15.0.— Alineando ideas

A lo largo de los escritos anteriores hemos utilizado frases como “IDA hace esto o IDA hace aquello”. Ciertamente ida puede hacer mucho más por nuestros intereses, su inteligencia se le tiene que atribuir a los distintos módulos de los cuales ida depende. Por ejemplo, el módulo de procesador es el que toma todas las decisiones durante la fase de análisis, con lo cual podemos asegurar que IDA es tan fabuloso gracias a dichos módulos de los que depende. De hecho los actuales propietarios de IDA, “Hex-Rays”, hacen un gran esfuerzo para que el usuario casual de ida, tenga bien ocultada la arquitectura de dichos módulos bajo la interfaz de usuario.

En un momento dado, podemos necesitar tener más capacidad sobre el análisis de un binario, que el que nos pueda proporcionar el lenguaje IDC, una de las causas puede ser por razones de ejecución o debido a que lo que deseamos realizar no estaba previsto en las funciones de IDC con lo cual no podemos realizarlo. Llegado dicho momento, tendremos que utilizar alguna herramienta mucho más avanzada, esta puede ser el kit de desarrollo de IDA **“software development kit (SDK)”**, con la cual podremos construir nuestros propios módulos compilados y utilizarlos en IDA.

> **Observación:** El motor de scripts de IDC, está desarrollado con base del SDK. Todas las funciones IDC, acaban trasladándose a llamadas de una o más funciones del SDK ejecutando éstas la tarea. Todo lo que podamos realizar con IDC lo podremos realizar con el SDK, pero esta premisa no se cumple a la inversa. El SDK nos proporciona mucho más poder que el que nos da el lenguaje IDC, y muchas de las acciones del SDK no tienen contrapartida en IDC.

El SDK nos muestra la interfaz de programación interna de IDA en la forma de librerías C++ y los encabezados de los archivos requeridos para interconectar con dichas librerías. El SDK se necesita para crear los módulos de carga que manejarán los nuevos formatos de archivo, los módulos de procesador para desensamblar los conjuntos de instrucciones de nuevas CPU y los módulos de plug-in, los cuales podríamos enumerarlos como la alternativa más poderosa de compilación frente a los scripts.

> EL PODER EN NUESTRAS MANOS
> Al trabajar con C++, podríamos presuponer tener acceso a una gran variedad de librerías C++, incluidas las API nativas del sistema operativo. Al utilizar dichas librerías, podemos pensar el poder utilizar la gran variedad de características sofisticadas de dicho lenguaje, en la construcción de nuestros módulos. Sin embargo, deberemos tener mucho cuidado en elegir cada funcionalidad que incorporaremos a ellos, ya que según lo realicemos podrá provocar inestabilidad en IDA. El ejemplo más concreto de este hecho, es que IDA es una aplicación de un solo hilo (thread). No se puede hacer nada para sincronizar el acceso a las estructuras de nivel bajo de la base de datos, incluso ni con las facilidades proporcionadas por el mismo SDK. Tengamos en cuenta que nunca deberemos crear hilos adicionales los cuales puedan acceder simultáneamente a la base de datos y además deberemos tener claro y entender, que ningún bloque de operaciones que se ejecute. Ida no nos responderá hasta que todas las operaciones se hayan completado.

En estos siguientes escritos aprenderemos algunas de las capacidades del núcleo de SDK. Encontraremos la utilidad de estas capacidades para crear plug-ins, módulos de carga y módulos de procesador. Cada tipo de módulo lo estudiaremos individualmente, en tres distintas partes; plug-in, módulos de carga y módulos de procesador. Los ejemplos de cada parte se ofrecerán intentando suministrar un contexto específico en el cual se podrían utilizar.

Performance Bigundill@