8.2.— Graficado en IDA

Debido a que las referencias cruzadas relacionan una dirección con otra, es el punto de inicio idóneo para realizar un dibujo gráfico del binario. Circunscribiéndonos a nuestros específicos tipos de referencias cruzadas, podemos trazar distintos gráficos útiles para analizar nuestro binario. Para los que no se acuerden, en los gráficos de IDA, las referencias cruzadas son las líneas flechadas que unen dos puntos. Dependiendo del tipo de gráfico que queramos generar los nodos individuales, los puntos en la gráfica, pueden ser; instrucciones individuales, grupos de instrucciones llamados bloques básicos (**basic blocks**) o funciones enteras. En IDA disponemos de dos capacidades distintas para dibujar una gráfica: Una es la capacidad heredada la cual dibuja la gráfica utilizando una aplicación gráfica no integrada y la otra es una capacidad gráfica interactiva e integrada en IDA. Ambas capacidades de graficado las vamos a estudiar seguidamente.

8.2.1.—Graficado heredado para IDA

El graficado heredado para IDA, está disponible para Windows y es proporcionada por una aplicación llamada **wingraph32**. Siempre que pidamos un gráficado "**legacy-style**", la fuente de dicho gráfico se genera y se guarda en un archivo temporal; y seguidamente se ejecuta **wingraph32** para mostrar el graficado. Una vez que el graficado ha sido cargado en memoria, wingraph32 borra el archivo temporal asociado; sin embargo podemos guardar el graficado generado utilizando la acción de wingraph32 **File > Save As**. Los archivos gráficos generados utilizan el **Graph Description Language (GDL)** para especificar sus gráficos. Las gráficas "**legacy**" incluyen lo siguiente:

** Ordinagrama de la función

** Gráfica de llamadas de todo el binario

** Gráfica de referencias cruzadas a un símbolo

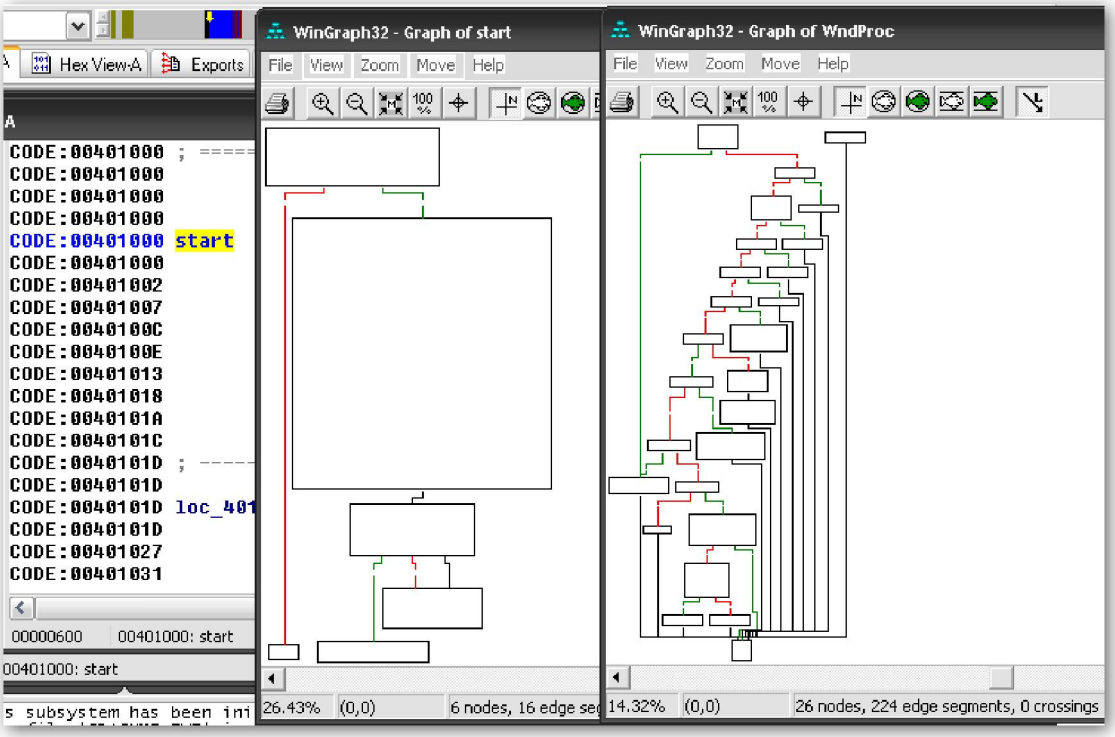
** Gráfica de referencias cruzadas desde un símbolo

** Gráfica de referencias cruzadas a medida

No obstante existen limitaciones cuando tratamos con una gráfica heredada. La principal es que las gráficas heredadas no son interactivas. La manipulación de las gráficas **legacy** está limitada a la ampliación y a la vista panorámica. No es posible editar la gráfica de ninguna forma, además no podemos manipular el contenido del desensamblado como estamos acostumbrados a realizarlo con la **graph view** integrada.

8.2.1.1.— Ordinagramas legacy

Si posicionamos el cursor en una función, y realizamos la acción **View > Graphs > Flow Chart** o utilizamos el atajo **F12**, se generará y mostrará un ordinagrama estilo legacy. La vista del ordinagrama legacy es el que más se asemeja a la vista gráfica del desensamblado que IDA tiene integrado. Ver figura abajo. Como puedes observar estos ordinagramas no son los típicos que te enseñan en las primeras clases de programación. En contraposición, en estas gráficas podemos nombrar mejor la gráfica de flujo de control así como el grupo de instrucciones de una función como un **basic block** y utilizar las líneas flechadas para indicar el flujo de un bloque a otro.

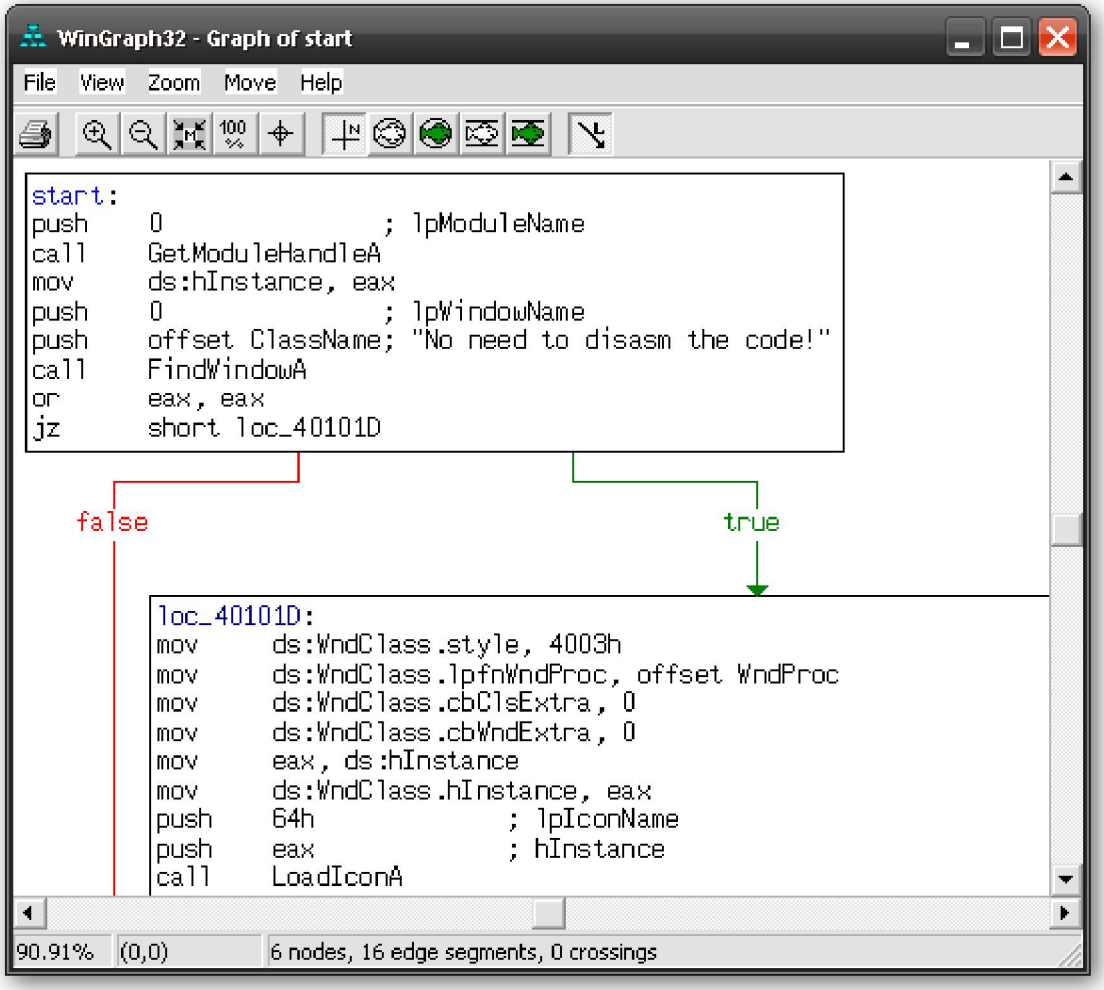


“kit-kat teórico”

Basic Blocks: En un programa, un basic block es un grupo de una o más instrucciones con una sola entrada al inicio del bloque y una sola salida al final del bloque. Generalmente, aparte de la última instrucción, cada instrucción dentro de un basic block transfiere el control a la siguiente instrucción dentro del bloque. De la misma forma, aparte de la primera instrucción, cada instrucción en un basic block recibe el control de la instrucción anterior dentro del bloque. Para cualquier basic block, el hecho de que las instrucciones call, de la función, transfieran el control fuera de la función actual no le importa y las ignora a menos que la función a la que se llama falle y no retorne normalmente. Un comportamiento característico de los basic block es; una vez que la primera instrucción es ejecutada, el resto del bloque tiene que ejecutarse hasta el final. Esto nos permite descomponer la ejecución de un programa en partes para su depuración, ya que no será necesario colocar un breakpoint en cada instrucción o avanzar instrucción a instrucción para poder saber y guardar las instrucciones que ha ejecutado. En vez de eso podemos colocar un breakpoint en la primera instrucción de cada basic block, y cada vez que rompa, cualquier instrucción asociada a dicho bloque puede marcarse como ejecutada. Como ejemplo podemos apuntar que la estructura de funcionamiento del proceso **Stalker** del **Paimei** de **Pedram Amini** se ejecuta exactamente de esta forma.

En la figura siguiente, se muestra una parte del ordinagrama relativo a la función **start** del **CRACKME.EXE**. Como podemos ver, los ordinagramas heredados ofrecen muy poca información sobre las direcciones, esto nos puede dificultar correlacionar la vista del graficado con la del listado de desensamblado correspondiente.

Los ordinagrama gráficos trazan los flujos ordinarios y de salto, de cada instrucción, dentro de una función desde el punto de entrada a la función hasta el final de ella.



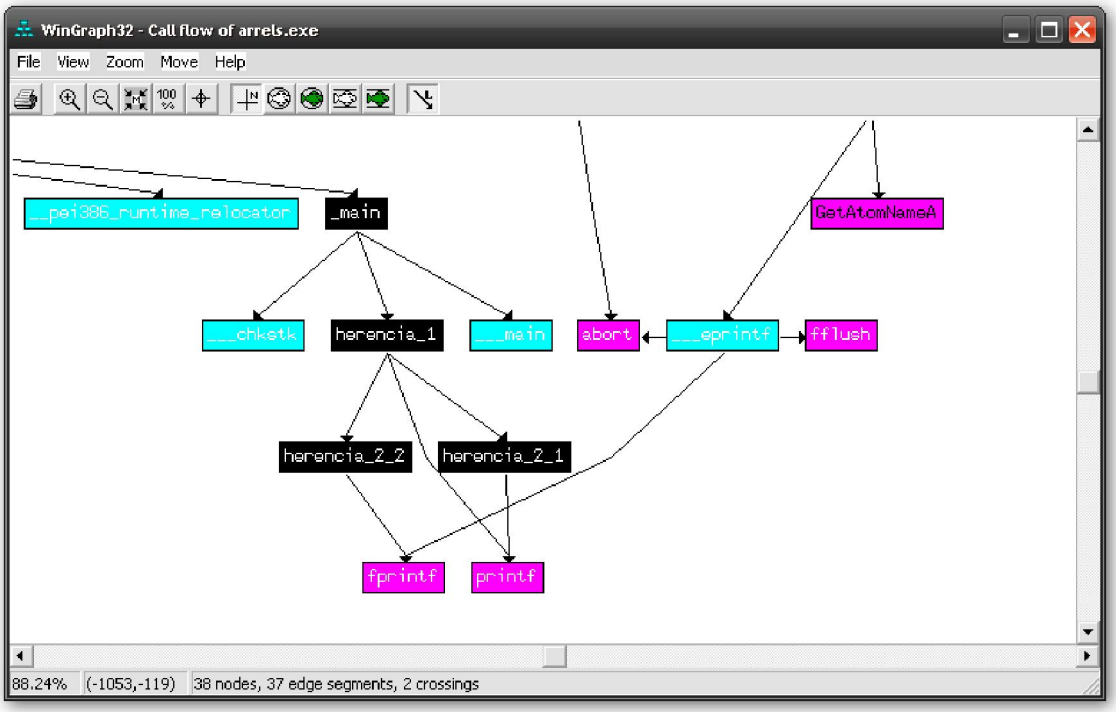
8.2.1.2.—Graficado heredado de llamadas

La gráfica de llamadas se utiliza para obtener la comprensión rápida de la jerarquía de una función utilizando las llamadas hechas dentro de un programa. Las gráficas de llamadas se generan creando una gráfica del nodo de cada función y conectando los nodos función, basándose en las referencias cruzadas de llamada existentes de una función a otra. El proceso de generación de una gráfica de llamada de una sola función puede definirse como un descenso recursivo a través de todas las funciones que son llamadas desde la función inicial. En la mayoría de los casos es suficiente descender por el árbol de llamadas hasta que se alcance una función de librería, ya que es fácil aprender cómo la función de librería opera con ella, leyendo la documentación asociada a la librería, antes de realizar ingeniería inversa con la versión compilada de la función. De hecho, en el caso de un binario enlazado dinámicamente no es posible descender hasta las funciones de librería, ya que el código para dichas funciones no está presente dentro del binario. En los binarios enlazados estáticamente la generación de las gráficas se realiza de una forma distinta. Los binarios enlazados estáticamente, contienen todo el código de las librerías que han sido asociadas al programa, con lo cual el graficado de las llamadas a función puede resultar extremadamente largo.

Para poder estudiar el graficado de las llamadas a función, utilizaremos el siguiente programa, que no hace nada más que crear una sencilla jerarquía de llamadas a función:

```
#include <stdio.h>

void herencia_2_1 ( )
  {
        printf (“ raiz herencia_2_1\n”);
  }

void herencia_2_2 ( )
   {
        fprintf (stderr, “raiz herencia_2_2\n”);
   }

void herencia_1 ( )
   {
        herencia_2_1 ( );
        herencia_2_2 ( );
        printf (“raiz herencia_1\n”);
   }

int main ( )
   {
        herencia_1 ( );
   }
```

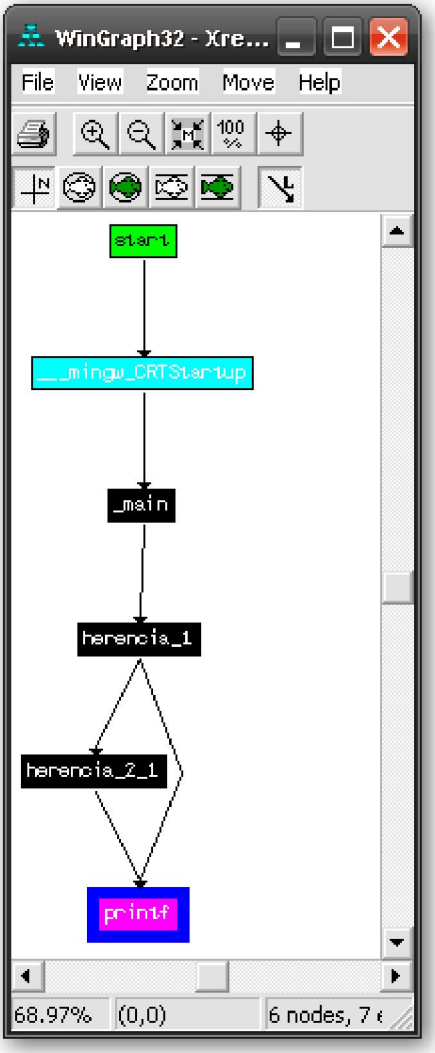
Después de compilar un binario enlazado dinámicamente utilizando **Dev C++**, podemos pedirle a IDA que nos genere el graficado de llamadas a función a través de la acción **View > Graphs > Function Calls**, y nos mostrará una vista parecida a la siguiente.



Hemos recortado la vista total del proceso para centrarnos con más detalle en la parte que nos interesa. El graficado de llamada asociado a la función **main** puede verse en el centro de la figura arriba.
IDA utiliza distintos colores para representar distintos tipos de nodos en el graficado, cambiar los colores no se puede realizar de ninguna forma.

8.2.1.3.— Graficado heredado de referencias cruzadas

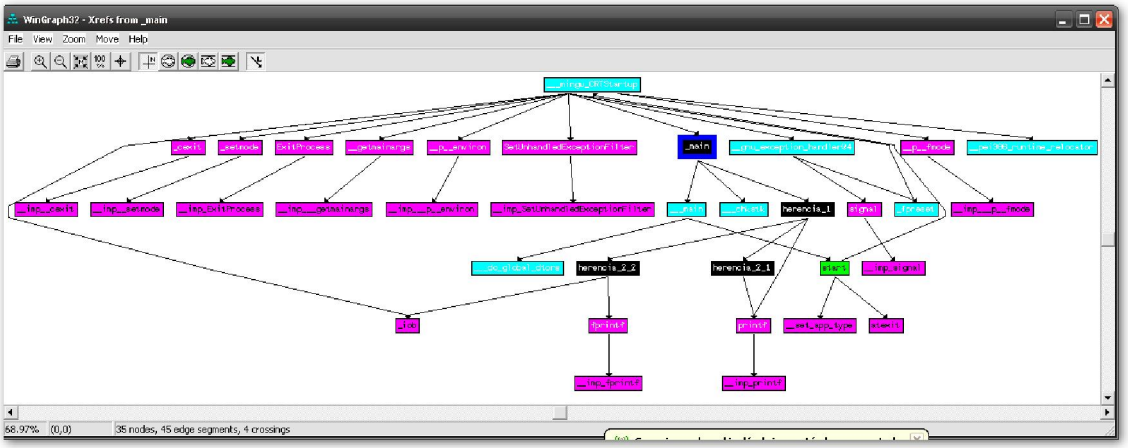
Pueden generarse dos tipos de graficados de referencias cruzadas para los símbolos globales, funciones o variables globales: Referencias cruzadas **a** un símbolo con la acción **View > Graphs > Xrefs to** y las referencias cruzadas **desde** un símbolo con la acción **View > Graphs > Xrefs from**. Para generar un graficado **Xrefs to**, se ejecuta un proceso recursivo ascendente trazando a la inversa todas las referencias cruzadas del símbolo seleccionado, hasta que se alcance un símbolo al que ninguno de los otros símbolos lo refieran. Cuando analicemos un binario, podemos utilizar un grafiado **Xrefs to** para responder a la pregunta, ¿Qué secuencia de llamadas hemos debido de realizar para alcanzar dicha función? La siguiente figura nos muestra la utilización de un grafiado **Xrefs to** para mostrarnos la ruta que se ha seguido para alcanzar a la función **printf**.



De forma similar, el graficado Xrefs to nos puede ayudar para visualizar todas las ubicaciones que referencia a una variable global y la cadena de llamadas de función

requeridas para alcanzar estas ubicaciones. El graficado de referencias cruzadas son los únicos capaces de incorporar la información de referencia cruzada de datos.

Para poder crear un graficado tipo **Xrefs From**, se ejecuta un proceso descendente recursivo para seguir las referencias cruzadas desde el símbolo seleccionado. Si el símbolo es un nombre de función, sólo son seguidas las referencias de llamada desde la función, ya que las referencias de datos a las variables globales no se muestran en el graficado. Si el símbolo es un puntero a una variable global inicializada, lo cual significa que apunta a algo, entonces es seguido el correspondiente offset de la referencia cruzada. Cuando realizamos un graficado de referencias cruzadas desde una función, el efecto es un graficado de llamada desde la raíz a la función seleccionada, como podemos ver en la figura siguiente.



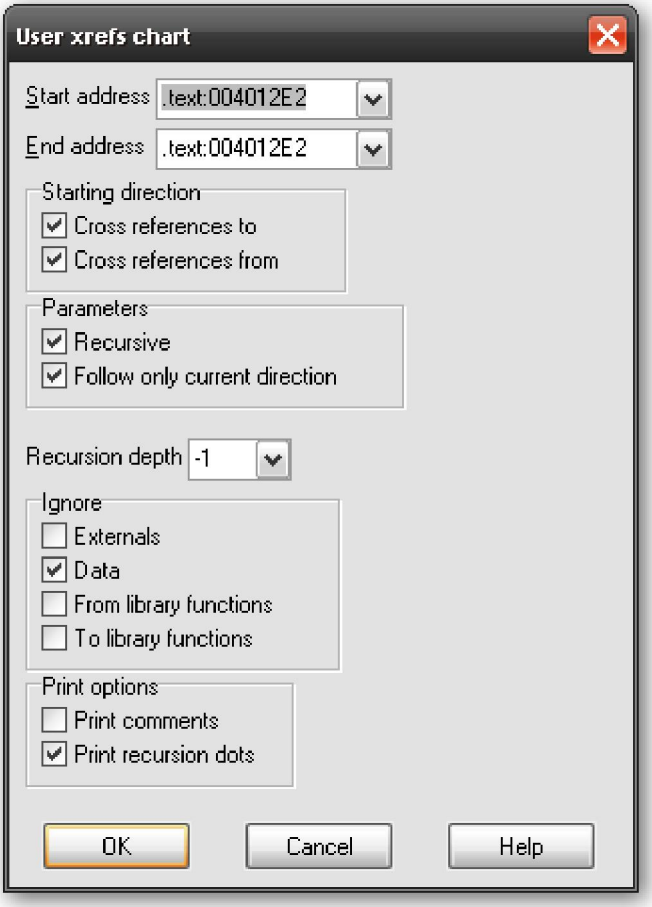
Por desgracia en una función muy compleja la vista de llamadas estará muy saturada, con lo cual nos llevará trabajo seguir las llamadas de una función.

8.2.1.4.—Graficados de referencias cruzadas a medida

El graficado a medida de referencias cruzadas, el cual en IDA toma el nombre de **User xref charts**, proporciona la máxima flexibilidad para poder generar el graficado que necesitemos. Además de combinar las referencias cruzadas a un símbolo y las referencias cruzadas desde un símbolo en un solo graficado, este graficado nos permite especificar hasta donde profundizar la recursión y los tipos de los símbolos que queramos incluir o excluir del resultado del graficado.

Para poder acceder a ello usaremos la acción **View > Graphs > User Xrefs Chart**, lo cual nos abrirá un diálogo de características que podemos acomodar a nuestro interés, figura abajo. Cada símbolo global que se encuentre dentro del rango de direcciones especificado aparecerá como un nodo dentro del graficado resultante, el cual se construye de acuerdo a las opciones especificadas en el diálogo. En la mayoría de los casos normales, las direcciones de inicio y final de las referencias cruzadas generadas desde un símbolo son las mismas. Si las direcciones de inicio y final difieren, entonces el graficado resultante es el generado por todos los símbolos no locales que se encuentren dentro del rango especificado. En el caso extremo en donde la dirección de inicio sea la dirección más baja de la base de datos y la dirección final sea la más alta en

la base de datos, el graficado resultante degenerará a un graficado de llamada a función de todo el binario.



Las opciones que están seleccionadas, figura arriba, son las opciones por defecto para los graficados a medida. El propósito de cada opción que se puede habilitar la expondremos seguidamente:

Starting direction
Dichas opciones nos permiten elegir entre buscar las referencias cruzadas; desde el símbolo seleccionado, al símbolo seleccionado o ambas a la vez. Si todas las otras opciones de la parte izquierda están habilitadas por defecto, si restringimos la dirección de inicio a **Cross referentes to** el resultado será un graficado **Xrefs To**, mientras que si restringimos la dirección de inicio a **Cross referentes from** generaremos un graficado **Xrefs From**.

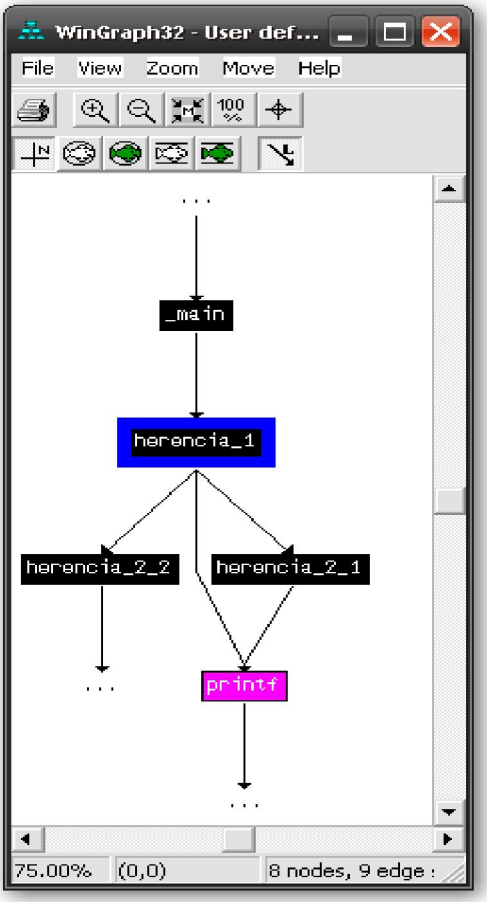
Parameters
La opción **Recursive** habilita el descenso recursivo (xrefs from) o el ascenso recursivo (xrefs to) desde el símbolo seleccionado. La opción **Follow Only current direction** fuerza cualquier recursión a sólo en una dirección. En otras palabras, si esta opción se selecciona, si un nodo **B** es descubierto desde un nodo **A**, el descenso recursivo en **B** añade todos los nodos encontrados los cuales puedan alcanzarse sólo “desde” el nodo **B**. Los nodos nuevos descubiertos referidos al nodo **B** no se añadirán al graficado. Si elegimos deshabilitar la opción, entonces se eligen ambas direcciones como inicio, con lo cual cada nodo nuevo añadido al graficado que se haya encontrado en ambas direcciones tanto “**a**” como “**desde**”.

Recursion depth
Esta opción habilita la máxima profundidad de recursión y es útil para limitar el tamaño de los graficados generados. Un valor de **-1**, tiene el efecto de la recursión de máxima profundidad generándonos el graficado más grande posible.

Ignore
Estas opciones indican los tipos de nodos que serán excluidos en el graficado generado. Es otro medio para restringir el tamaño del graficado resultante. Particularmente si ignoramos las referencias cruzadas de las funciones de librería, simplificaremos drásticamente el graficado en los binarios enlazados estáticamente. El truco está en que IDA reconozca cualquier función de librería posible. El reconocimiento del código de librería lo estudiaremos más adelante.

Print options
Estas opciones controlan dos aspectos de formato del graficado. **Print Comments** nos proporciona cualquier comentario de una función, el cual se incluirá en el graficado del nodo de la función. Si seleccionamos **Print recursión dots**, y queremos continuar la recursión más allá de los límites especificados, se nos mostrará un nodo con puntos suspensivos indicándonos que es posible una recursión más amplia.

En la figura siguiente vemos un graficado de referencias cruzadas a medida de la función **herencia_1** de nuestro ejemplo, utilizando las opciones por defecto y una recursión depth de 1.



La capacidad más poderosa de graficado heredado en IDA, la proporciona el graficado de referencias cruzadas a medida. Los ordinagramas heredados pueden reemplazarse por la vista gráfica integrada de IDA basada en la vista del desensamblado, con lo cual los graficados heredados simplemente se utilizan para generar el graficado de las referencias cruzadas.

8.2.2.—Vista gráfica integrada de IDA

Con la versión 5.0 IDA se introduce una vista de desensamblado interactivo, con base gráfica integrada estrechamente con IDA. Como ya hemos mencionado anteriormente, el modo gráfico integrado proporciona una alternativa al listado de desensamblado tipo texto. El desensamblado de funciones en el modo gráfico se muestra como el ordinagrama de graficado de control de flujo del tipo heredado. En el modo gráfico, debido al uso del control de flujo orientado a una función, sólo se puede mostrar una función graficada a la vez, y además no se pueden utilizar las instrucciones que dependen de la función fuera de ella. Por el contrario en los casos en que queramos mirar varias funciones al momento, o cuando necesitemos mirar instrucciones que no son parte de la función, deberemos referirnos al listado de texto del desensamblado.

En la parte 4 detallamos básicamente la manipulación de la "**graph view**", por lo que reiteraremos algunos puntos otra vez. La conmutación entre vista de desensamblado y vista gráfica se realiza pulsando la barra de espaciado o haciendo click derecho en cualquier parte del desensamblado y escogiendo la opción **Text View** o **Graph View**. La forma más fácil para movernos por la vista gráfica es hacer click en el fondo de la vista y arrastrarlo a la posición deseada. Para gráficas muy grandes podemos utilizar la opción de ventana **Graph Overview**. La ventana Graph Overview muestra un rectángulo discontinuo el cual podemos ir moviendo y en la ventana de desensamblado se nos muestra la parte por donde éste va pasando. Debido a que la ventana Graph Overview muestra una versión en miniatura del gráfico entero, usándola como panorámica nos ahorramos irnos posicionando cada vez utilizando el ratón.

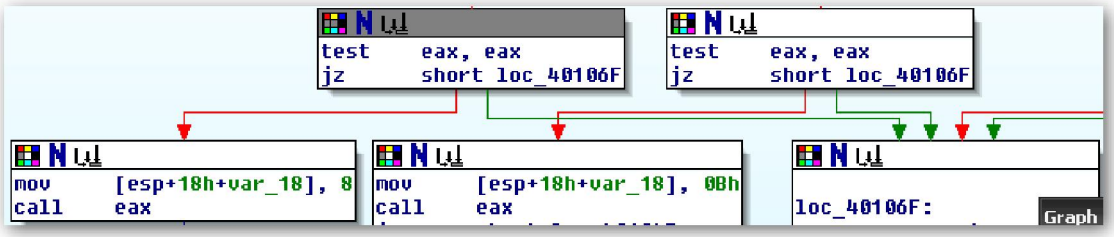
No existen diferencias significativas entre manipular un desensamblado en modo gráfico o hacerlo en modo texto. Si hacemos doble click la navegación del desensamblado continua en el mismo lugar sea gráfico o texto. En cualquier momento que queramos trasladarnos a una ubicación que no esté dentro de la función, por ejemplo a una variable global, ésta se mostrará automáticamente cambiando a modo texto. El gráfico se restaurará de forma automática cuando volvamos a trasladarnos a la función. Para acceder a las variables de pila se realiza de la misma forma que en el modo texto, con la vista resumida de pila mostrada en el bloque básico de la función. La vista detallada del **stack frame** será mostrada si realizamos doble click en cualquier variable de pila, igual que en modo texto. Todas las opciones de formato de operando de instrucción en modo texto, están al alcance de la misma forma en modo gráfico.

La principal diferencia en cuanto a interfaz de usuario del modo gráfico es el modo de manejar las funciones, ya que se muestran como gráficos individuales, nodos, de ellas. En la figura siguiente vemos un gráfico nodo con su barra y sus botones de control.

De izquierda a derecha en la barra de título, existen tres botones los cuales nos permiten cambiar el color del nodo, asignar o cambiar el nombre del nodo y acceder a la lista de referencias cruzadas del nodo. Colorear los nodos es una forma de saber qué nodos hemos analizado o simplemente hacerlos sobresalir de los demás, porque contengan algún tipo de código de particular interés. Una vez le asignamos un color al nodo, dicho color también se utiliza como fondo en las correspondientes instrucciones en modo texto. Para eliminar cualquier color realizaremos un click derecho en la barra de título y seleccionaremos **Set node color to default**.

El segundo botón de la barra de título se utiliza para asignar un nombre a la dirección de la primera instrucción del bloque básico del nodo. A menudo los bloques básicos son objetivos de instrucciones de salto, algunos nodos pueden tener asignados **dummy names** como resultado de ser objetivo de algún salto de referencia cruzada. Sin embargo es posible asignarle otro nombre, aunque ya tenga uno asignado. Consideremos las siguientes líneas de código:

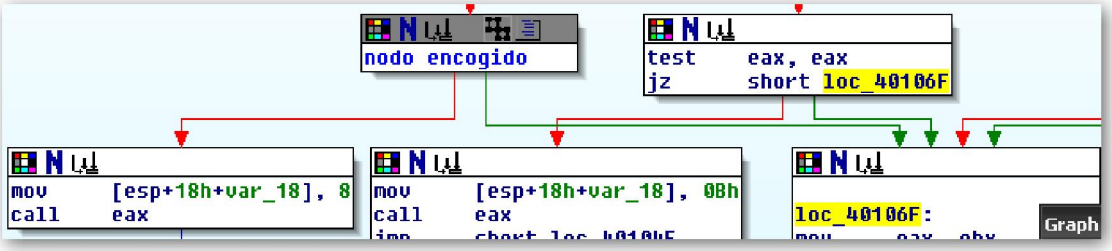
La instrucción en **00401043** tiene dos posibles sucesiones, una **loc_40106F** y la otra **00401045**. Debido a que tiene dos sucesiones, debe terminar un bloque básico, lo cual da como resultado que la primera instrucción de un nuevo bloque básico será **00401045**, y como no es marcado explícitamente por un salto no tendrá Dummy name asignado.



El tercer botón es utilizado para acceder a la lista de referencias cruzadas del nodo, ya que los comentarios no se muestran por defecto en el modo gráfico, es el modo más fácil para trasladarnos a cada ubicación referenciada en el nodo. Al igual que las listas de referencias cruzadas estudiadas anteriormente, la lista de referencias cruzadas generadas por el nodo también contiene una entrada del flujo ordinario en el nodo, designado por tipo de salto. Esto es necesario debido a que no siempre es obvio, en la vista gráfica, que nodo es el predecesor lineal del nodo seleccionado. Si queremos ver los comentarios normales de referencia cruzada en modo gráfico, lo podemos conseguir accediendo a **Options > General > Cross-References** y habilitar la opción **Number of displayed xrefs** con un número que no sea **cero**.

Los nodos dentro de una gráfica los podemos agrupar solos o junto con otros, a fin de reducir el colapso gráfico. Para agrupar múltiples nodos, realizamos **CTRL-click** en la barra de título de cada nodo que queramos agrupar, y después **click derecho** en la barra de título de cualquiera de los nodos a agrupar y seleccionamos **Group Nodes**. Esto nos proporcionará poder introducir un texto, por defecto en la primera instrucción del grupo,

para mostrarse en el nodo encogido. La siguiente figura nos muestra el resultado de agrupar el nodo de la figura anterior y cambiarle el texto por "**nodo encogido**".



Observemos que han aparecido dos botones más en la barra de título. El primero es para expandir el nodo encogido y el segundo para editar el texto del nodo. Al expandir el nodo nos proporcionará el grupo de nodos con su forma original; de hecho agruparlos no cambia en nada a los nodos. Cuando un grupo es expandido, los dos nuevos botones desaparecen y son reemplazados por un solo botón **Collapse Group**. Un grupo expandido puede fácilmente encogerse otra vez utilizando el botón **Collapse Group** o haciendo click derecho en la barra de título de cualquier nodo en el grupo y seleccionando **Hide Group**. Para eliminar completamente una agrupación de uno o más nodos, haremos click derecho en la barra de título del nodo encogido o de uno de los nodos expandidos y seleccionar **Ungroup Nodes**. Esta acción también tiene el efecto de expandir el grupo si este está encogido.

Performance Bigundill@