15.2.1.—Conocer los archivos Header

Aunque el **readme.txt** del SDK nos proporciona una vista general de los archivos header más utilizados normalmente, en este apartado sacaremos a la luz alguna que otra información útil para trabajar con dichos archivos. Para empezar diremos que la mayoría de archivos header utiliza el sufijo **.hhp**, mientras que el resto utiliza el sufijo **.h**. Este es uno de los errores triviales que se comete cuando nombramos archivos header para incluirlos en nuestros archivos. En segundo lugar, el archivo **ida.hhp** es el archivo header principal del SDK y debe incluirse en todos los proyectos relacionados con el SDK. Y por último, el SDK utiliza directivas de preprocesado diseñadas para excluir el acceso a las funciones que Ilfak considera peligrosas como **strcpy** y **sprintf**.

Si queremos tener una lista completa de dichas funciones nos tendremos que referir al archivo header **pro.h**. Para restaurar el acceso a estas funciones, antes deberemos definir una macro llamada **USE_DANGEROUS:FUNCTIONS,** antes de incluir el header **ida.hhp** en nuestros archivos. Un ejemplo sería el siguiente:

```
#define USE_DANGEROUS_FUNCTIONS
#include <ida.hhp>
```

Olvidarse definir **USE_DANGEROUS_FUNCTIONS** nos da como resultado un error de estructura provocando que, en el caso de utilizar la función **sprintf**, nos mostraría **dont_use_sprintf** ya que lo considera un símbolo no definido. Para compensar el acceso restringido a dichas funciones, el SDK define las equivalentes seguras de ellas, generalmente en la forma de función **qXXXX** por lo tanto para nuestros ejemplos serían **qstrcpy** y **qsprintf**. Estas versiones seguras también están declaradas en **pro.h**.

De la misma forma, el SdK restringe el acceso a muchas variables estándares de entrada y salida (input/output) lo mismo que a funciones como **stdin, stdout, fopen, fwrite** y **fprintf**. Estas restricciones son debidas en parte a las limitaciones del compilador Borland. Para estos casos el SDK también define funciones seguras con la misma forma **qXXXX**, por lo tanto podría ser **qfopen** y **qfprintf**. Para poder acceder a las funciones estándares debemos definir la macro **USE_STANDARD_FILE_FUNCTIONS** antes de incluir el archivo **fpro.h** el cual está incluido en **kernwin.hpp** y que éste a su vez está incluido en otros archivos.

En la mayor parte de los casos, cada archivo header del SDK contiene una descripción breve de su propósito y extensos comentarios describiendo las estructuras de datos y las funciones que son declaradas en el archivo. El conjunto de estos comentarios constituye la documentación API de IDA. Una breve descripción de algunos de los header más comúnmente utilizados son los siguientes.

area.hpp

Este archivo define la struct **area_t**, la cual representa un bloque de direcciones contiguas dentro de una base de datos. Esta struct sirve como clase base de otras clases que se basan en el concepto de un rango de direcciones. Es raro tener que incluir dicho archivo directamente, ya que normalmente está incluido en archivos que definen las subclases de **area_t**.

auto.hpp

En este archivo se declaran las funciones utilizadas para trabajar con el autoanalizado de IDA. El autoanalizado ejecuta tareas de análisis, en espera, cuando IDA no está procesando ningún evento introducido por el usuario.

bytes.hpp
En este archivo se declaran las funciones utilizadas para trabajar con byte de la base de datos. Dichas funciones se utilizan para leer y escribir byte de la base de datos, así como manipular las características de dichos byte. Diversas funciones también proporcionan acceso a las banderas asociadas con los operandos de una instrucción, y otras permiten la manipulación de comentarios ordinarios y repetitivos.

dbg.hpp
En este archivo se declaran las funciones que proporcionan control programático del depurador de IDA.

entry.hpp
En este header se declaran funciones para trabajar con los entry point de un archivo. Para librerías compartidas, cada función o valor de dato exportado es considerado un entry point.

expr.hpp
En este archivo se declaran las funciones y las estructuras de datos para trabajar con los constructores IDC. Con éste es posible modificar funciones IDC, añadir nuevas funciones IDC o ejecutar declaraciones IDC dentro de los módulos.

fpro.h
Este archivo contiene las funciones, de archivos I/O, alternativas. Como **qfopen**, explicadas anteriormente.

frame.hpp
Este header contiene las funciones utilizadas para manipular el **stack frame**.

funcs.hpp
Este header contiene funciones y estructuras de datos para trabajar con las funciones de desensamblado, así como funciones para trabajar con las firmas **FLIRT**.

gdl.hpp
En este archivo se declaran las rutinas para generar gráficos utilizando el **GDL**.

ida.hpp
Este es el archivo header principal, requerido para trabajar con el SDK. Este archivo contiene la definición de la estructura **idainfo**, así como la declaración de la variable global **inf**, la cual contiene varios campos que contienen la información de la actual base de datos así como los campos inicializados con las características del archivo de configuración.

idp.hpp
Este archivo contiene las declaraciones de estructuras que forman los cimientos de los módulos de procesador. En este se definen la variable global **ph**, la cual describe el

actual módulo de procesador y la variable global **ash**, que describe el actual ensamblador.

kernwin.hpp
En este archivo se declaran las funciones para interactuar con el usuario y la interfaz de usuario. Las funciones SDK equivalentes a las **AskXXX** de IDC, son declaradas aquí como las funciones utilizadas para habilitar la posición de pantalla y configuración de atajos de teclado asociados.

lines.hpp
En este archivo se declaran las funciones para generar las líneas del desensamblado formateadas y coloreadas.

loader.hpp
Este archivo contiene las declaraciones de estructuras **loader_t** y **plugin_t** requeridas para la creación de módulos de carga y módulos plug-in, respectivamente, así como las funciones utilizadas durante la fase de carga de archivos y las funciones para la activación de los plug-in.

name.hpp
En este archivo se declaran las funciones para manipular las ubicaciones nombradas, opuestamente a los nombres dentro de estructuras o stack frame, los cuales se tratan en **stuct.hpp** y **funcs.hpp**, respectivamente.

netnode.hpp
Los netnode son la estructura de almacenamiento de bajo nivel accesible a través de API. Los detalles de los netnode normalmente se ocultan por la interfaz de usuario de IDA. Este archivo contiene la definición de la clase **netnode** y las funciones de bajo nivel para la manipulación de ella.

pro.h
Este archivo incluye el nivel superior de las definiciones de tipo (typedefs) y las macros requeridas en cualquier módulo SDK. No necesitamos incluirlo, explícitamente en nuestros proyectos, ya que está incluido en **ida.hpp**. Entre otras cosas, en este archivo se define la macro **IDA_SDK_VERSION**. Ésta proporciona los medios para determinar con que versión SDK se ha construido un módulo y puede ser verificada para proporcionar una posible compilación al utilizar versiones distintas del SDK. Observemos que la IDA_SDK_VERSION se introdujo con la versión SDK 5.2. Anteriormente a esta versión, no existía forma para poder saber que SDK se ha utilizado.

search.hpp
En este archivo se declaran las funciones para ejecutar distintos tipos de búsqueda en la base de datos.

segment.hpp
Este archivo contiene la declaración de la clase **segment_t** y la subclase **area_t**, las cuales son utilizadas para describir secciones particulares **(.text, .data. etc.**) dentro de un binario. Las funciones para trabajar con segmentos están declaradas aquí.

struct.hpp
Este archivo contiene la declaración de la clase **struc_t** y las funciones para manipular estructuras dentro de la base de datos.

typeinf.hpp
En este archivo se declaran las funciones para trabajar con las librerías tipo de IDA. Además, también se declaran las funciones que nos proporcionan acceso a las funciones de firmas, incluidos los tipos de retorno de función y las secuencias de parámetros.

ua.hpp
Este archivo declara las clases **op_t** e **insn_t** utilizadas extensamente en los módulos de procesador. También se declaran las funciones utilizadas para el desensamblado individual de instrucciones y para la generación de texto en distintas partes de cada línea del desensamblado.

xref.hpp
En este se declaran los tipos de dato y funciones requeridas para añadir, borrar e iterar referencias cruzadas de código y de datos.

Esta lista describe aproximadamente la mitad de los archivos header que existen en el SDK. Te aconsejamos que te familiarices no sólo con los archivos de esta lista sino que también lo hagas con los demás archivos header para poder profundizar en el SDK. Las funciones que componen alguna de las API publicadas, son marcadas como **ida_export**. Sólo las funciones designadas como **ida_export** serán exportadas de las librerías de enlace contenidas en el SDK, las que no estén designadas de dicha forma no podrán ser utilizadas en nuestros módulos.

Performance Bigundill@