

# SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software  
Curso 2019-20

---

**Práctica [2].** Ingeniería inversa y vulnerabilidades.

**Sesión [1].** El formato ELF (Executable and Linkable Format) en Linux.

**Autor<sup>1</sup>:** Nazaret Román Guerrero

---

## Ejercicio 1.

---

Construye y compila un programa simple, por ejemplo uno similar al “Hola, Mundo”, en dos versiones: una en C y otra en C++.

- a) Consulta los manuales o en Internet qué contienen las secciones `.interp`, `.got` y `.got.plt`.
- b) Compara los ELF de las dos versiones listando las secciones. ¿Hay alguna diferencia relevante respecto a las secciones de un programa compilado con gcc? ¿Qué contienen las secciones `.ctors` y `.dtors`?
- c) Con la opción `readelf -r` podemos ver las secciones de reubicación. Indicar qué contienen estas secciones.
  - a) Cada sección contiene:
    - `.interp`: contiene el path del intérprete del programa. Si el archivo contiene un segmento cargable que incluya la reubicación, incluirá `SHF_ALLOC` entre sus atributos.
    - `.got`: contiene los offsets para la reubicación de variables globales.
    - `.got.plt`: contiene los offsets utilizados para llamar a funciones y procedimientos externos al programa.
  - b) Los programas hechos son dos “Hola Mundo” en C y en C++. Las secciones de cada uno se verán a continuación. Antes de nada, para poder compilar los programas es necesario tener instalado en el sistema el compilador de C, gcc, y el de C++, g++.

El código de cada programa está añadido al final de este documento, en un anexo con los códigos para los distintos ejercicios.

El listado de secciones del programa en C es el siguiente:

---

1 Como autor declaro que los contenidos del presente documento son originales y elaborados por mí. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la “Normativa de evaluación y de calificaciones de los estudiantes de la Universidad de Granada” esto “conllevará la calificación numérica de cero ... independientemente del resto de calificaciones que el estudiante hubiera obtenido ...”

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
nazaretroque@nazaretroque:~$ readelf -S hello_c
There are 29 section headers, starting at offset 0x1930:

Encabezados de Sección:
[Nr] Nombre                               Tipo          Dirección      Despl
     Tamaño                               TamEnt        Opts  Enl   Info  Alin
[ 0]                                     NULL          0  0  0
0000000000000000 0000000000000000 0 0 0
[ 1] .interp                                PROGBITS      0000000000000238 00000238
000000000000001c 0000000000000000 A 0 0 1
[ 2] .note.ABI-tag                          NOTE          0000000000000254 00000254
0000000000000020 0000000000000000 A 0 0 4
[ 3] .note.gnu.build-id                     NOTE          0000000000000274 00000274
0000000000000024 0000000000000000 A 0 0 4
[ 4] .gnu.hash                              GNU_HASH      0000000000000298 00000298
000000000000001c 0000000000000000 A 5 0 8
[ 5] .dynsym                                DYNSYM        00000000000002b8 000002b8
00000000000000a8 0000000000000018 A 6 1 8
[ 6] .dynstr                                STRTAB        0000000000000360 00000360
0000000000000082 0000000000000000 A 0 0 1
[ 7] .gnu.version                           VERSYM        00000000000003e2 000003e2
000000000000000e 0000000000000002 A 5 0 2
[ 8] .gnu.version_r                         VERNEED       00000000000003f0 000003f0
0000000000000020 0000000000000000 A 6 1 8
[ 9] .rela.dyn                              RELA          0000000000000410 00000410
00000000000000c0 0000000000000018 A 5 0 8
[10] .rela.plt                              RELA          00000000000004d0 000004d0
0000000000000018 0000000000000018 AI 5 22 8
[11] .init                                   PROGBITS      00000000000004e8 000004e8
0000000000000017 0000000000000000 AX 0 0 4
[12] .plt                                   PROGBITS      0000000000000500 00000500
0000000000000020 0000000000000010 AX 0 0 16
[13] .plt.got                               PROGBITS      0000000000000520 00000520
0000000000000008 0000000000000008 AX 0 0 8
[14] .text                                  PROGBITS      0000000000000530 00000530
00000000000001a2 0000000000000000 AX 0 0 16

[15] .finit                                 PROGBITS      00000000000006d4 000006d4
0000000000000009 0000000000000000 AX 0 0 4
[16] .rodata                               PROGBITS      00000000000006e0 000006e0
0000000000000011 0000000000000000 A 0 0 4
[17] .eh_frame_hdr                         PROGBITS      00000000000006f4 000006f4
000000000000003c 0000000000000000 A 0 0 4
[18] .eh_frame                             PROGBITS      0000000000000730 00000730
0000000000000108 0000000000000000 A 0 0 8
[19] .init_array                           INIT_ARRAY    0000000000000db8 00000db8
0000000000000008 0000000000000008 WA 0 0 8
[20] .fini_array                           FINI_ARRAY    0000000000000dc0 00000dc0
0000000000000008 0000000000000008 WA 0 0 8
[21] .dynamic                              DYNAMIC       0000000000000dc8 00000dc8
00000000000001f0 0000000000000010 WA 6 0 8
[22] .got                                   PROGBITS      0000000000000fb8 00000fb8
0000000000000048 0000000000000008 WA 0 0 8
[23] .data                                  PROGBITS      0000000000001000 00001000
000000000000010 0000000000000000 WA 0 0 8
[24] .bss                                  NOBITS        0000000000001010 00001010
0000000000000008 0000000000000000 WA 0 0 1
[25] .comment                              PROGBITS      00000000000001010 00001010
000000000000002b 0000000000000001 MS 0 0 1
[26] .symtab                               SYMTAB        00000000000001040 00001040
000000000000005e8 0000000000000018 27 43 8
[27] .strtab                               STRTAB        00000000000001628 00001628
00000000000000203 0000000000000000 0 0 1
[28] .shstrtab                             STRTAB        0000000000000182b 0000182b
00000000000000fe 0000000000000000 0 0 1

Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
l (large), p (processor specific)
nazaretroque@nazaretroque:~$

```

El listado de secciones para el programa en C++ es el que sigue:

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
nazaretroque@nazaretroque:~$ readelf -S hello_cpp
There are 29 section headers, starting at offset 0x1b98:

Encabezados de Sección:
[Nr] Nombre                               Tipo          Dirección      Despl
     Tamaño                               TamEnt        Opts  Enl   Info  Alin
[ 0]                                     NULL          0  0  0
0000000000000000 0000000000000000 0 0 0
[ 1] .interp                                PROGBITS      0000000000000238 00000238
000000000000001c 0000000000000000 A 0 0 1
[ 2] .note.ABI-tag                          NOTE          0000000000000254 00000254
0000000000000020 0000000000000000 A 0 0 4
[ 3] .note.gnu.build-id                     NOTE          0000000000000274 00000274
0000000000000024 0000000000000000 A 0 0 4
[ 4] .gnu.hash                              GNU_HASH      0000000000000298 00000298
0000000000000024 0000000000000000 A 5 0 8
[ 5] .dynsym                                DYNSYM        00000000000002c0 000002c0
0000000000000138 0000000000000018 A 6 1 8
[ 6] .dynstr                                STRTAB        00000000000003f8 000003f8
0000000000000163 0000000000000000 A 0 0 1
[ 7] .gnu.version                           VERSYM        000000000000055c 0000055c
000000000000001a 0000000000000002 A 5 0 2
[ 8] .gnu.version_r                         VERNEED       0000000000000578 00000578
0000000000000040 0000000000000000 A 6 2 8
[ 9] .rela.dyn                              RELA          00000000000005b8 000005b8
0000000000000120 0000000000000018 A 5 0 8
[10] .rela.plt                              RELA          00000000000006d8 000006d8
0000000000000060 0000000000000018 AI 5 22 8
[11] .init                                   PROGBITS      0000000000000738 00000738
0000000000000017 0000000000000000 AX 0 0 4
[12] .plt                                   PROGBITS      0000000000000750 00000750
0000000000000050 0000000000000010 AX 0 0 16
[13] .plt.got                               PROGBITS      00000000000007a0 000007a0
0000000000000008 0000000000000008 AX 0 0 8
[14] .text                                  PROGBITS      00000000000007b0 000007b0
0000000000000212 0000000000000000 AX 0 0 16

[15] .finit                                 PROGBITS      00000000000009c4 000009c4
0000000000000009 0000000000000000 AX 0 0 4
[16] .rodata                               PROGBITS      00000000000009d0 000009d0
0000000000000012 0000000000000000 A 0 0 4
[17] .eh_frame_hdr                         PROGBITS      00000000000009e4 000009e4
000000000000004c 0000000000000000 A 0 0 4
[18] .eh_frame                             PROGBITS      0000000000000a30 00000a30
0000000000000148 0000000000000000 A 0 0 8
[19] .init_array                           INIT_ARRAY    0000000000000d78 00000d78
0000000000000010 0000000000000008 WA 0 0 8
[20] .fini_array                           FINI_ARRAY    0000000000000d88 00000d88
0000000000000008 0000000000000008 WA 0 0 8
[21] .dynamic                              DYNAMIC       0000000000000d90 00000d90
0000000000000200 000000000000010 WA 6 0 8
[22] .got                                   PROGBITS      0000000000000f90 00000f90
0000000000000070 0000000000000008 WA 0 0 8
[23] .data                                  PROGBITS      0000000000001000 00001000
000000000000010 0000000000000000 WA 0 0 8
[24] .bss                                  NOBITS        0000000000001020 00001010
0000000000000118 0000000000000000 WA 0 0 32
[25] .comment                              PROGBITS      00000000000001010 00001010
000000000000002b 0000000000000001 MS 0 0 1
[26] .symtab                               SYMTAB        00000000000001040 00001040
000000000000006d8 0000000000000018 27 47 8
[27] .strtab                               STRTAB        00000000000001718 00001718
00000000000000382 0000000000000000 0 0 1
[28] .shstrtab                             STRTAB        000000000000019a9 000019a9
00000000000000fe 0000000000000000 0 0 1

Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
l (large), p (processor specific)
nazaretroque@nazaretroque:~$

```

Como se puede observar, no hay diferencias respecto a las secciones; las únicas diferencias se corresponden, como es normal, a la ubicación del programa en memoria, sus variables y procedimientos, ya que son dos programas diferentes.

La sección .ctors contiene una lista global con punteros a los constructores, no obstante no aparece dicha sección aquí porque no hay ninguna declaración de variables ni clases creadas con constructores definidos.

La sección `.dtors` contiene una lista con los destructores, pero al igual que con `.ctors` no existe dicha sección porque no hay destructores declarados en el programa ni se están destruyendo variables de otras clases.

c) La salida del comando `readelf -r` para ambos programas es diferente:

- Para el programa en C la salida es la siguiente:

```
nazaretroque
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ readelf -r hello_c

La sección de reubicación '.rela.dyn' at offset 0x410 contains 8 entries:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200db8  000000000008  R_X86_64_RELATIVE          630
000000200dc0  000000000008  R_X86_64_RELATIVE          5f0
000000201008  000000000008  R_X86_64_RELATIVE          201008
000000200fd8  000100000006  R_X86_64_GLOB_DAT 0000000000000000 __ITM_deregisterTMClone + 0
000000200fe0  000300000006  R_X86_64_GLOB_DAT 0000000000000000 __libc_start_main@GLIBC_2.2.5 + 0
000000200fe8  000400000006  R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000200ff0  000500000006  R_X86_64_GLOB_DAT 0000000000000000 __ITM_registerTMCloneTa + 0
000000200ff8  000600000006  R_X86_64_GLOB_DAT 0000000000000000 __cxa_finalize@GLIBC_2.2.5 + 0

La sección de reubicación '.rela.plt' at offset 0x4d0 contains 1 entry:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200fd0  000200000007  R_X86_64_JUMP_SLO 0000000000000000 puts@GLIBC_2.2.5 + 0
nazaretroque@nazaretroque:~$
```

- Para el programa en C++ la salida es:

```
nazaretroque@nazaretroque
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ readelf -r hello_cpp

La sección de reubicación '.rela.dyn' at offset 0x5b8 contains 12 entries:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200d78  000000000008  R_X86_64_RELATIVE          8b0
000000200d80  000000000008  R_X86_64_RELATIVE          936
000000200d88  000000000008  R_X86_64_RELATIVE          870
000000201008  000000000008  R_X86_64_RELATIVE          201008
000000200fc8  000100000006  R_X86_64_GLOB_DAT 0000000000000000 __cxa_finalize@GLIBC_2.2.5 + 0
000000200fd0  000200000006  R_X86_64_GLOB_DAT 0000000000000000 __ZSt4endl1cSt11char_tr@GLIBCXX_3.4 + 0
000000200fd8  000700000006  R_X86_64_GLOB_DAT 0000000000000000 __ITM_deregisterTMClone + 0
000000200fe0  000800000006  R_X86_64_GLOB_DAT 0000000000000000 __libc_start_main@GLIBC_2.2.5 + 0
000000200fe8  000900000006  R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000200ff0  000a00000006  R_X86_64_GLOB_DAT 0000000000000000 __ITM_registerTMCloneTa + 0
000000200ff8  000b00000006  R_X86_64_GLOB_DAT 0000000000000000 __ZNSt8ios_base4InitD1E@GLIBCXX_3.4 + 0
000000201020  000c00000005  R_X86_64_COPY      0000000000201020 __ZSt4cout@GLIBCXX_3.4 + 0

La sección de reubicación '.rela.plt' at offset 0x6d8 contains 4 entries:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200fa8  000300000007  R_X86_64_JUMP_SLO 0000000000000000 __cxa_atexit@GLIBC_2.2.5 + 0
000000200fb0  000400000007  R_X86_64_JUMP_SLO 0000000000000000 __ZStlsISt11char_traits@GLIBCXX_3.4 + 0
000000200fb8  000500000007  R_X86_64_JUMP_SLO 0000000000000000 __ZNSoIsEPFRSoS_E@GLIBCXX_3.4 + 0
000000200fc0  000600000007  R_X86_64_JUMP_SLO 0000000000000000 __ZNSt8ios_base4InitC1E@GLIBCXX_3.4 + 0
nazaretroque@nazaretroque:~$
```

En ambos programas, hay dos secciones, de reubicación `.rela.dyn` y `.rela.plt`.

La primera, `.rela.dyn` (runtime/dynamic relocation table), contiene la información sobre la reubicación de las variables en tiempo de carga para aquellos ejecutables que utilizan memoria dinámica.

La segunda, `.rela.plt` (runtime/dynamic relocation table), es similar a `.rela.dyn` con la diferencia de que es para funciones y procedimientos en lugar de variables.



## Ejercicio 2.

---

Mira en el manual en línea o en Internet las opciones de la orden `objdump`, e indica:

- a) qué opciones nos permiten ver la información que nos suministra `readelf`,
  - b) y qué otras opciones nos permite realizar `objdump` desde el punto de la ingeniería inversa.
- a) Con `objdump` podemos mostrar la información de archivos `.obj`.

Si lo que queremos es extraer por ejemplo la información que muestra la orden `readelf -r`, que muestra las secciones de reubicación, utilizando `objdump`, debemos utilizar la opción `-r` y `-R` (o `--reloc` y `--dynamic-reloc`) que muestran la relocation y la dynamic relocation respectivamente, por lo que sería algo como

- `readelf -r <archivo> = objdump -rR <archivo>`

Si creemos mostrar toda la información igual que lo hace la orden `readelf -a`, el equivalente sería utilizar las siguientes opciones:

- `-f` o `--file-headers`
- `-x` o `--all-headers`
- `-j <section name>` o `--section=<section name>`
- `-t` o `--syms`
- `-T` o `--dynamic-syms`
- `-r` o `--reloc`
- `-R` o `--dynamic-reloc`
- `-V` o `--version`
- `-m <machine architecture>` o `--architecture=<machine>`

De modo que el comando quedaría como

- `readelf -a <archivo> = objdump -fxtTrRV -j <section> -m <architecture> <archivo>`

- b) La orden `objdump` nos permite desensamblar un archivo `.obj` lo que nos permite ver las instrucciones en ensamblador del programa según la máquina en la que se haya compilado. Esto es muy útil a la hora de intentar descubrir el funcionamiento de un programa cuando solo tenemos el binario, y es muy importante en ingeniería inversa precisamente por eso, porque analizando las instrucciones podemos saber si es, por ejemplo, un malware.

Las opciones que permiten desensamblar son `-d`, `-D` y `-z`:

- `-d` permite desensamblar el programa y sacar aquellas instrucciones que llevan a cabo una función dentro de éste.
- `-D` desensambla el código completo, no solo las partes que contienen instrucciones, es decir, cuando se utiliza esta opción, la suposición de que los símbolos presentes más allá de los límites de las instrucciones no se desensamblan desaparece, por lo que se está desensamblando más allá del límite del código del propio programa.

- -z desensambla todo el código, incluyendo los bloques rellenos de 0 del programa, que por lo general no se muestran en la salida del desensamblado.

### Ejercicio 3.

Modifica el programa realizado en el ejercicio anterior para que el programa se detenga durante un rato, por ejemplo, con un `sleep()`, al objeto de que podamos visualizar el archivo `/proc/<PID>/maps` de su ejecución. Ahora analiza la información de su ELF para entrever cómo se ha construido dicho proceso a través de la información del ELF, por ejemplo, las direcciones y permisos de las regiones de texto y datos, etc.

El código con la correspondiente modificación se encuentran en el anexo, al final de este mismo documento.

Vamos a ejecutar la versión en C. Para que no se quede la terminal colgada mientras ejecuta el programa, lo lanzamos en background:

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ gcc hello.c -o hello_c_mod
nazaretroque@nazaretroque:~$ ./hello_c_mod&
[1] 25866
```

El PID es 25866, que nos sirve para buscar el archivo `/proc/25866/maps` que necesitamos, cuya salida es la siguiente:

```
nazaretroque@nazaretroque:~$ cat /proc/25866/maps
55ae02a20000-55ae02a21000 r-xp 00000000 08:01 285450
55ae02c20000-55ae02c21000 r--p 00000000 08:01 285450
55ae02c21000-55ae02c22000 rw-p 00001000 08:01 285450
55ae02f26000-55ae02f47000 rw-p 00000000 00:00 0
7ff892c7f000-7ff892e66000 r-xp 00000000 08:01 399362
7ff892e66000-7ff893066000 --p 001e7000 08:01 399362
7ff893066000-7ff89306a000 r--p 001e7000 08:01 399362
7ff89306a000-7ff89306c000 rw-p 001eb000 08:01 399362
7ff89306c000-7ff893070000 rw-p 00000000 00:00 0
7ff893070000-7ff893097000 r-xp 00000000 08:01 399334
7ff893280000-7ff893282000 rw-p 00000000 00:00 0
7ff893297000-7ff893298000 r--p 00027000 08:01 399334
7ff893298000-7ff893299000 rw-p 00028000 08:01 399334
7ff893299000-7ff89329a000 rw-p 00000000 00:00 0
7fff5ce97000-7fff5ceb8000 rw-p 00000000 00:00 0
7fff5cf47000-7fff5cf4a000 r--p 00000000 00:00 0
7fff5cf4a000-7fff5cf4b000 r-xp 00000000 00:00 0
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0
/home/nazaretroque/hello_c_mod
/home/nazaretroque/hello_c_mod
/home/nazaretroque/hello_c_mod
[heap]
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/ld-2.27.so
/lib/x86_64-linux-gnu/ld-2.27.so
/lib/x86_64-linux-gnu/ld-2.27.so
[stack]
[vvar]
[vdso]
[vsyscall]
```

Comparando con la salida del comando `readelf -S`, podemos ver que la primera línea y segunda línea del archivo `maps` se corresponden con el código y los datos del programa, puesto que el desplazamiento es 0, como se ve en la imagen:

```
nazaretroque@nazaretroque:~$ readelf -S hello_c_mod
There are 29 section headers, starting at offset 0x1958:

Encabezados de Sección:
[Nr] Nombre          Tipo          Dirección      Despl
      Tamaño          TamEnt        Opts  Enl  Info  Alin
[ 0] 0000000000000000 NULL          0000000000000000 0 0 0
```

También podemos apreciar que la tercera línea de maps se corresponde con las variables estáticas del programa, puesto que el desplazamiento es 1000:

```
[23] .data          PROGBITS          00000000000201000 00001000
      00000000000000010 00000000000000000 WA          0          0          8
```

Estas secciones tienen permisos de lectura y ejecución y mapeo privado (el código), lectura y mapeo privado para los datos del programa y lectura y escritura y mapeo privado para las variables estáticas.

## Anexo.

---

```
#include <stdio.h>

int main(){
    printf("Hello world!\n");

    return 0;
}
```

*Figure 1: hello.c*

```
#include <iostream>

using namespace std;

int main(){
    cout << "Hello world! << endl;

    return 0;
}
```

*Figure 2: hello.cpp*

```
#include <stdio.h>
#include <unistd.h>

int main(){
    printf("Hello world!\n");
    sleep(1000); # El programa queda bloqueado durante 1000 segundos, unos 16 minutos
    return 0;
}
```

*Figure 3: hello\_modificado.c*

## Bibliografía.

---

- <http://bottomupcs.sourceforge.net/csbu/x3824.htm>
- <https://reverseengineering.stackexchange.com/questions/1992/what-is-plt-got>
- <https://stackoverflow.com/questions/11676472/what-is-the-difference-between-got-and-got-plt-section>
- <https://www.cs.stevens.edu/~jschauma/631A/elf.html>