15.5.3.1.— Enumerar los llamadores de una función

En el siguiente ejemplo, realizaremos lo opuesto del ejemplo anterior, iteraremos a través de todas las referencias cruzadas a un símbolo en particular. Además aprovecharemos la ocasión anterior de haber hablado sobre las funciones peligrosas como **strcpy** y **sprintf**, para que el script nos muestre dónde son llamadas y realizadas dichas funciones en un programa

Script enumerar los llamadores de una función, en este caso "peligrosa"

```
#include <idc.idc>
static lista_llamadores (funcion_peligrosa)    {
   auto funcion, direccion, refcruzada, fuente;
      funcion = LocByName (funcion_peligrosa);
   if (funcion == BADADDR)    {
      Warning ("Lo siento, %s no existe en la base de datos", funcion_peligrosa);
   }
   else    {
              for (direccion = RfirstB (funcion); direccion != BADADDR; direccion = RnextB
(funcion, direccion))    {
                  refcruzada = XrefType ( );
                  if (refcruzada == fl_CN || refcruzada == fl_CF)    {
                      fuente = GetFunctionName (direccion);
                      Message ("La funcion peligrosa %s es llamada desde 0x%x en la dirección %s\n",
funcion_llamada, direccion, fuente);
             }
         }
     }
}
static main ( )    {
   lista_llamadores ("_strcpy");
   lista_llamadores ("_sprintf");
}
```

En este ejemplo utilizamos la función **LocByName** para hallar la dirección de una función peligrosa dada por su nombre,

```
funcion = LocByName (funcion_peligrosa);
```

Si es encontrada en una dirección dicha función, se ejecuta un bucle el cual procesa todas las referencias cruzadas a la función peligrosa.

```
for (direccion = RfirstB (funcion); direccion != BADADDR; direccion = RnextB (funcion,
direccion))
```

Para cada referencia cruzada,

```
refcruzada = XrefType ( );
```

y si el tipo de referencia cruzada coincide,

```
if (refcruzada == fl_CN || refcruzada == fl_CF)
```

se determina el nombre de la función llamada

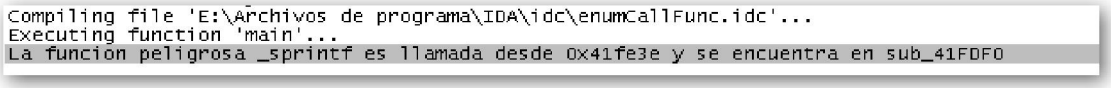
```
fuente = GetFunctionName (direccion);
```

y es mostrada al usuario

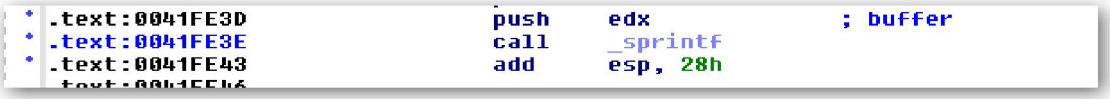
Message (“La funcion peligrosa %s es llamada desde 0x%x en la dirección %s\n”, funcion_llamada, direccion, fuente);

Bien como siempre cogeremos nuestro script, que llamaremos **enumCallFunc.idc** y lo colocaremos en la carpeta **idc**, en este caso para ver actuar a nuestro script cargaremos el programa **ar2idt.exe** de las **idsutils v5.10** en IDA y ejecutamos el script con el siguiente resultado:

La primera función buscada en el programa es **strcpy**, y como esta no existe el diálogo “**Warning**” nos lo indica. Aceptamos y el script sigue su ejecución designada.



Con la siguiente función que es **sprintf**, tenemos más suerte y en la ventana mensajes de Ida nos informa que es llamada en **0041fE3E** y que ella se encuentra en **sub_41FDF0**.



Ahora ya sabemos donde empezar a investigar, y averiguar si el programador, como ya apuntamos ha tenido la buena idea de verificar los tamaños de los buffers y no permitir un posible buffer overflow.

Es importante remarcar que pueden requerirse algunas transformaciones para ejecutar una búsqueda apropiada del nombre de una función importada. En especial en los ejecutables **ELF**, ya que combinan una tabla de procedimiento de enlace (**PLT**) con una tabla de Offset global (**GOT**) para manejar los detalles de enlace a las librerías compartidas, pues los nombres asignados por ida a las funciones importadas pueden ser poco claros. Por ejemplo, una entrada **PLT** puede aparecer nombrada como **_memcpy**, cuando de hecho es nombrada **.memcpy**, esto es debido a que ida reemplaza el punto por un guión porque considera que los puntos son un carácter inválido dentro de los nombres. Si lo complicamos más nos encontramos con el hecho de que ida puede crear un símbolo llamado **memcpy** el cual reside en una sección llamada por ida **extern**. Al intentar enumerar las referencias cruzadas a **memcpy**, nos interesará la versión **PLT** del símbolo ya que es la versión que será llamada desde otras funciones en el programa, por lo tanto la versión a la cual todas las referencias cruzadas se referirán.

Performance Bigundill@