Application Binary Interface

Hacking Ético

©Ismael Ripoll &
Hector Marco

Universidad Politècnica de València

February 1, 2022

Índice

Qué vamos a trabajar

- Conocer el ABI de la arquitectura x86 (i386) de Linux.
- Ver la relación entre el código “C” y el ensamblador.
- Adquirir práctica en el uso del GDB.
- Familiarizarse con el espacio de memoria virtual donde se ejecutan los procesos.

El ABI

- Antes de empezar la práctica busca información sobre el ABI del sistema.
- Busca en la Wikipedia la entrada “x86 calling conventions”.
- Estudia la hoja de manual de las llamadas al sistema (`man syscall`).

Llamadas a función

- Escribe un programa en “C” que contenga una función que reciba muchos parametros (enteros) y los imprima.
- Utiliza el GDB para ver los valores de los registros al inicio de ejecución de esa función.
- Comprueba que el `calling convention` utilizado por el GCC es el esperado.
- Escribe una función cuadrado (calcula el cuadrado del número que se le pasa) y comprueba que efectivamente, el valor devuelto por la función está en el registro `rax`.
- ¿Es el mismo el calling convention del kernel de Linux que el usado por las librerías?

La Pila (I)

- Escribe un programa (pila.c) con 4 funciones donde cada una llame a la siguiente.
- Compílalo con el flag "`-fomit-frame-pointer`" para simplificar la estructura de la pila.

```
$ gcc -no-pie -fno-pie -m32 -O0 -fomit-frame-pointer pila.
    c -o pila
```

- Síguelo paso a paso y mira el contenido de la pila cada vez que se entra en una de las funciones que has creado.
- Utiliza tanto los comandos `stepi` para seguir el programa por nuestras funciones como `next` para llamar seguirlo cuando se llama a funciones de librería.
- Recuerda que estamos en una arquitectura de 32 bits. Utiliza el modificar "`w`". Por ejemplo:

La Pila (II)

```
(gdb) x /10xw $rsp
```

- Compara las direcciones que hay en la pila con las direcciones del programa.
- ¿Dónde apuntan las direcciones de la pila?
- ¿Qué pasa si cambias una de las direcciones de la pila?

Variables (I)

- Escribe un programa con una variable de tipo `volatile int` global y otra local a `main`.
- Puedes utilizar el GDB para averiguar "donde vive cada tipo de variable" pero también puede imprimir las direcciones desde el propio programa. Puedes utilizar el formato "`p`" de printf:

```
printf ("Variable direccion: %p\n", &la_global);
```

- ¿Dónde está cada variable?
- ¿Qué diferencia hay entre las variables `volatile` y las normales?
- Ahora declara una variable entera para utilizarla solo como contador de un bucle (no imprimas su dirección).
- ¿Dónde está esa variable? Revisa el ABI.

Variables (II)

- Declara un arrays de enteros local e inicialízalo con un bucle. Inspecciona el contenido del array conforme se va inicializando con el bucle.

ELF (I)

- El formato de los ficheros ejecutables también forma parte del ABI. Utiliza el comando `readelf` para mostrar el contenido de un fichero ELF.
- Se suelen llamar *ficheros objeto* a los que contienen código máquina.
- El parámetro "`-aW`" sirve para mostrar toda la información.
- CUIDADO: No es necesario comprender casi nada de todo lo que muestra. Solo:
  - Mira la información del encabezado: clase ejecutable, tipo de ELF, dirección de entrada.
  - Fíjate también en las funciones y variables que utiliza el programa. Esta información es necesaria para poder utilizar las funciones de librería.

Objdump (I)

- El comando "`objdump`" sirve para mostrar el contenido de los ficheros objeto.
- El parámetro "`-d`" muestra los memotécnicos (desensamblados) de las secciones ejecutables.
- **objdump -d** es un desensamblador.
- También puedes usar el desemsamblador "`radare2`". Pero es bastante más complicado de usar.

Actividades (I)

1. Realiza los pasos que se indican en la práctica.
2. ¿Qué es el base frame pointer? ¿Para qué sirve? Compila un programa compilado con y sin frame pointer. Compara los resultados.
3. ¿Es necesario el frame pointer?
4. Compila los programas desarrollados con y sin el flag de optimización "`gcc -O2 pila.c -o pila_opt`". Compara el código generado con y sin optimización.
5. Haz un esquema del mapa de memoria (en Gigabytes) de alguno de tus programas.