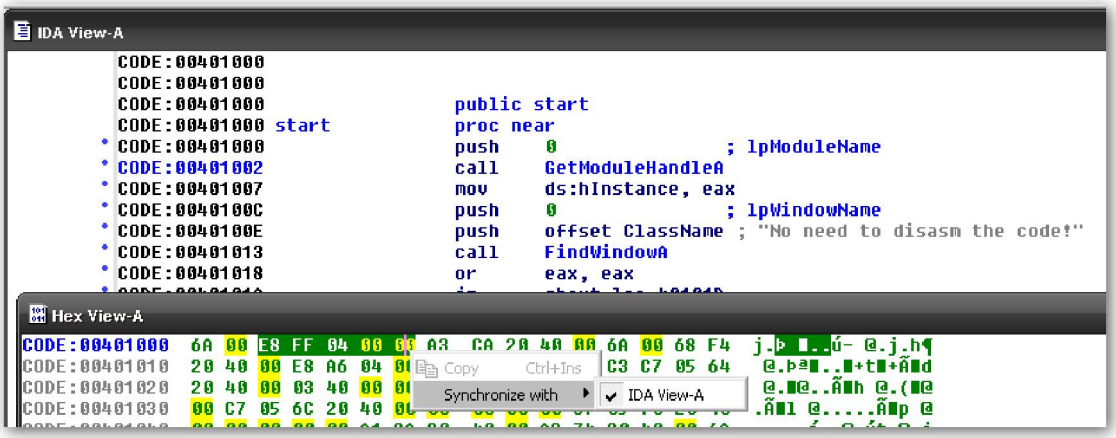
4.2.—Ventanas secundarias de IDA

Además de las ventanas de desensamblado, Names, mensajes y opcionalmente Strings abiertas. Existen otras minimizadas en el escritorio. Cada una se puede abrir rápidamente seleccionando la solapa correspondiente, justo debajo de la barra de navegación.

Dichas ventanas se utilizarán para proporcionarnos vistas alternativas o especializadas de la base de datos. La utilidad de estas dependerá, en principio, de las características del binario analizado y las habilidades que hayamos adquirido con la utilización de IDA. Algunas de estas ventanas son muy especiales con lo cual requerirán una detallada explicación que realizaremos más adelante.

4.2.1.—Ventana Hex View

La ventana **Hex View** nos proporciona un listado del volcado hexadecimal del contenido del programa, en cada línea nos muestra 16 bytes con sus equivalentes ASCII mostrados a un lado. Al igual que la ventana de desensamblado podemos abrir varias ventanas Hex View simultaneamente. La primera será **Hex View-A**, la segunda **Hex View-B** y así sucesivamente. Por defecto la primera ventana Hex View está sincronizada con la primera ventana de desensamblado. Cuando una vista de desensamblado está sincronizada con una vista hexadecimal, colocarte en la dirección virtual de una, automáticamente te colocará en la misma dirección en la otra. Además cuando seleccionamos un elemento en la vista de desensamblado, los bytes correspondientes a dicha selección se mostrarán resaltados en la vista hex en verde. En la figura abajo, podemos ver la vista de desensamblado en la ubicación **00401002**, una instrucción **call,** la cual nos marca en la vista hex los 5 bytes correspondientes a ella.



En la misma figura también podemos observar el menú contextual de la vista, el cual se nos muestra haciendo click derecho en ella. En este menú de contexto puedes especificar con cual, si existe alguna, vista de desensamblado quieres sincronizar la vista hexadecimal. Por otro lado si deseleccionamos la opción de sincronización permitiremos a la vista hexadecimal moverse independientemente de la de desensamblado.

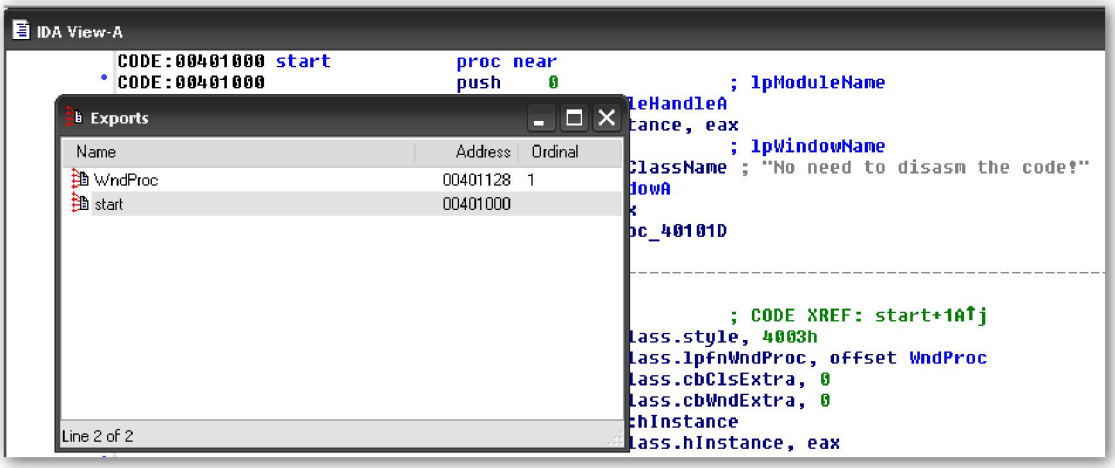
Como hemos dicho al principio esta vista nos proporciona el volcado hexa del contenido de un binario. En algunos casos podrás encontrarte en que dicha vista solamente te proporcionará signos de interrogación. Esta es la forma en que IDA te comunica que no tiene ni idea de los valores que ocupan el rango de la dirección virtual dada. Este es el caso de un programa que contenga una sección **bss**, que normalmente no ocupa ningún espacio dentro del archivo pero es expandida por el cargador para colocar el almacenamiento estático necesario del programa.

Aprovechemos. Una sección **bss** es creada por un compilador para albergar las variables estáticas de un programa no inicializadas. En el momento en que ningún valor inicial es asignado a estas variables, no existe ninguna necesidad para distribuir un espacio para ellas en la imagen del archivo del programa, así pues el tamaño de la sección es anotada en uno de los encabezados del programa. Cuando el programa es ejecutado el cargador distribuye el espacio requerido e inicializa el bloque entero con ceros.

4.2.2.—Ventana Exports

La ventana **Exports** nos proporciona el listado de los **entry points** en un archivo. Estos incluyen al entry point de ejecución del programa, especificado en la sección de encabezados, conjuntamente con todas las funciones y variables que el archivo exporta para el uso de otros archivos. Las funciones exportadas normalmente son proporcionadas por las librerías compartidas, en Windows archivos **DLL**. Las entradas exportadas están listadas por nombre, dirección virtual y si es aplicable por número ordinal. Un número ordinal exportado puede utilizarse en una librería compartida para acceder a una función por un número en vez de por nombre. La utilización de ordinales incrementa la rapidez del proceso de mirar a las direcciones y permite al programador esconder los nombres de funciones. Como hemos dicho la exportación de ordinales se utiliza en las DLL de Windows.

Para los archivos ejecutables, la ventana **Exports** siempre contiene al menos una entrada; el punto de entrada de ejecución del programa. IDA a este punto de entrada le da el nombre de start. Ver figura abajo.

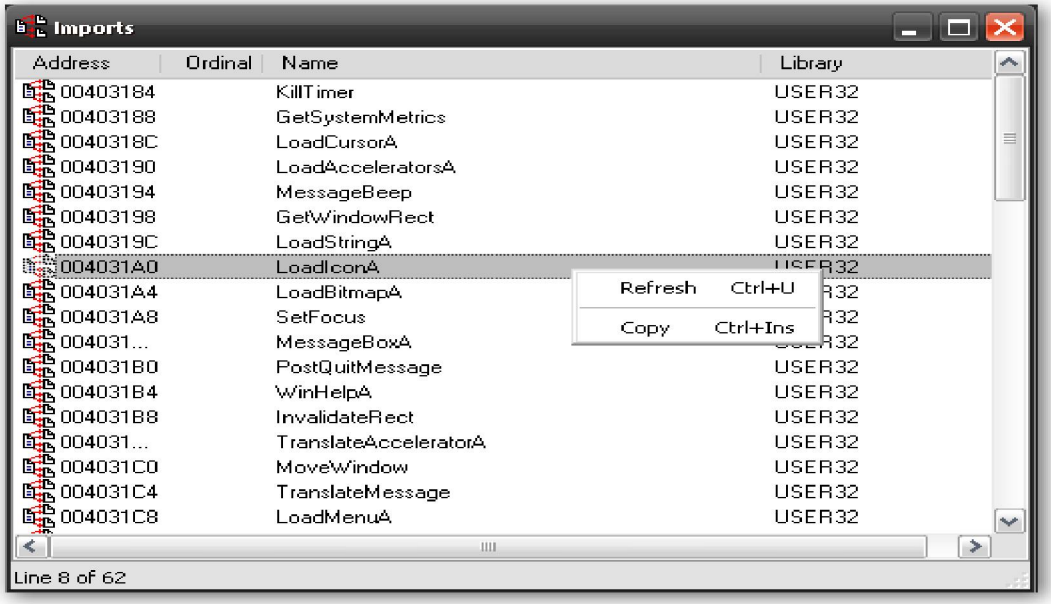


También como cualquier otra ventana de IDA, haciendo doble click en una de las entradas de dicha ventana saltaremos a la ventana de desensamblado en la dirección asociada con la entrada. Apuntaremos que la ventana Exports proporciona distintas

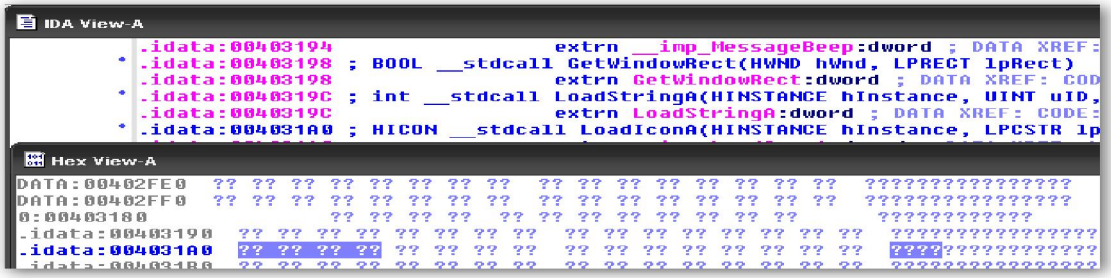
funcionalidades utilizando la línea de ordenes de IDA como **objdump (-T), readelf (-s) y dumpbin (/EXPORTS).**

4.2.3.—Ventana Imports

La ventana **Imports** es la contrapartida de la ventana Exports. Tiene el listado de todas las funciones importadas por el binario mientras es analizado. Esta ventana tiene se importancia solamente cuando un binario utiliza librerías compartidas. En modo estático el binario enlazado a una librería compartida no tiene ninguna dependencia externa y por lo tanto no tiene ninguna importación. Cada entrada de la ventana Imports lista el nombre de un elemento importado, función o dato, y el nombre de la librería que contiene dicho elemento. Para el código una función importada residente en una librería compartida, las direcciones listadas con cada entrada se refieren a la dirección virtual de la entrada asociada a la tabla de importación. Una tabla de importación proporciona espacio para que un cargador almacene direcciones de funciones importadas una vez que las librerías requeridas hayan sido cargadas y las direcciones de esas funciones sean conocidas. Simplemente una tabla de importación tiene en cuenta la dirección para cada una de las funciones importadas. La ventana y ejemplos de entrada ver figura abajo.



Si hacemos doble click en la entrada con dirección **004031A0** nos trasladará a ella en la ventana de desensamblado. El contenido de esta ubicación de memoria en la **hex view** deberia ser **?? ?? ?? ??.** Ya que IDA es una herramienta de análisis estático, y no tiene forma de conocer qué dirección será colocada en esta ubicación de memoria cuando el programa se ejecute.

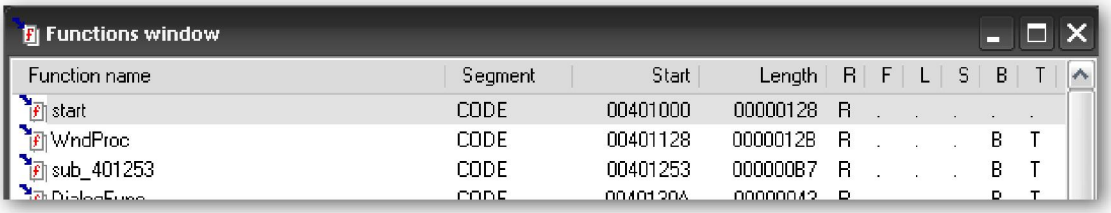


La ventana Imports también ofrece funcionalidades a través de la línea de órdenes de IDA, **objdump (-T), readelf (-s) y dumpbin (/IMPORTS).**

Un apunte importante, a recordar de la ventana **Imports** es; que sólo muestra los símbolos del binario manejados automáticamente por el cargador dinámico. Los símbolos que un binario elija cargar utilizando sus propios mecanismos como puede ser **dlopen/dlsym** o **LoadLibrary/GetProcAddress** no serán listados en la ventana Imports.

4.2.4.—Ventana Functions

La ventana **Functions** se utiliza para listar cualquier función de la base de datos. A diferencia de la ventana **Names**, que no lista las funciones que tengan un nombre autogenerado (**sub_xxxxxx**), la ventana Functions lista todas las funciones que IDA tiene reconocidas en la base de datos. Una entrada de la ventana Functions es como, figura abajo.

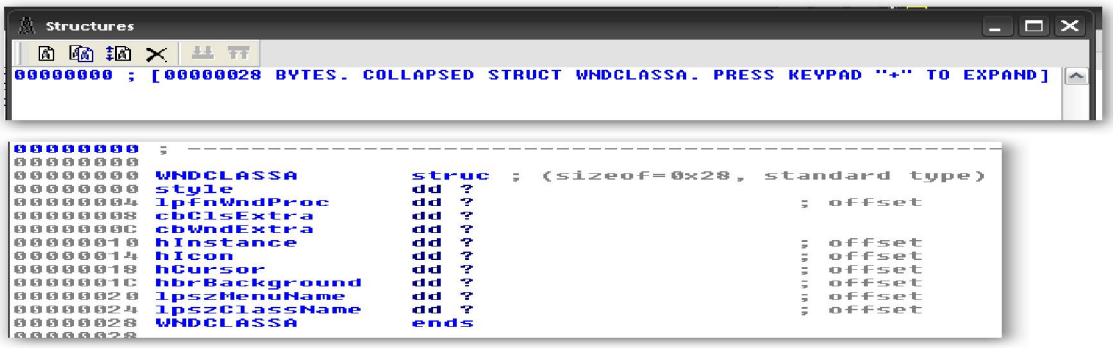


La línea resaltada nos indica que la función **start** se puede encontrar en la sección **CODE,** en la dirección virtual del binario **00401000**, que tiene **296 bytes (128 hexa)** de longitud, que retorna al llamador (**R**), y no tiene ninguna variable local referenciada. Las banderas utilizadas para describir una función, como **R** o **B**, las puedes encontrar en **IDA Help > Functions windows**.

También como las demás ventanas, doble click en una entrada en la ventana Functions produce el salto de la ventana desensamblado en la posición de la función seleccionada.

4.2.5.—Ventana Structures

La ventana **Structures** se utiliza para mostrar el esquema de cualquier estructura compleja de datos, como estructuras C o uniones, que IDA ha determinado en uso dentro del binario. Durante la fase de análisis, IDA consulta una extensa librería de firmas de funciones tipo en un intento de hacer coincidir los tipos de parámetros de función con la memoria usada en el programa. La figura nos muestra una estructura **WNDCLASSA**, que como se indica para expandirla tienes que pulsar + numérico.



Los dos usos principales de la ventana **Structures** son: primero para proporcionar un referencia preparada para el esquema de las estructuras de datos estandares y segundo para proporcionarte los medios para crear tus propias estructuras de datos para utilizarlas como plantillas de esquema de memoria cuando descubres estructuras de datos a medida en un programa. La definición de estructuras y la aplicación de estas en los desensamblados lo trataremos más adelante.

4.2.6.—Ventana Enums

La ventana **Enums** es similar a la ventana Structures . Cuando IDA detecta el uso de dato tipo enumerado (**C enum**), este tipo de dato es listado en la ventana Enum. Puedes hacer más legibles los desensamblados utilizando enumeraciones en lugar de constantes enteras. Al igual que la ventana estructuras, esta nos ofrece facilidades para definir nuestros propios tipos de enumerados los cuales podremos utilizar en nuestro desensamblado.