15.1.—Introducción al SDK

El SDK de IDA se distribuye de la misma forma que los otros extras de IDA vistos hasta el momento. El archivo zip que contiene el SDK se puede encontrar en el CD de IDA, o los usuarios autorizados pueden descargarlo del website de Hex-Rays. Cada versión del SDK es nombrado para la versión compatible de IDA, por ejemplo, idasdk52.zip va con la versión 5.2 de Ida. El SDK se caracteriza por la mínima información proporcionada, que en este caso no es más que un archivo **readme.txt** para el SDK y unos archivos README adicionales para plug-ins, módulos de carga y módulos de procesador.

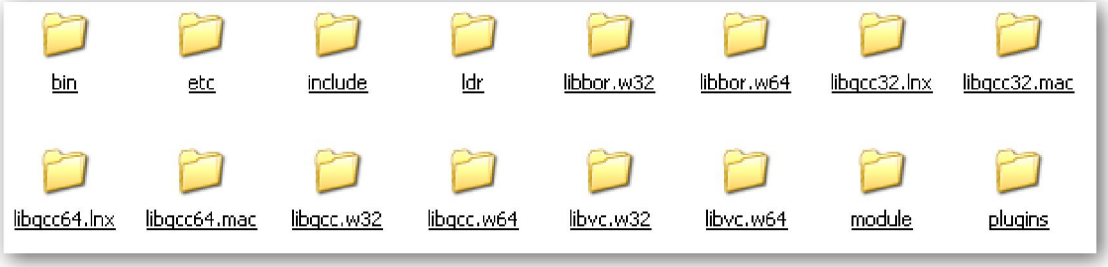
El SDK interpreta el enlace de los módulos programados para que puedan interactuar recíprocamente con IDA. Antes de la versión SDK 4.9 no era fácil compilar un módulo con otra versión y utilizarla en otra, sin necesidad de cambios. Desde la versión SDK 4.9, Hex-Rays eligió normalizar las API existentes, lo que significó que no sólo los módulos no necesitaban cambios para compilar con un buen resultado con nuevas versiones del SDK, sino que los módulos también son compatibles con los binarios de nuevas versiones de IDA. Esto significa que los usuarios de los módulos no necesitan esperar a que sus autores actualicen su código fuente o hagan versiones actualizadas de los binarios de los módulos disponibles, cada vez que una nueva versión de IDA salga a la luz. Esto no significa que los enlaces de API existentes no sufran ninguna alteración; Hex-Rays continua introduciendo nuevas características con cada versión nueva del SDK es decir, cada nuevo SDK supera al anterior. Los módulos que utilizan las nuevas características normalmente son los que no tienen compatibilidad con versiones anteriores de IDA y del SDK.

15.1.1.—Instalación del SDK

El archivo Zip que contiene el SDK no contiene un directorio a nivel superior. Esto es debido a que SDK comparte varios nombres de subdirectorios con IDA, es recomendable crear un directorio dedicado a SDK y extraer el contenido de SDK a dicho directorio. Esto nos hará mucho más fácil distinguir entre los componentes de SDK y los componentes de IDA. Si tenemos en mente instalar distintas versiones del SDK , es conveniente dar el nombre del directorio con la versión utilizada. A partir de este momento cuando nos refiramos a <SDKDIR> nos referiremos al directorio **Archivos de Programa\IDA\SDK**, lugar donde tendremos nuestro SDK.

15.1.2.—Esquema del SDK

Una comprensión básica de la estructura de directorios utilizados dentro del SDK nos será útil, para saber dónde encontrar documentación y saber dónde podemos encontrar los módulos que construimos. Un resumen rápido y detallado de lo que encontraremos en el SDK es lo que sigue.



Directorio bin
En este directorio se guardan los módulos compilados de nuestros scripts realizados y construidos con éxito. Instalar un módulo, supone copiar el módulo desde el subdirectorio apropiado dentro de **bin** al subdirectorio apropiado en donde tengamos instalado IDA. La instalación de un módulo la estudiaremos con más detalle en las tres partes siguientes a esta 16, 17 y 18. Este directorio también contiene una herramienta de post-procesamiento requerida para la creación de los módulos de procesador.

Directorio etc
Este directorio contiene el código fuente de las dos utilidades requeridas para construir ciertos módulos SDK. Las versiones compiladas de estas utilidades también están incluidas en el SDK.

Directorio include
Este directorio contiene los encabezados de los archivos que definen el enlace con las API de IDA. Resumiendo, cada estructura de datos API que es permitida utilizar y cada función de API que es permitida llamar y que esté declarada en uno de los encabezados de archivo de este directorio. El archivo **readme.txt** del SDK contiene una visión general de algunos de los encabezados más utilizados de este directorio. Los archivos de este directorio constituyen la mayor documentación, es como leer el código fuente del SDK.

Directorio Idr
Este directorio contiene el código fuente y scripts construidos de algunos ejemplos de módulos de carga. El archivo README para los módulos de carga no es más que un informe detallado del contenido de este directorio.

Directorios lib XXX.YYY
Estos directorios contienen las librerías con las que hay que vincular nuestros módulos. La parte **XXX** del nombre del directorio indica el compilador usado por la librería, mientras que la parte **YYY** indica la plataforma usada por la librería. Por ejemplo, **libvc.w32** contiene la librería para utilizar con Visual Studio y la version IDA 32-bit en la plataforma Windows, mientras que **libgcc64.lnx** es la librería para utilizar con gcc y la versión de IDA 64-bit en la plataforma Linux.

Directorio module
Este directorio contiene el código fuente y scripts construidos de distintos ejemplos de módulos de procesador. El archivo README para los módulos de procesador no es más que un informe detallado de este directorio.

Directorio plug-ins
Este directorio contiene el código fuente de varios scripts construidos como ejemplo de módulos plug-in. El archivo README para plug-ins proporciona una explicación general de la arquitectura plug-in

Directorio SDK (nivel superior)
Este directorio contiene varios archivos usados para construir módulos así como el archivo **readme.txt** principal. Varios archivos **install_xxx.txt** que contienen

información respecto a instalación y configuración de varios compiladores, por ejemplo, **install_visual.txt** explica la configuración de Visual Studio.

Tengamos presente que la documentación para utilizar el SDK está esparcida. Para la mayor parte de programadores, el conocimiento del SDK se ha adquirido con el método prueba y error y por la exploración intensiva de los contenidos del SDK. Podemos informarnos realizando preguntas en distintos foros con personas familiarizadas con el SDK. Un excelente recurso que proporciona una introducción al SDK y a la programación de plug-in es una guía escrita por Steve Micallef titulada **IDA Plug-in Writing in C/C++.**

15.1.3.—Configurar un entorno de trabajo

Uno de los aspectos más frustrantes en utilizar el SDK es que no tiene relación alguna con la programación. Por otro lado podemos encontrar relativamente fácil el código para solucionar un problema, pero también nos podemos encontrar en que sea virtualmente imposible construir nuestro módulo. Esto es así debido a que es difícil que con un simple código podamos soportar la distinta variedad de compiladores existentes, y de hecho es complicado que los formatos de librerías reconocidos por los compiladores Windows sean compatibles con otros.

Todos los ejemplos incluidos en el SDK has sido construidos utilizando herramientas Borland. En el **install_make.txt** nos encontramos con el siguiente apunte de Ilfak:

Las versiones WIN32 han sido creadas sólo para Borland C++ CBuilder v4.0. Probablemente en los BCC v5.2 también funcionen, pero no ha sido verificado.

Sepamos también, que otros archivos **install_xxx** ofrecen indicaciones de cómo construir módulos con otros compiladores. Algunos de los módulos de ejemplo contienen archivos Visual Studio (por ejemplo, <SDKDIR>\plugins\vcsample), en el **install_visual.txt** nos encontramos con los pasos apropiados para configurar proyectos SDK utilizando Visual C++ Express 2005. Para poder construir módulos utilizando herramientas estilo UNIX, dentro de un sistema estilo UNIX como Linux o utilizando un entorno como **Cygwin**, el SDK nos proporciona un script llamado **idamake.pl** el cual convierte los archivos hechos con Borland a archivos tipo Unix antes de iniciar el proceso de construcción. Dicho proceso se detalla en el archivo **install_linux.txt**

Observación: La línea de órdenes para crear scripts del SDK espera una variable de entorno llamada **IDA** para apuntar al <**SDKDIR**>. Para habilitarla globalmente para los scripts editaremos **<SDKDIR>\allmake.mak** y **<SDKDIR>\allmak.unx** o añadiremos una variable de entorno **IDA** a nuestro entorno global de sistema.

Como ya hemos apuntado antes en la guía de Steve-Micallef, se nos proporcionan excelentes instrucciones para configurar nuestro entorno de trabajo para poder realizar plugins con distintos compiladores. Como preferencia personal para construir módulos para versiones de IDA Windows con SDK, es utilizar las herramientas **gcc** y **make** del entorno **Cygwin**. Los ejemplos que se presentarán en las partes 16, 17 y 18 incluirán los makefiles y los archivos de Visual Studio los cuales son fáciles de modificar a conveniencia de nuestras necesidades. La configuración específica de cada módulo también se explicará en cada una de dichas partes.