17.5.—Estrategias para otros cargadores

Si tenemos tiempo, podemos examinar los cargadores de ejemplo que contiene el SDK. En ellos encontraremos distintos tipos de cargadores. Uno de ellos es el cargador de Java **<SDKDIR>\ldr\javaldr**. En ciertos formatos, el emparejamiento entre módulo de carga y módulo de procesador es ínfimo. Veamos, una vez que el cargador anota los puntos de entrada en el código, el módulo de procesador no necesitará ninguna otra información para poder realizar el desensamblado del código. No obstante, ciertos módulos de procesador pueden requerir más información sobre el archivo fuente y puede ser necesario que ejecuten, otra vez, gran parte del análisis realizado previamente por el cargador. Para evitar dicha duplicación de trabajo, se puede realizar un mayor acoplamiento entre el módulo de carga y el de procesador. De hecho, la idea utilizada en el cargador Java es la de pasar todo el trabajo de carga (normalmente se realiza en la función del cargador **load_file**) al módulo de procesador utilizando el siguiente código:

```
static void load_file(linput_t *li, ushort neflag, const char *) {
  if (ph.id != PLFM_JAVA) {
    set_processor_type("java", SETPROC_ALL | SETPROC_FATAL);
  }
  if (ph.notify(ph.loader, li, (bool)(neflag & NEF_LOPT))) {
    error("Internal error in loader<->module link");
  }
}
```

El cargador Java, la única tarea que realiza es verificar que el tipo de procesador esté habilitado para **java,** cuando lo ha verificado el cargador envía un mensaje de notificación, **ph.loader** (definido en **idp.hpp**), al módulo de procesador para informarle que la fase de carga se ha iniciado. Recibida la notificación, el módulo de procesador Java se hace responsable de la carga y en el proceso toma una importante cantidad de información de estado interna que será utilizada otra vez cuando al procesador se le ordene la ejecución de tareas de desensamblado.

Determinar si dicha estrategia nos conviene o no, dependerá de si estamos programando ambos módulos asociados y si además, queremos que el procesador tenga acceso a la información normalmente tomada por el cargador como: segmentación, campos del encabezado de archivo, información de depuración, etc.

Otros medios para pasar información de estado del cargador al módulo de procesador suponen la utilización de **netnodes** de la base de datos. Durante la fase de carga, el cargador puede optar por rellenar netnodes específicos con información la cual se podrá recuperar más tarde a través del módulo de procesador durante la fase de desensamblado. Para finalizar, tengamos en cuenta que el acceso frecuente a la base de datos para recuperar información almacenada puede ser algo lento utilizando los tipos dato C++ disponibles.

Performance Bigundill@