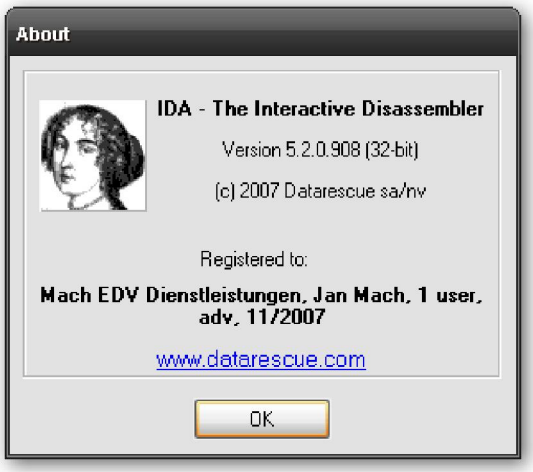
Conociendo IDA de “cabo a rabo”

3.1.-- Iniciándonos con IDA

Es el momento de utilizar IDA. La idea de este escrito es estudiar distintas características de IDA y cómo pueden realizarse de la mejor forma para nuestras necesidades de ingeniería inversa. Ahora empezaremos a explicar las opciones por defecto que tiene IDA cuando se ejecuta y explicaremos que va sucediendo cuando abrimos un archivo binario para analizarlo. Para finalizar, daremos una visión general de la pantalla de usuario para irnos preparando para los conceptos siguientes.

3.1.1.-- Ejecutar IDA

Cuando ejecutas IDA, se nos presenta brevemente una pantalla donde muestra un resumen de la licencia. Una vez desaparece, IDA muestra otro diálogo ofreciéndonos tres posibles acciones de entorno, veamos:

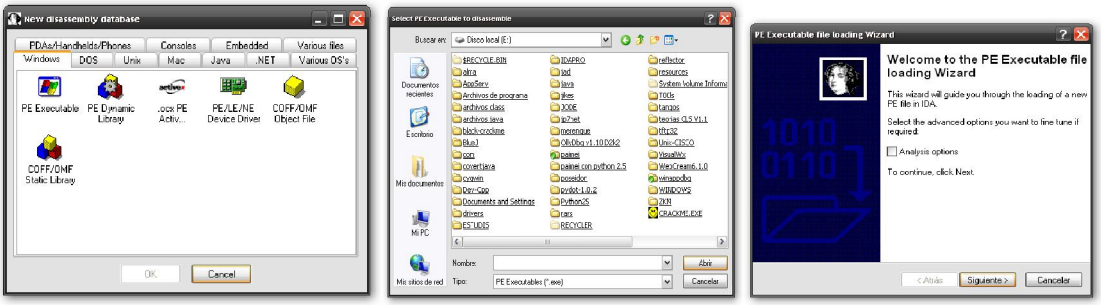


Si prefieres no ver los mensajes de bienvenida, selecciona el checkbox que pone **Don’t display this dialog box again**. Si lo seleccionas la próxima vez que ejecutes IDA, tendrá el mismo efecto que si hubieses pulsado el botón **Go**, e irás directamente a un espacio de trabajo IDA vacío. Si quieres ver otra vez los mensajes, necesitarás editar un archivo clave de registro y habilitar el valor **1** en **DisplayWelcome**.

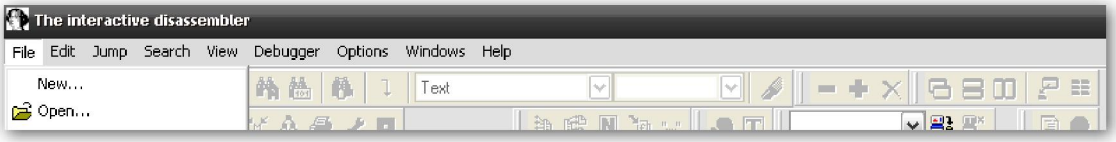
Nota: Cuando IDA se instala en Windows, se crea un archivo clave de registro en mi caso : **HKEY_CURRENT_USER\Software\DataRescue\IDA**. Algunas opciones del mismo IDA pueden ser configuradas y son guardadas en esta clave de registro. Mientras que en otras plataformas los guarda en un archivo de datos binarios **($HOME/.idapro/ida.cfd)** el cual no es editable fácilmente.

Cada una de las opciones propuestas ofrece una forma distinta de proceder de IDA en el escritorio. Las tres opciones realizan lo siguiente:

New
Eligiendo la opción **New** ejecutas un proceso automático el cual te guía para poder seleccionar un nuevo archivo a analizar. Los diálogos los podemos ver en las figuras. En dicho diálogo podemos inicialmente identificar el tipo de archivo que intentamos abrir. Una vez especificado el tipo, se nos proporciona un diálogo **Abrir Archivo** utilizado para seleccionar el archivo a analizar. Finalmente uno o más diálogos iniciales son mostrados para permitirnos especificar las opciones de análisis antes de que el archivo sea cargado, analizado y mostrado.



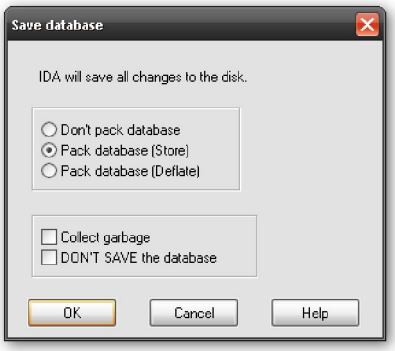
Un problema del proceso automático de **New** es que el usuario tiene que saber qué tipo de archivo se va a abrir, lo cual puede saberse o no. Más adelante hablaremos sobre la utilidad **file** la cual es una de las herramientas disponibles para identificar el tipo de archivo. Si no sabemos el tipo de archivo es mejor utilizar un método distinto de cargar el archivo en IDA, que es la opción **Go** que veremos seguidamente. El botón **New** corresponde a la acción **File > New**.



Go
El boton **Go** termina el proceso de carga y provoca iniciar IDA con un área de trabajo vacía. Si ahora quisiéramos abrir un archivo, puedes arrastrar y soltar un archivo binario en el área de trabajo de IDA o puedes utilizar la opción del menú **Open**. La acción **File>New** abre el diálogo descrito antes. La acción **File>Open** sobrepasa el proceso automático de **New** y muestra un diálogo para abrir dicho archivo. Por defecto IDA utiliza un filtro de extensiones conocidas para limitar la vista del diálogo. Asegúrate de modificar el filtro eligiendo "**Todos los archivos**" con lo cual te mostrará el archivo que te interese abrir. Cuando abras el archivo de esta forma, IDA intentará automáticamente identificar y seleccionar el tipo de archivo; sin embargo deberás poner atención al diálogo de carga para ver que los cargadores han seleccionado el archivo a procesar.

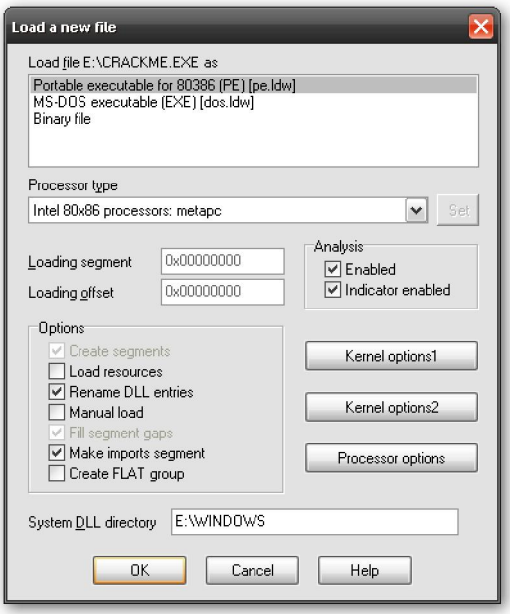
Previous
Deberás utilizar el botón **Previous** cuando quieras abrir uno de los archivos de la lista de **recent files** debajo del botón Previous. La lista de archivos utilizados recientemente se rellenará con los valores de la subclave **History** de la clave de registro IDA en Windows. El largo máximo de la lista inicialmente está habilitada en 10 entradas, pero puedes colocar el límite hasta 100. Esto lo haremos editando la entrada apropiada en el archivo **idagui.cfg** o **idatui.cfg**, más adelante estudiaremos configuraciones. Si

queremos utilizar la lista de historial de archivos es muy conveniente utilizar la opción, resumir el trabajo utilizado en archivos base de datos "**Pack database (Store)**".

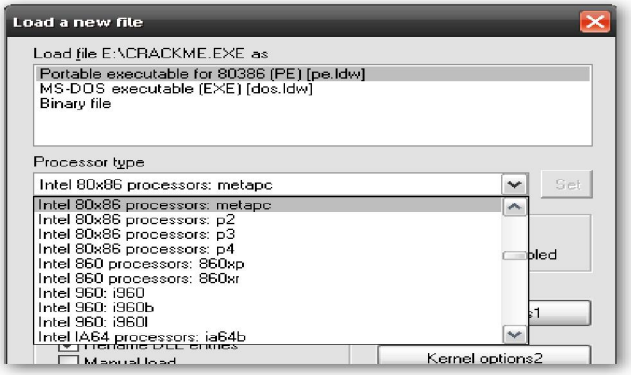


3.1.2.-- Cargar un archivo en IDA

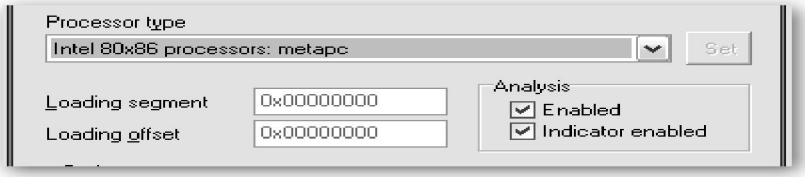
Cuando escogemos abrir un nuevo archivo utilizando la orden **File > Open**, se te presentará un diálogo. En este, IDA genera una lista de los tipos de archivo potenciales en la parte superior del diálogo, ver figura. Esta lista representa los mejores cargadores elegidos por IDA según el archivo seleccionado. Esta lista se crea ejecutando cada uno de los cargadores del directorio **loaders** eligiendo cada uno factible para cargar el archivo, esta acción reconoce el nuevo archivo. Observa en el diálogo que ambos cargadores el Windows **PE loader (pe.ldw)** y el **MS-DOS EXE (dos.ldw)** reconocen al archivo seleccionado. Si estamos familiarizados con el formato PE no nos sorprenderá esto, ya que el formato PE es la forma extendida del formato MS-DOS EXE. La última entrada de la lista, **Binary file** siempre está presente, es la opción por defecto de IDA para cargar los archivos que no reconoce y esta acción nos proporciona un nivel de carga bajo (soft) para cargar cualquier archivo.



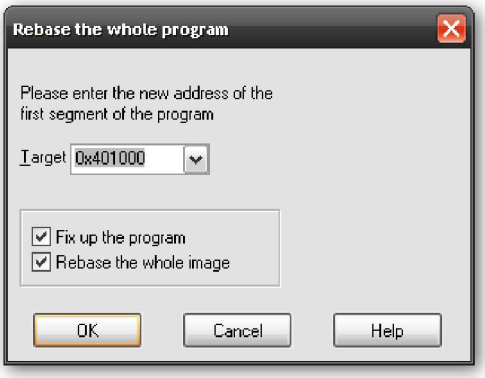
En algunas ocasiones el **Binary File** será la única entrada que aparecerá en la lista de cargadores. En estos casos, el mensaje implica que ninguno de los cargadores reconoce el archivo escogido. Si optas en continuar el proceso de carga, asegúrate de escoger el tipo de procesador de acuerdo al contenido del archivo.

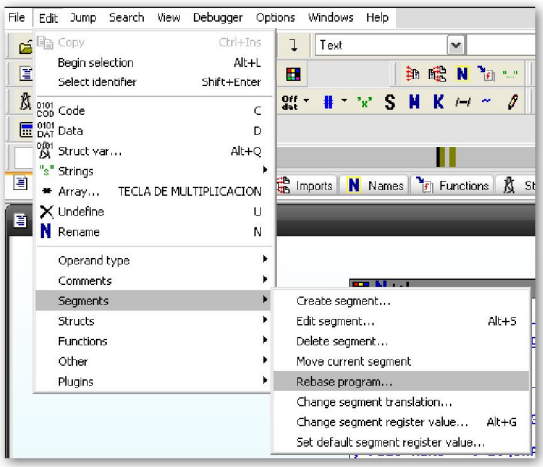


El **Processor Type** despliega un menú que nos permite especificar cualquier módulo de procesador, desde el directorio **procs**, el cual se utilizará durante el proceso de desensamblado. En la mayoría de los casos, IDA elegirá el procesador más apropiado basándose en la información leída de los encabezados (headers) del archivo ejecutable. Cuando IDA no pueda determinar el tipo de procesador asociado al archivo abierto, necesitarás seleccionar manualmente el tipo de procesador para continuar la operación de cargado.



Los campos **Loading segment** y **Loading Offset** se activarán solamente cuando se elija el formato **Binary File** conjuntamente con el procesador de la **familia x86**. Cuando el cargador binario no pueda extraer la información del esquema de memoria, los valores de **segment** y **offset** introducidos se combinaran para formar la dirección base del contenido cargado del archivo. Si te olvidas de especificar la dirección base durante el cargado inicial, esta se puede modificar en cualquier momento utilizando la acción **Edit > Segments > Rebase Program**.





Los botones **Kernel Options** proporcionan acceso a las opciones de configuración específica del análisis de desensamblado las cuales IDA utiliza para mejorar el proceso descendiente recursivo. En la gran mayoría de los casos, las opciones por defecto proporcionan el mejor desensamblado posible. Los archivos de ayuda de IDA proporcionan mayor información sobre las opciones posibles de kernel.

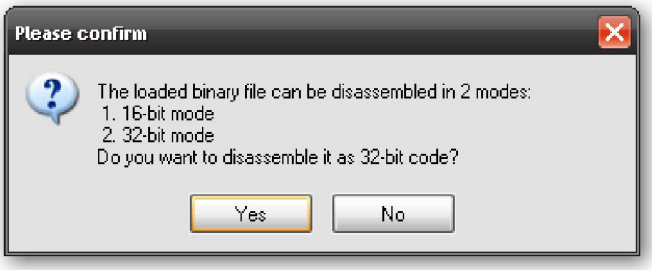
El botón **Processor Options** proporciona acceso a las opciones de configuración aplicables al módulo de procesador seleccionado. Sin embargo, estas no están disponibles para todos los módulos de procesador. La ayuda de estas opciones depende del módulo de procesador elegido y del autor de la programación de cada módulo.

Los restantes **checkbox** se utilizan para conseguir un control más ajustado sobre el proceso de carga del archivo. Cada una de estas opciones está explicada en el archivo de ayuda de IDA. Dichas opciones no son aplicables ni a todos los archivos ni en todos los casos, puede depender de las selecciones por defecto. En los casos específicos que se tendrán que modificar estas opciones las estudiaremos más adelante.

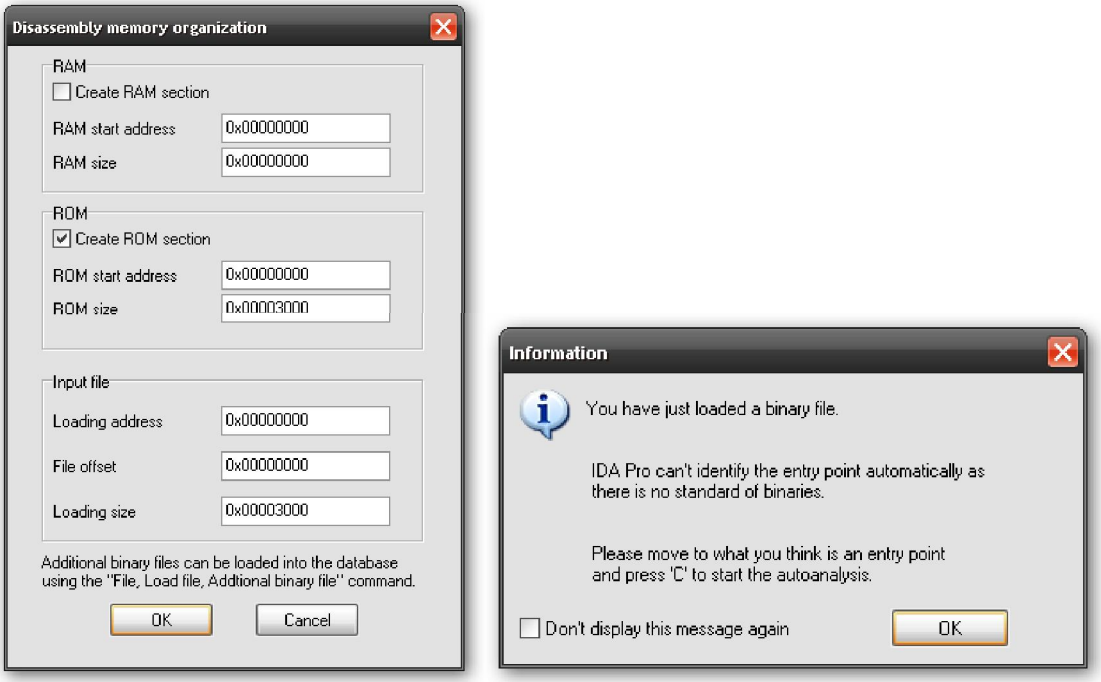
3.1.3.-- Utilizar el cargador Binary file

Cuando optes en utilizar el cargador binario, **Binary file** necesitarás realizar alguna preparación extra al trabajo de procesamiento. Sin la información de los encabezados del archivo para guiar el proceso de análisis, tendrás que realizar algún paso extra y ejecutar trabajos manualmente, que los cargadores realizan automáticamente. Situaciones en las que es necesario utilizar el cargador binario son análisis de imágenes ROM, explotar datos empaquetados que pueden haberse extraído de una captura de red o archivos de registro.

Cuando el módulo de procesador x86 se combina con el cargador binario, se mostrará el diálogo siguiente. Sin conocerse los encabezados del archivo para ayudar a IDA, se te informará que se puede desensamblar el código en modo 16-bit o en modo 32-bit. Otros módulos de procesador que pueden distinguir entre modo 16-bit y 32-bit son ARM y MIPS.



Los archivos binarios no contienen ninguna información concerniente al esquema de memoria, es decir; a menos ninguna información que IDA sepa reconocer. Cuando se escoge un procesador tipo x86, la información de la dirección base debe especificarse en los campos del diálogo de cargador **Loading Segment** y **Loading Offset**, como ya hemos dicho anteriormente. Para todos los tipos de procesador restantes, IDA muestra un diálogo de esquema de memoria. Para que a tu conveniencia puedas crear una sección RAM, una sección ROM o ambas y designar el rango de dirección de cada una, ver figura siguiente. Las opciones en la etiqueta **Input File** se utilizan para especificar qué porción del archivo de entrada, por defecto el archivo entero, deberá cargarse y qué direcciones de contenido del archivo deberán mapearse.



El último paso de la carga de un binario es un recordatorio de cómo tienes que realizar bien el trabajo. El mensaje informa del hecho de que IDA no tiene ninguna información disponible para ayudarte a distinguir los **bytes de código** de los **bytes de datos** en el archivo binario. Además te recuerda que indiques a IDA la dirección en el código que tú creas como punto de entrada y pulses **'C'** para iniciar el análisis desde ese punto. Para los archivos binarios, IDA no ejecuta ningún desensamblado hasta que no identifique al menos un byte como código.

3.2.-- Archivos de base de datos de IDA.

Cuando haz finalizado con las opciones de cargado y pulsas **OK** para cerrar el diálogo, o **Finish** si has utilizado el **New File** guiado, se realiza el cargado real del archivo. En este punto, IDA carga el ejecutable seleccionado en memoria y analiza las partes relevantes. Esto da como resultado la creación de una base de datos de IDA cuyos componentes son guardados en cuatro archivos, cada uno con el nombre base del ejecutable seleccionado y con las siguientes extensiones **.id0, .id1, .nam, .til**. El archivo **.id0** tiene el contenido del árbol de la base de datos, mientras que **.id1** contiene las banderas que describen cada byte del programa. El archivo **.nam** contiene el índice de la información relacionada con las localizaciones nombradas las cuales se muestran en la ventana de IDA **Names**, la estudiaremos también. Finalmente el archivo **.til** es utilizado para guardar la información concerniente a las definiciones de tipo local específicas a la base de datos dada. Los formatos de cada uno de estos archivos son propiedad de IDA, con lo cual no son fácilmente editables fuera de su entorno.

Por conveniencia, estos cuatro archivos son archivados y opcionalmente comprimidos, en un solo archivo **IDB** siempre que cierres tu proyecto actual adecuadamente. Cuando alguien se refiere a una base de datos IDA, normalmente se están refiriendo a un archivo **IDB**. Un archivo de base de datos no comprimido ocupa normalmente diez veces el tamaño del archivo binario entrado. Cuando se cierra correctamente la base de datos, nunca verás los archivos **.id0, .id1, .nam o .til** en tus directorios de trabajo. La

presencia de estos indica que alguna base de datos no ha sido cerrada correctamente. Esto puede ocurrir cuando IDA falla y la base de datos es corrompida.

3.3.-- Advertencias del cargador

Una vez el cargador ha iniciado el análisis de un archivo, puede encontrarse con circunstancias que requieran entradas adicionales del usuario para completar el proceso de carga. Un ejemplo de esto, ocurre con archivos **PE** que han sido creados con la información del depurador **PDB**. Para que IDA lo determine debe existir un archivo **Program Database (PDB)**, por lo tanto se te preguntará si quieres que IDA localice y procese el correspondiente archivo **PDB** mostrándote este mensaje:

> **IDA Pro has determined that the input file was linked with debug information. Do you want to look for the corresponding PDB file at the local symbol store and the Microsoft Symbol Server?**

Otro ejemplo de generación de un mensaje de información, ocurre con programas ofuscados como puede ser malware. Las técnicas de ofuscación eliminan las especificaciones de formato del archivo, lo cual puede causar problemas al cargador que espera un archivo bien estructurado. Sabiendo esto, el cargador PE ejecuta verificaciones de la tabla de importación y si la tabla de importación no aparece con el formato convenido, IDA mostrará el siguiente mensaje:

> **The imports segment seems to be destroyed. This MAY mean that the file was packed or otherwise modified in order to make it more difficult to analyze. If you want to see the imports segment in the original form, please reload it with the 'make imports section' checkbox cleared.**

Ejemplos de estos errores y como solucionarlos los estudiaremos más adelante.

Es importante comprender que una vez creada una base de datos de un ejecutable, IDA ya no necesita acceder a dicho ejecutable a menos que quieras depurarlo con el depurador incorporado en IDA. Desde un punto de vista de seguridad, esta es una buena característica. Por ejemplo si estamos analizando un malware, puedes pasar la base de datos a otros analizadores sin tener que manipular el ejecutable malicioso. Por otra parte hasta ahora, no se tiene conocimiento de que se haya utilizado una base de datos IDA para realizar ningún ataque malicioso.

En el fondo, IDA no es más que una aplicación de base de datos. Nuevas bases de datos son creadas y rellenadas de forma automática por los archivos ejecutables. Las distintas muestras que nos proporciona IDA son simples vistas de la base de datos las cuales muestran información en un formato utilizado para realizar ingeniería inversa de software. Cualquier modificación realizada por el usuario a la base de datos se reflectará en las vistas y será guardada en la base de datos, pero estos cambios no tendrán efecto en el archivo ejecutable original. La potencia de IDA estriba en las herramientas que contiene para analizar y manipular los datos dentro de la base de datos.