

SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software
Curso 2018-19

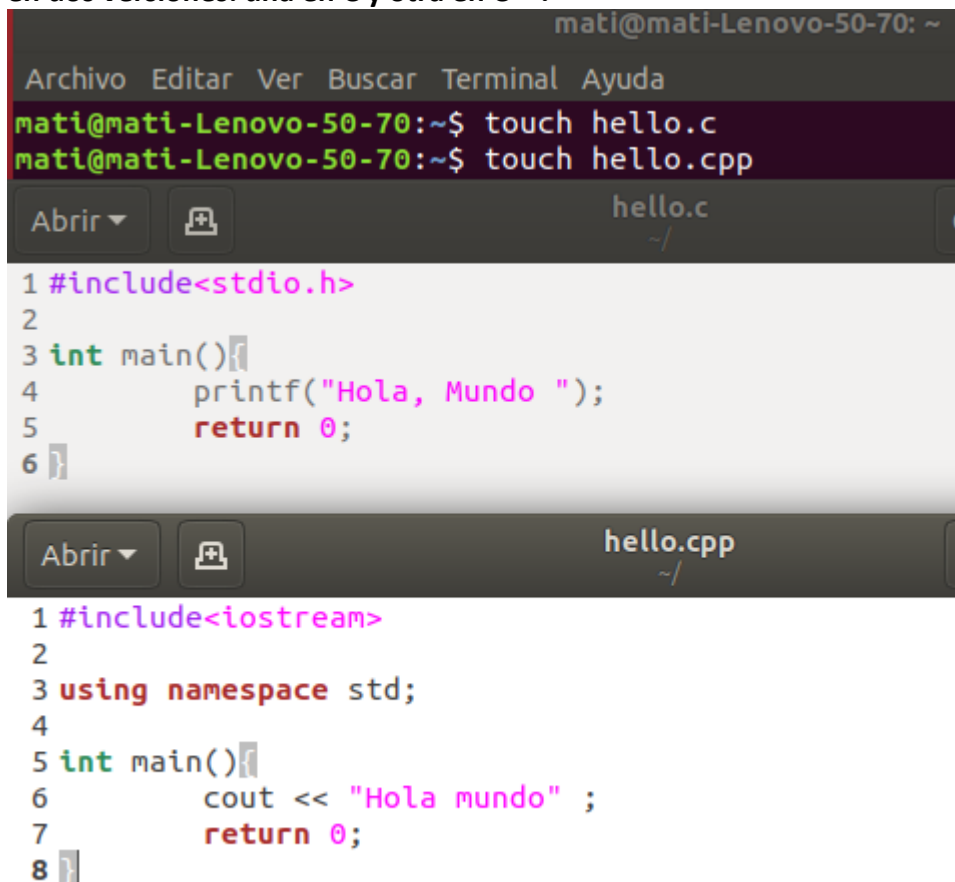
Práctica [2]. Ingeniería inversa y vulnerabilidades

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

Autor¹: Matilde Cabrera González

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++.



```
mati@mati-Lenovo-50-70: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
mati@mati-Lenovo-50-70:~$ touch hello.c  
mati@mati-Lenovo-50-70:~$ touch hello.cpp  
Abrir ▾ hello.c  
1 #include<stdio.h>  
2  
3 int main()  
4     printf("Hola, Mundo ");  
5     return 0;  
6 }  
Abrir ▾ hello.cpp  
1 #include<iostream>  
2  
3 using namespace std;  
4  
5 int main()  
6     cout << "Hola mundo" ;  
7     return 0;  
8 }
```

¹ Como autor declaro que los contenidos del presente documento son originales y elaborados por mi. 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 ...”

(a) Consulta los manuales, o en Internet que contienen las secciones `.interp`, `.got`, `got.plt`.

INTERP se refiere a la ruta de un intérprete de programa, es decir, el programa incluye una reubicación de forma dinámica.

GOT se refiere a la tabla de variables globales que se usan dentro de un programa, que se usaran en tiempo de ejecución

PLT se refiere a la tabla de vinculación de procedimientos, que se utiliza, sencillamente, para llamar a procedimientos / funciones externas cuya dirección no se conoce en el momento de la vinculación, y el vinculador dinámico debe resolverla en tiempo de ejecución.

(b) Compara los ELFs 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`?

```
matl@matl-Lenovo-50-70:~$ objdump -h helloc hellocpp

helloc:      formato del fichero elf64-x86-64

Secciones:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .interp         0000001c  0000000000000238 0000000000000238 00000238 2**0
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.ABI-tag   00000020  0000000000000254 0000000000000254 00000254 2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .note.gnu.build-id 00000024  0000000000000274 0000000000000274 00000274 2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .gnu.hash       0000001c  0000000000000298 0000000000000298 00000298 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .dynsym         000000a8  00000000000002b8 00000000000002b8 000002b8 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynstr         00000084  0000000000000360 0000000000000360 00000360 2**0
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .gnu.version    0000000e  00000000000003e4 00000000000003e4 000003e4 2**1
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .gnu.version_r  00000020  00000000000003f8 00000000000003f8 000003f8 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .rela.dyn       000000c0  0000000000000418 0000000000000418 00000418 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .rela.plt       00000018  00000000000004d8 00000000000004d8 000004d8 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
10 .init           00000017  00000000000004f0 00000000000004f0 000004f0 2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
11 .plt            00000020  0000000000000510 0000000000000510 00000510 2**4
    CONTENTS, ALLOC, LOAD, READONLY, CODE
12 .plt.got        00000008  0000000000000530 0000000000000530 00000530 2**3
    CONTENTS, ALLOC, LOAD, READONLY, CODE
13 .text           000001a2  0000000000000540 0000000000000540 00000540 2**4
    CONTENTS, ALLOC, LOAD, READONLY, CODE
14 .fini           00000009  00000000000006e4 00000000000006e4 000006e4 2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
15 .rodata         00000011  00000000000006f0 00000000000006f0 000006f0 2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
16 .eh_frame_hdr   0000003c  0000000000000704 0000000000000704 00000704 2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
17 .eh_frame       00000108  0000000000000740 0000000000000740 00000740 2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
18 .init_array     00000008  0000000000200db8 0000000000200db8 00000db8 2**3
    CONTENTS, ALLOC, LOAD, DATA
19 .fini_array     00000008  0000000000200dc0 0000000000200dc0 00000dc0 2**3
    CONTENTS, ALLOC, LOAD, DATA
20 .dynamic        000001f0  0000000000200dc8 0000000000200dc8 00000dc8 2**3
    CONTENTS, ALLOC, LOAD, DATA
```

20	.dynamic	000001f0	00000000000200dc8	00000000000200dc8	00000dc8	2**3
	CONTENTS, ALLOC, LOAD, DATA					
21	.got	00000048	00000000000200fb8	00000000000200fb8	00000fb8	2**3
	CONTENTS, ALLOC, LOAD, DATA					
22	.data	00000010	00000000000201000	00000000000201000	00001000	2**3
	CONTENTS, ALLOC, LOAD, DATA					
23	.bss	00000008	00000000000201010	00000000000201010	00001010	2**0
	ALLOC					
24	.comment	0000002a	0000000000000000	0000000000000000	00001010	2**0
	CONTENTS, READONLY					

hellocpp: formato del fichero elf64-x86-64

Secciones:

Idx	Name	Size	VMA	LMA	File off	Algn
0	.interp	0000001c	00000000000000238	00000000000000238	00000238	2**0
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
1	.note.ABI-tag	00000020	00000000000000254	00000000000000254	00000254	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
2	.note.gnu.build-id	00000024	00000000000000274	00000000000000274	00000274	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
3	.gnu.hash	00000024	00000000000000298	00000000000000298	00000298	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
4	.dynsym	00000108	000000000000002c0	000000000000002c0	000002c0	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
5	.dynstr	00000117	000000000000003c8	000000000000003c8	000003c8	2**0
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
6	.gnu.version	00000016	000000000000004e0	000000000000004e0	000004e0	2**1
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
7	.gnu.version_r	00000040	000000000000004f8	000000000000004f8	000004f8	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
8	.rela.dyn	00000108	00000000000000538	00000000000000538	00000538	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
9	.rela.plt	00000048	00000000000000640	00000000000000640	00000640	2**3
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
10	.init	00000017	00000000000000688	00000000000000688	00000688	2**2
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
11	.plt	00000040	000000000000006a0	000000000000006a0	000006a0	2**4
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
12	.plt.got	00000008	000000000000006e0	000000000000006e0	000006e0	2**3
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
13	.text	00000202	000000000000006f0	000000000000006f0	000006f0	2**4
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
14	.fini	00000009	000000000000008f4	000000000000008f4	000008f4	2**2
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
15	.rodata	00000010	00000000000000900	00000000000000900	00000900	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					

15	.rodata	00000010	00000000000000900	00000000000000900	00000900	2**2
		CONTENTS, ALLOC, LOAD, READONLY, DATA				
16	.eh_frame_hdr	0000004c	00000000000000910	00000000000000910	00000910	2**2
		CONTENTS, ALLOC, LOAD, READONLY, DATA				
17	.eh_frame	00000148	00000000000000960	00000000000000960	00000960	2**3
		CONTENTS, ALLOC, LOAD, READONLY, DATA				
18	.init_array	00000010	0000000000200d88	0000000000200d88	00000d88	2**3
		CONTENTS, ALLOC, LOAD, DATA				
19	.fini_array	00000008	0000000000200d98	0000000000200d98	00000d98	2**3
		CONTENTS, ALLOC, LOAD, DATA				
20	.dynamic	00000200	0000000000200da0	0000000000200da0	00000da0	2**3
		CONTENTS, ALLOC, LOAD, DATA				
21	.got	00000060	0000000000200fa0	0000000000200fa0	00000fa0	2**3
		CONTENTS, ALLOC, LOAD, DATA				
22	.data	00000010	0000000000201000	0000000000201000	00001000	2**3
		CONTENTS, ALLOC, LOAD, DATA				
23	.bss	00000118	0000000000201020	0000000000201020	00001010	2**5
		ALLOC				
24	.comment	0000002a	0000000000000000	0000000000000000	00001010	2**0
		CONTENTS, READONLY				

He observado que no hay ninguna diferencia en lo que se refiere a información, si hay diferencia en las cabeceras que nos indica las direcciones de memoria de las secciones de los programas.

CTORS es el constructor y DTORS el destructor o recolector de basura, se especifican en la imagen anterior como .init_array e .fini_array y apuntan a la misma dirección de memoria.

(c) Con la opción `readelf -r` podemos ver las secciones de reubicación. Indicar que contienen estas secciones.

```
mati@mati-Lenovo-50-70:~$ readelf -r hellocpp

La sección de reubicación '.rela.dyn' at offset 0x538 contains 11 entries:
  Desplaz      Info      Tipo      Val. Símbolo  Nom. Símbolo + Adend
000000200d88  000000000008  R_X86_64_RELATIVE  7f0
000000200d90  000000000008  R_X86_64_RELATIVE  861
000000200d98  000000000008  R_X86_64_RELATIVE  7b0
000000201008  000000000008  R_X86_64_RELATIVE  201008
000000200fd0  000100000006  R_X86_64_GLOB_DAT  0000000000000000 __cxa_finalize@GLIBC_2.2.5 + 0
000000200fd8  000500000006  R_X86_64_GLOB_DAT  0000000000000000 __ITM_deregisterTMClone + 0
000000200fe0  000600000006  R_X86_64_GLOB_DAT  0000000000000000 __libc_start_main@GLIBC_2.2.5 + 0
000000200fe8  000700000006  R_X86_64_GLOB_DAT  0000000000000000 __gmon_start__ + 0
000000200ff0  000800000006  R_X86_64_GLOB_DAT  0000000000000000 __ITM_registerTMCloneTa + 0
000000200ff8  000900000006  R_X86_64_GLOB_DAT  0000000000000000 __ZNSt8ios_base4InitD1E@GLIBCXX_3.4
+ 0
000000201020  000a00000005  R_X86_64_COPY      0000000000201020 _ZSt4cout@GLIBCXX_3.4 + 0

La sección de reubicación '.rela.plt' at offset 0x640 contains 3 entries:
  Desplaz      Info      Tipo      Val. Símbolo  Nom. Símbolo + Adend
000000200fb8  000200000007  R_X86_64_JUMP_SLO  0000000000000000 __cxa_atexit@GLIBC_2.2.5 + 0
000000200fc0  000300000007  R_X86_64_JUMP_SLO  0000000000000000 _ZStlsISt11char_traits@GLIBCXX_3.4
+ 0
000000200fc8  000400000007  R_X86_64_JUMP_SLO  0000000000000000 _ZNSt8ios_base4InitC1E@GLIBCXX_3.4
+ 0
mati@mati-Lenovo-50-70:~$
```



```

mati@mati-Lenovo-50-70:~$ readelf -r helloc

La sección de reubicación '.rela.dyn' at offset 0x418 contains 8 entries:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200db8  000000000008  R_X86_64_RELATIVE          640
000000200dc0  000000000008  R_X86_64_RELATIVE          600
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 0x4d8 contains 1 entry:
  Desplaz      Info          Tipo          Val. Símbolo  Nom. Símbolo + Adend
000000200fd0  000200000007  R_X86_64_JUMP_SLO  0000000000000000  printf@GLIBC_2.2.5 + 0

```

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`.

```

mati@mati-Lenovo-50-70:~$ objdump --section helloc
Modo de empleo: objdump <opcion(es)> <fichero(s)>
Muestra la información de <fichero(s)> objeto.
Se requiere por lo menos una de las siguientes opciones:
-a, --archive-headers      Display archive header information
-f, --file-headers         Display the contents of the overall file header
-p, --private-headers      Display object format specific file header contents
-P, --private=OPT,OPT...  Display object format specific contents
-h, --[section-]headers   Display the contents of the section headers
-x, --all-headers          Display the contents of all headers
-d, --disassemble          Display assembler contents of executable sections
-D, --disassemble-all    Display assembler contents of all sections
-S, --source               Intermix source code with disassembly
-s, --full-contents        Display the full contents of all sections requested
-g, --debugging            Display debug information in object file
-e, --debugging-tags       Display debug information using ctags style
-G, --stabs                Display (in raw form) any STABS info in the file
-W[LIaprmfFsoRtUuTgAckK] or
--dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
=frames-interp,=str,=loc,=Ranges,=pubtypes,
=gdb_index,=trace_info,=trace_abbrev,=trace_aranges,
=addr,=cu_index,=links,=follow-links]
                        Display DWARF info in the file
-t, --syms                 Display the contents of the symbol table(s)
-T, --dynamic-syms         Display the contents of the dynamic symbol table
-r, --reloc                Display the relocation entries in the file
-R, --dynamic-reloc        Display the dynamic relocation entries in the file
@<file>                   Read options from <file>
-v, --version              Display this program's version number
-i, --info                 List object formats and architectures supported
-H, --help                 Display this information

```

`Objdump -a -f -h -p -P -t -x`, son ordenes que nos muestran información que también podemos obtener con `readelf`.

(b) qué otras opciones nos permiten realizar `objdump` desde el punto de la ingeniería inversa.

Podemos ver el código desensamblado de un programa. Con las opciones `-s, -D`.

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.

```
mati@mati-Lenovo-50-70:~$ ./helloworld &
[1] 4112
mati@mati-Lenovo-50-70:~$ cat /proc/4112/maps
55f57a6e4000-55f57a6e5000 r-xp 00000000 08:01 530228 /home/mati/hello
55f57a8e4000-55f57a8e5000 r--p 00000000 08:01 530228 /home/mati/hello
55f57a8e5000-55f57a8e6000 rw-p 00001000 08:01 530228 /home/mati/hello
55f57c551000-55f57c572000 rw-p 00000000 00:00 0 [heap]
7efdacf983000-7efdacf9a000 r-xp 00000000 08:01 400048 /lib/x86_64-linux-gnu/l
libc-2.27.so
7efdacf9a000-7efdacf9b000 ---p 001e7000 08:01 400048 /lib/x86_64-linux-gnu/l
libc-2.27.so
7efdacf9b000-7efdacf9c000 r--p 001e7000 08:01 400048 /lib/x86_64-linux-gnu/l
libc-2.27.so
7efdacf9c000-7efdacf9d000 rw-p 001eb000 08:01 400048 /lib/x86_64-linux-gnu/l
libc-2.27.so
7efdacf9d000-7efdacf9e000 rw-p 00000000 00:00 0
7efdacf9e000-7efdacf9f000 r-xp 00000000 08:01 400020 /lib/x86_64-linux-gnu/l
ld-2.27.so
7efdacf9f000-7efdacf00000 rw-p 00000000 00:00 0
7efdacf00000-7efdacf01000 r--p 00027000 08:01 400020 /lib/x86_64-linux-gnu/l
ld-2.27.so
7efdacf01000-7efdacf02000 rw-p 00028000 08:01 400020 /lib/x86_64-linux-gnu/l
ld-2.27.so
7efdacf02000-7efdacf03000 rw-p 00000000 00:00 0
7fffc86ac6000-7fffc86ae7000 rw-p 00000000 00:00 0 [stack]
7fffc86b07000-7fffc86b0a000 r--p 00000000 00:00 0 [vvar]
7fffc86b0a000-7fffc86b0c000 r-xp 00000000 00:00 0 [vdso]
fffffffff600000-fffffffff601000 r-xp 00000000 00:00 0 [vsyscall]
mati@mati-Lenovo-50-70:~$
```

r – lectura

w – escritura

x – ejecución

p – pagina privada

Primera sección, de código, tiene permisos r-xp, de datos rw-p, de BSS y de heap tienen rw-p.

Desensamblamos el código para analizar:

```
mati@mati-Lenovo-50-70:~$ objdump -d helloc
```

```
helloc:      formato del fichero elf64-x86-64
```

```
Desensamblado de la sección .init:
```

```
0000000000000528 <_init>:
 528:  48 83 ec 08          sub    $0x8,%rsp
 52c:  48 8b 05 b5 0a 20 00  mov    0x200ab5(%rip),%rax      # 200fe8 <__gmon_start__>
 533:  48 85 c0             test   %rax,%rax
 536:  74 02              je     53a <_init+0x12>
 538:  ff d0             callq  *%rax
 53a:  48 83 c4 08          add    $0x8,%rsp
 53e:  c3                retq
```

```
Desensamblado de la sección .plt:
```

```
0000000000000540 <.plt>:
 540:  ff 35 72 0a 20 00    pushq 0x200a72(%rip)          # 200fb8 <_GLOBAL_OFFSET_TABLE_+0x8>
 546:  ff 25 74 0a 20 00    jmpq   *0x200a74(%rip)        # 200fc0 <_GLOBAL_OFFSET_TABLE_+0x10>
 54c:  0f 1f 40 00          nopl   0x0(%rax)

0000000000000550 <printf@plt>:
 550:  ff 25 72 0a 20 00    jmpq   *0x200a72(%rip)        # 200fc8 <printf@GLIBC_2.2.5>
 556:  68 00 00 00 00      pushq  $0x0
 55b:  e9 e0 ff ff ff      jmpq   540 <.plt>

0000000000000560 <sleep@plt>:
 560:  ff 25 6a 0a 20 00    jmpq   *0x200a6a(%rip)        # 200fd0 <sleep@GLIBC_2.2.5>
 566:  68 01 00 00 00      pushq  $0x1
 56b:  e9 d0 ff ff ff      jmpq   540 <.plt>
```

```
Desensamblado de la sección .plt.got:
```