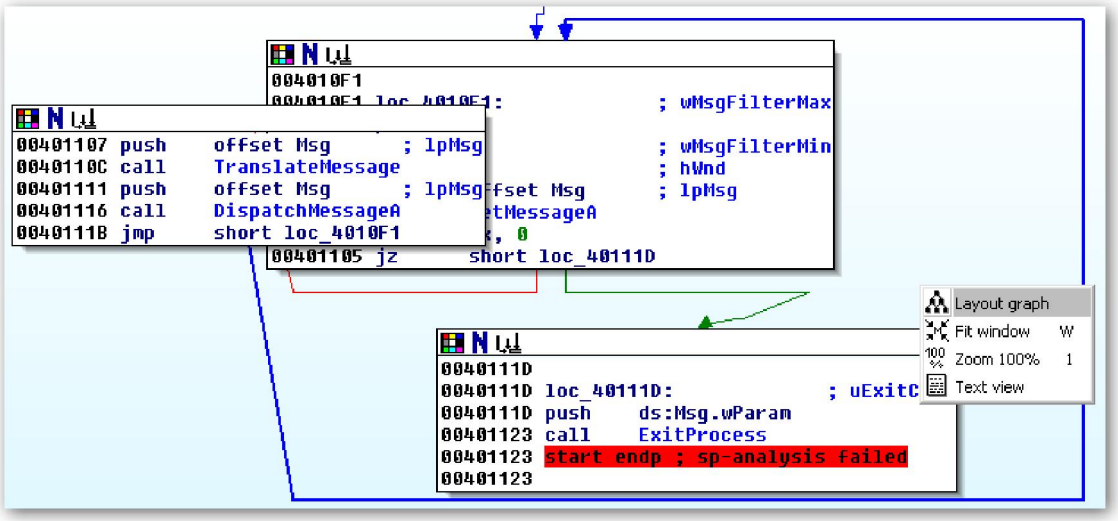



Esto ya me gusta más, uno se ha acostumbrado al Sr. Olly, y pasa lo que pasa.

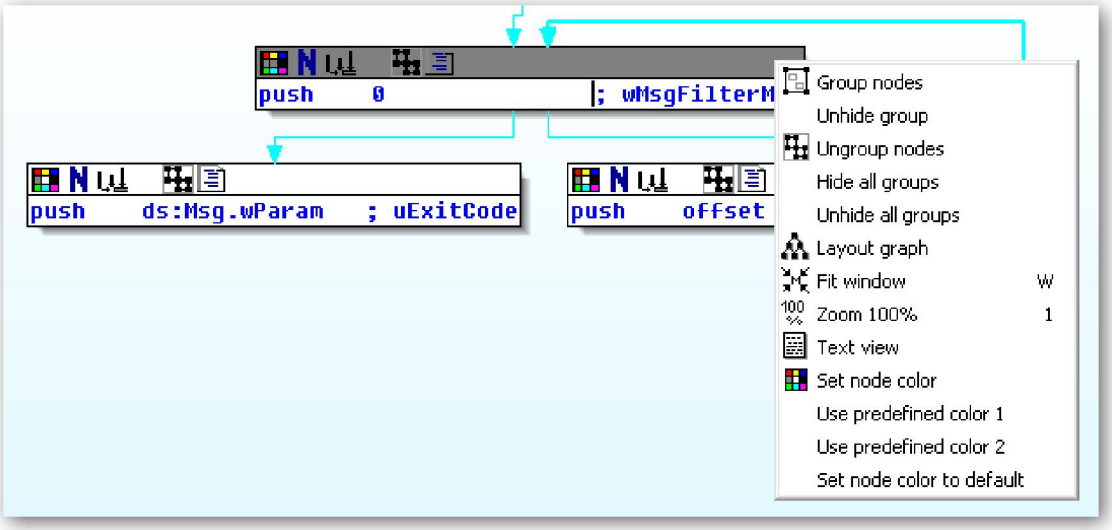
Bien sigamos. En la vista gráfica también disponemos de varias formas con que manipular la vista de datos a nuestra conveniencia:

**Vista panorámica**
Además de utilizar la ventana de visión gráfica general para reposicionar rápidamente la gráfica, también puedes realizar lo mismo haciendo **doble click** en el fondo de la vista gráfica de la ventana de desensamblado y arrastrándola donde te interese.

**Reagrupar o mover bloques**
Los bloques individuales se pueden mover dentro de la vista gráfica, **haciendo click en la barra** de dicho bloque y arrastrándolo a la posición donde desees. Al mover el bloque lo único que reforma IDA son las líneas de flujo, asimismo puedes dirigir la línea de flujo a una nueva posición clickando dos veces más la tecla mayúscula. Si quieres recuperar el aspecto anterior de la gráfica, deberás realizar un click derecho y elegir Layout Graph. Observa la figura es la misma que la anterior pero movida.

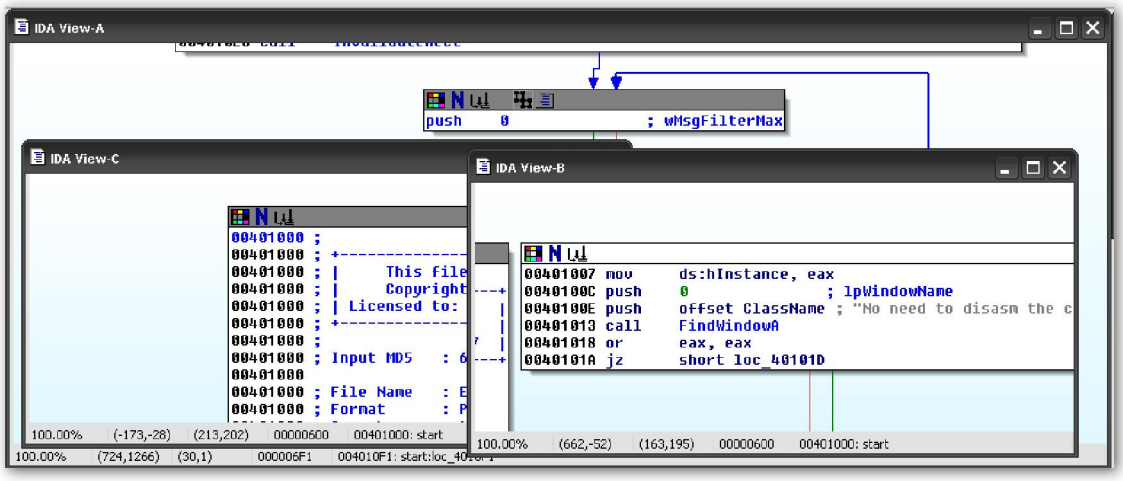


**Agrupar bloques**
También puedes agrupar bloques, individualmente o junto con otros bloques, y reagrupar para reducir la confusión en la vista. Reagrupar bloques es una técnica particularmente útil para mantener el rastro de los bloques ya analizados. Puedes reagrupar cualquier bloque haciendo **click derecho en la barra de título del bloque** y seleccionando **Group Nodes**. Es lógico decir que se haces **Ungroup nodes** los desagrupas.




La mista figura de ejemplo anterior agrupada.

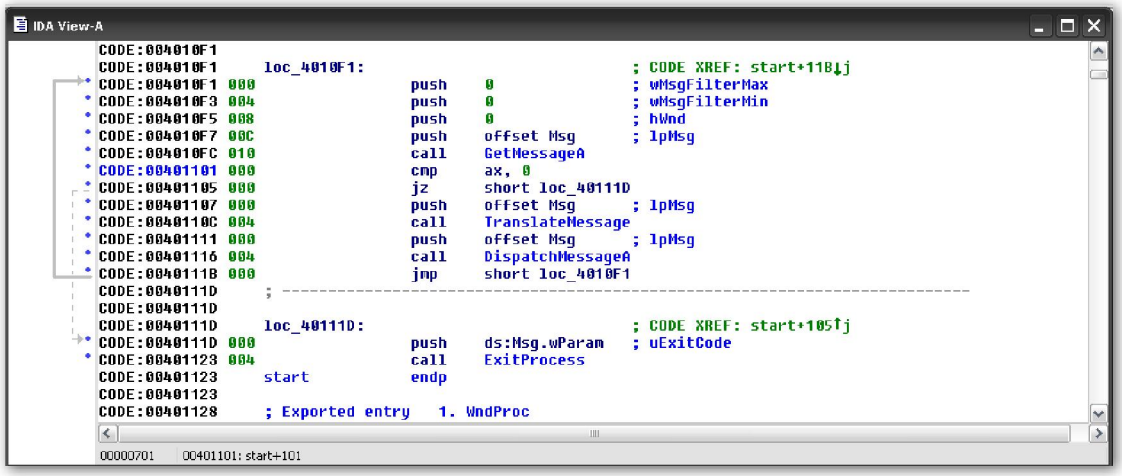
**Crear ventanas de desensamblado adicionales**
Si alguna vez te interesa poder mirar el gráfico de dos funciones distintas simultáneamente, lo único que necesitas es abrir otra ventana de desensamblado realizando la acción **Views > Open Subviews > Disassembly**. La primera ventana de desensamblado abierta tiene el título **IDA View-A**. En consecuencia la siguiente ventana abierta tendrá el título **IDA View-B, IDA View-C**, y sucesivamente. Cada desensamblado es independiente del otro y se puede realizar perfectamente la vista gráfica en una y la vista del listado en otra o ver tres gráficos distintos en tres ventanas distintas.



Tengamos en cuenta que la manipulación de la vista de desensamblado no acaba aquí, tenemos muchas más opciones que las veremos más adelante. No obstante existe mucha más información, fácil, para manipular la vista gráfica en el **IDA help file > Windows > Graph view**, aunque casi toda se ha presentado aquí.

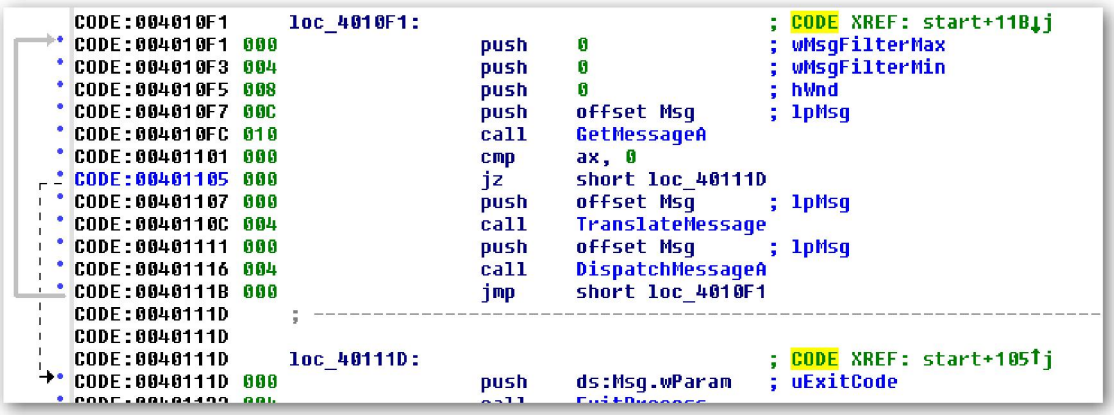
### 4.1.2.—Vista de listado IDA

La ventana de desensamblado en la forma listado, es la vista que normalmente es utilizada para ver y manipular los desensamblados generados por IDA. La vista listado muestra el listado completo del desensamblado de un programa, en contraposición de la vista gráfica la cual muestra función por función, y proporciona el único medio que tendremos para observar las regiones de datos de un binario. Toda la información mostrada en la vista gráfica, está disponible de una forma u otra en el listado.



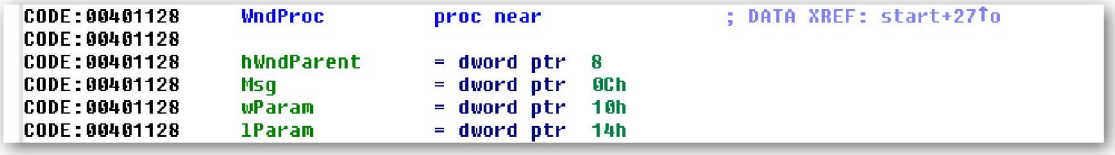

En la figura arriba, vemos el listado de las tres funciones utilizadas en las figuras anteriores de la vista gráfica. El desensamblado es mostrado de forma lineal, con la dirección virtual mostrada. La dirección virtual normalmente se muestra con el siguiente formato **[NOMBRE SECCION] : [DIRECCION VIRTUAL]**, en este caso **CODE : 004010F1.**

La parte izquierda de la vista, figura abajo, recibe el nombre de **arrows windows** y es utilizada para mostrar el flujo de ejecución no lineal dentro de una función.



Las flechas no discontinuas representan saltos incondicionales, mientras que las flechas discontinuas representan saltos condicionales. Cuando un salto, condicional o incondicional, transfiere el control a una dirección del programa cercana, la línea continua o discontinua queda resaltada, en la figura arriba vemos resaltada la del salto condicional. Por otra parte si nos encontramos con una línea no discontinua pero con el flujo de ejecución hacia arriba, ver en la figura **0040111B a 004010F1**, nos indica que estamos ante un bucle.

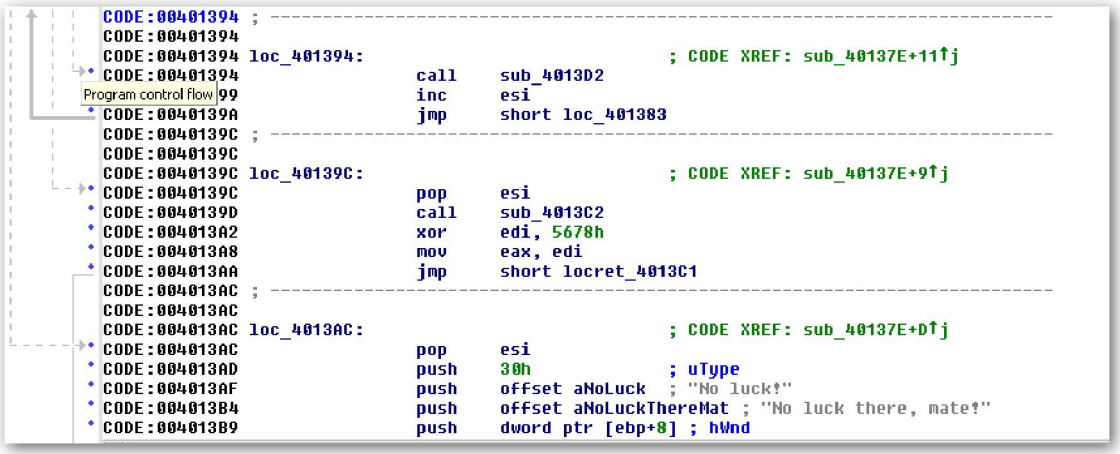
Las declaraciones del tipo, figura abajo, presentes también en la vista gráfica, representan la mejor estimación posible que ha podido realizar IDA del **stack frame** (estructura de pila).




Como sabemos. El **stack frame**, es un bloque de memoria, distribuida en la **pila** de un programa en ejecución, esta contiene los parámetros de inicio que serán pasados a una función y también las variables locales declaradas en esa función. El **stack frame** es distribuido en la entrada de una función y liberado a la salida de esta. De momento hasta aquí, más adelante estudiaremos detalladamente esta estructura.

Los comentarios, cadenas iniciadas con **( ; )** son comentarios, son referencias cruzadas **(XREF)**. En este caso, figura abajo, podemos ver referencias cruzadas **CODE**, opuestas a las referencias cruzadas **DATA**. Estas indican otra instrucción del programa, la mostrada en el comentario, referenciada a esta ubicación. Las "**Cross-references**" también las estudiaremos más adelante.

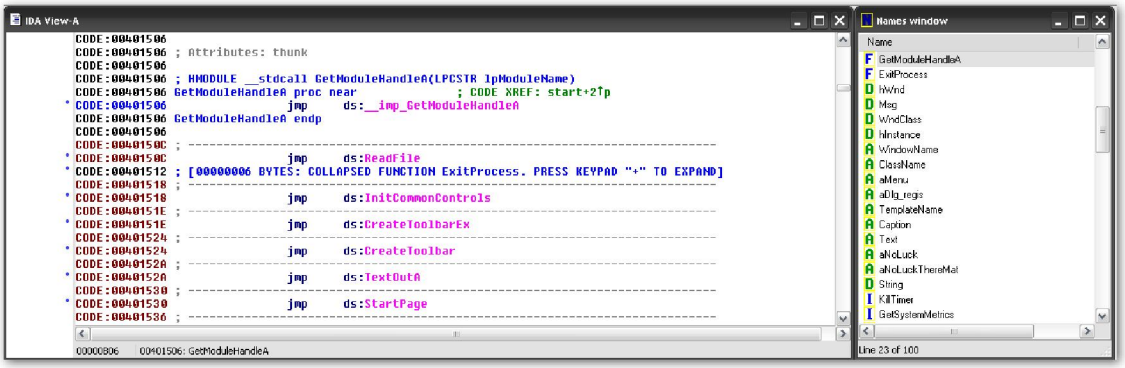
Bien, creo que para no liarnos más de momento, no hablaremos más del listado, hasta que hablemos de cómo limpiarlo y realizar anotaciones en él.




## 4.1.3.—La ventana Names

La ventana **Names**, figura abajo, proporciona un registro listado de todos los nombres globales en un binario. Un **name** (nombre), no es más que una descripción simbólica de una dirección virtual tomada de un programa. IDA inicialmente obtiene la lista de

nombres de la tabla de símbolos (**symbol-table**) y del análisis de la firma (**signature**) del archivo durante su cargado inicial. Los nombres pueden ser clasificados alfabéticamente, por orden de dirección virtual o por ascendente/descendente. La ventana **Names** se utiliza para navegar rápidamente a ubicaciones conocidas en el listado del programa. Si realizamos doble click en cualquier entrada de la ventana Names, nos mostrará en la ventana de desensamblado el nombre seleccionado.




Los **Names** tienen distintos colores y letras de código. El esquema del código se resume de esta forma:

**F** Indica una función utilizada permanentemente (regular). Estas funciones IDA no las reconoce como funciones de librerías.

**L** Indica una función de librería. IDA reconoce las funciones de librerías a través de los algoritmos de coincidencia de firma. Si no existe una firma para una función de librería tomada, la función será etiquetada como función regular.

**I** Muestra un nombre importado, normalmente un nombre de función importada desde una librería compartida. La diferencia entre esta y una librería de función es que para la importada no existe código, mientras que para una función de librería el cuerpo de esta estará presente en el desensamblado.

**C** Indica código nombrado. Estas son ubicaciones de instrucciones del programa con nombre las cuales IDA no puede considerar parte de ninguna función. Este caso se produce cuando IDA halla un nombre en la tabla de símbolos pero que nunca tiene un **call** en su ubicación en el programa.

**D** Indica datos. Ubicaciones de datos nombrados, normalmente representando variables globales.

**A** Indica datos con caracteres ASCII. Ubicaciones referenciadas de datos las cuales contienen cuatro o más caracteres ASCII terminados con un byte nulo.

Para finalizar diremos, que cuando nos desplazamos por los desensamblados, podemos ver que existen muchas ubicaciones las cuales no están listadas en la ventana **Names.** En el proceso de desensamblado que realiza IDA, se generan los nombres de cualquier cosa referenciada como código, una desviación o un call objetivo, cualquier dato leído, escrito o cualquier dirección tomada. Si una ubicación es nombrada en la tabla de símbolos del programa, IDA adopta su nombre de la tabla de símbolos. Si una ubicación de un programa no tiene entrada en dicha tabla, IDA generará un nombre por defecto que utilizará en el desensamblado. Cuando elije un nombre para la ubicación, la dirección virtual de la ubicación se combina con un prefijo el cual indica el tipo de la ubicación nombrada. Incorporar la dirección virtual al nombre generado nos asegura

que éste será único en el desensamblado ya que ninguna ubicación puede compartir la misma dirección virtual. Los nombres generados de los siguientes tipos no son mostrados en la ventana **Names**:

**sub_xxxxxx** . Una subrutina a la dirección xxxxxx

**loc_xxxxxx** . Una ubicación de instrucción en la dirección xxxxxx

**byte_xxxxxx** . 8-bit de datos en la ubicación xxxxxx.

**word_xxxxxx**. 16-bit de datos en la ubcación xxxxxx

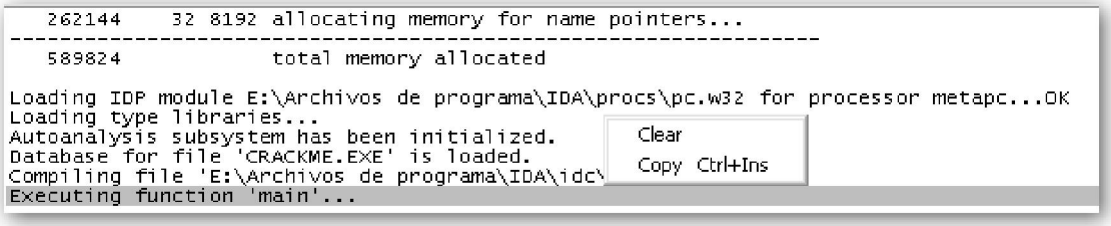
**dword_xxxxxx**. 32-bit de datos en la ubicación xxxxxx

**unk_xxxxxx** . Datos de tamaño desconocidos en la ubicación xxxxxx

AL ir avanzando en nuestro entendimiento de IDA, estudiaremos los algoritmos que aplica IDA para escoger los nombres de los datos ubicados en el programa.

## 4.1.4.—Ventana de mensajes

La ventana de mensajes situada en la parte baja del escritorio de IDA se muestra por defecto cuando se carga un archivo. La ventana de mensajes es la salida de consola de IDA en donde podemos observar la información de todo lo que está ejecutando IDA. Por ejemplo cuando un archivo es abierto por primera vez, los mensajes generados nos indican la fase de análisis, y cada acción que realiza se va creando una nueva base de datos. Cuando trabajes con una base de datos, la ventana de mensajes te mostrará el estatus de las distintas operaciones que se ejecutan. Los contenidos de dicha ventana se pueden copiar en el portapapeles del sistema o limpiar la ventana haciendo click izquierdo en la ventana y escogiendo la operación apropiada. Esta ventana normalmente es fundamental para ver las salidas proporcionadas por los **scripts y plugins** programados para IDA.

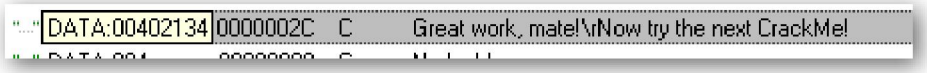



## 4.1.5.—La ventana Strings

La ventana **Strings** de IDA es el equivalente a la utilidad strings de otros desensambladores, recuerdo ahora mi querido W32Dasm. Esta ventana la podemos habilitar ejecutando la acción **View > Open Subviews > Strings**.

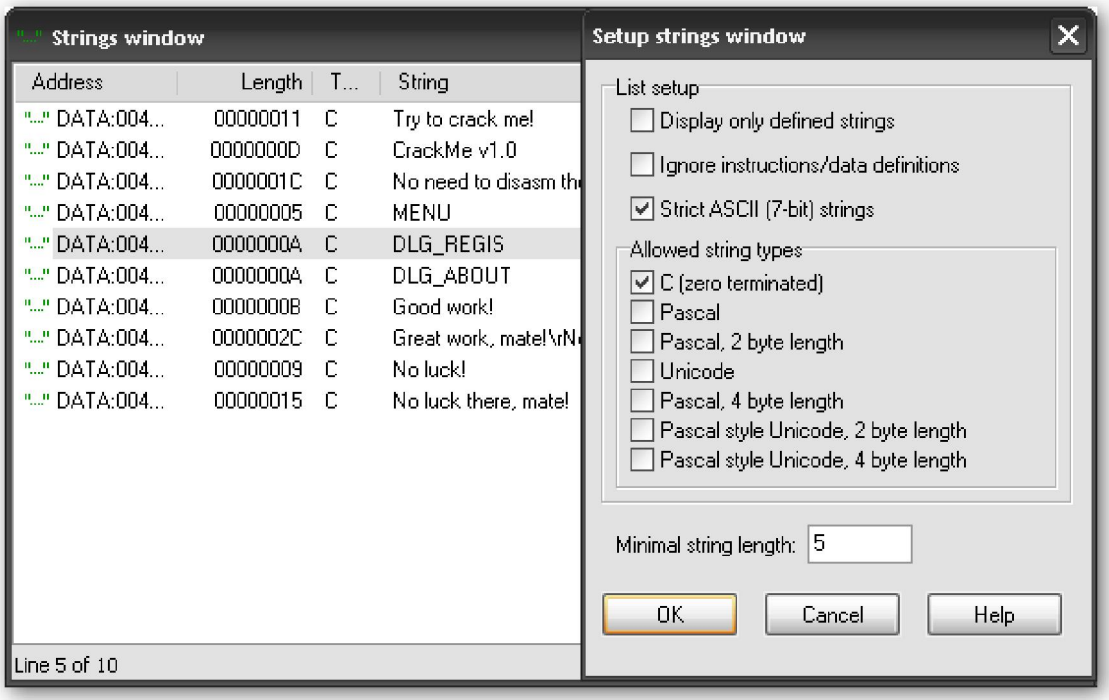
El propósito de dicha ventana es mostrar un listado de las cadenas de caracteres juntamente con su dirección en donde residen, extraídas de un binario. Al igual que la ventana Names, si realizamos un doble click en la cadena elegida esta se mostrará automáticamente en la ventana de desensamblado. Cuando se utiliza conjuntamente con referencias cruzadas, ya lo estudiaremos, la ventana **Strings** proporciona los medios

para situar rápidamente una cadena interesante y ver todas las referencias del programa a esta cadena. Por ejemplo puedes ver la cadena:




La cual está referenciada en la dirección 00402134. Después de tratar todas las ventanas de IDA, veremos como navegar a las ubicaciones referenciadas de las strings con sólo cuatro clicks.

Entender la tarea que realiza la ventana **Strings** es esencial para utilizarla con efectividad. IDA no almacena permanentemente las cadenas extraídas de un binario. Por lo tanto cada vez que esta ventana se abre, la base de datos tiene que ser escaneada o rescaneada para volver a sacar las cadenas. El escaneo de cadenas es realizado de acuerdo con las características habilitadas de la ventana Strings, podemos acceder a estas características haciendo click derecho en dicha ventana y seleccionar **Setup**. Nos mostrará un diálogo, figura abajo. La ventana **Setup Strings** se utiliza para especificar a IDA que tipos de cadenas deberá escanear. Por defecto el tipo de cadena que escanea IDA son cadenas C-style (zero-terminated), 7-bit ASCII, de más de cinco caracteres de longitud.

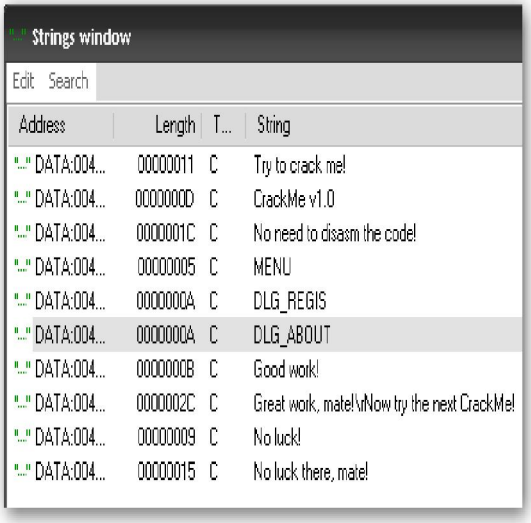


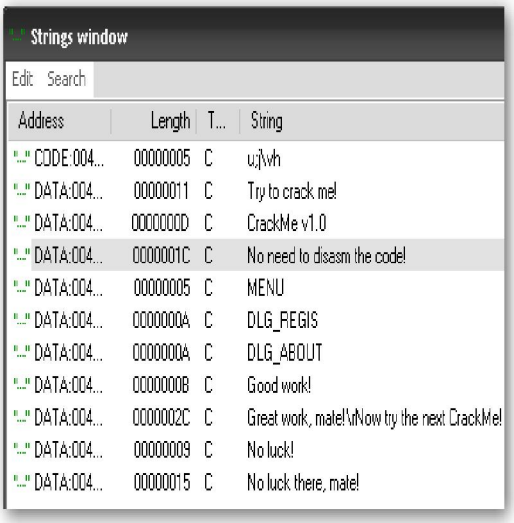

Si te interesa encontrar otro tipo de cadenas, deberás reconfigurar la disposición de características de la ventana y escoger el tipo apropiado a buscar. Por ejemplo los programas de **Windows** utilizan cadenas **Unicode**, mientras que los binarios **Borland Delphi** utilizan para las cadenas **Pascal-style con 2-byte** de longitud. Recuerda, cada vez que cierras la ventana **Setup Strings** haciendo click en **OK**, IDA rescaneará la base de datos para hallar cadenas de acuerdo con las nuevas características. Existen dos opciones que merecen una mención especial:

**Display only defined strings**
Si habilitamos esta opción, la ventana Strings sólo nos mostrará los elementos de cadena de datos nombrados, los que IDA haya creado automáticamente o los creados manualmente por el usuario. Si seleccionamos esta opción, todas las demás serán deshabilitadas y además IDA no escaneará automáticamente el contenido de cadenas.

**Ignore instructions/data definitions**
Esta opción produce el escaneado de cadenas entre las instrucciones y las definiciones de datos existentes. Utilizar esta opción permite dos acciones a IDA, primero ver las cadenas adjuntadas al código de un binario y que por error se hayan convertido en instrucciones y segundo ver cadenas en los datos que pueden estar en un formato distinto al de una cadena, por ejemplo, un conjunto de bytes o enteros. Esta opción generará muchas cadenas basura, las cuales serán sucesiones de cinco o más caracteres ASCII unas legibles y otras no.

La moraleja es que IDA no tiene porque mostrar todas las cadenas de caracteres de un binario si no tenemos el **Setup strings** configurado apropiadamente.




Vemos que con la opción **Ignore instructions/data definitions** en este caso nos ha proporcionado una cadena más, en este caso es ilegible pero pudiera ser alguna cadena interesante y haber quedado olvidada.

Performance Bigundill@