ELF[101] a Linux executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED FILE

```
~$ uname -m
x86_64
~$ ./simple64.elf
Hello World!
```

SIMPLE64.ELF

SHA-1: 9682FE77A50FE43EEFCC0920E2EA8DEE52E13EF6B38

DOWNLOAD @ ELF101.CORKAMI.COM

HEADER[1/2]
TECHNICAL DETAILS FOR IDENTIFICATION AND EXECUTION

ELF HEADER
IDENTIFY AS AN ELF TYPE
SPECIFY THE ARCHITECTURE

PROGRAM HEADER TABLE
EXECUTION INFORMATION

SECTIONS
CONTENTS OF THE EXECUTABLE

CODE
EXECUTABLE INFORMATION

DATA
INFORMATION USED BY THE CODE

SECTIONS' NAMES

HEADER[2/2]
TECHNICAL DETAILS FOR LINKING
(IGNORED FOR EXECUTION)

SECTION HEADER TABLE
LINKING (CONNECTING PROGRAM OBJECTS) INFORMATION

HEXADECIMAL DUMP | ASCII DUMP

```
7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00   .ELF............
02 00 3E 00 01 00 00 00 80 00 00 10 00 00 00 00   ..>.............
40 00 00 00 00 00 00 00 F0 00 00 00 00 00 00 00   @...............
00 00 00 00 40 00 38 00 01 00 40 00 04 00 03 00   ....@.8...@.....
```

1 e_ident

FIELDS	VALUES	EXPLANATION
EI_MAG	0x7F, "ELF"	CONSTANT SIGNATURE
EI_CLASS, EI_DATA	2 ELFCLASS64, 1 ELFDATA2LSB	64 BITS, LITTLE-ENDIAN
EI_VERSION	1 EV_CURRENT	ALWAYS 1
e_type	2 ET_EXEC	EXECUTABLE
e_machine	0x3E EM_AMD64	AMD 64 (AND LATER)
e_version	1 EV_CURRENT	ALWAYS 1
e_entry	0x10000080	ADDRESS WHERE EXECUTION STARTS
e_phoff	0x40	PROGRAM HEADERS' OFFSET
e_shoff	0xF0	SECTION HEADERS' OFFSET
e_ehsize	0x40	ELF HEADER'S SIZE
e_phentsize	0x38	SIZE OF A SINGLE PROGRAM HEADER
e_phnum	1	COUNT OF PROGRAM HEADERS
e_shentsize	0x40	SIZE OF A SINGLE SECTION HEADER
e_shnum	4	COUNT OF SECTION HEADERS
e_shstrndx	3*	INDEX OF THE NAMES' SECTION IN THE TABLE

2

```
Offset:0x40/Address:0x00000040
01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 10 00 00 00 00 00 00 00 00 10 00 00 00   ................
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00                           ........
```

FIELDS	VALUES	EXPLANATION
p_type	1 PT_LOAD	THE SEGMENT SHOULD BE LOADED IN MEMORY
p_flags	5 PF_READ_X	READABLE AND EXECUTABLE
p_offset	0	OFFSET WHERE IT SHOULD BE READ
p_vaddr	0x10000000	VIRTUAL ADDRESS WHERE IT SHOULD BE LOADED
p_paddr	0x10000000	PHYSICAL ADDRESS WHERE IT SHOULD BE LOADED
p_filesz	0xD0	SIZE ON FILE
p_memsz	0xD0	SIZE IN MEMORY

3

```
Offset:0x80/Address:0x00000080
48 BA 0D 00 00 00 00 00 00 00 48 C7 C6 C0 00 00   H.........H.....
10 48 C7 C7 01 00 00 00 48 C7 C0 01 00 00 00 0F   .H......H.......
05 48 C7 C7 01 00 00 00 48 C7 C0 3C 00 00 00 0F   .H......H..<....
05                                                .
```

X64 ASSEMBLY | EQUIVALENT C CODE

```
mov rdx, 0xD        (rdx: p_filesz)
mov rsi, 0x100000C0 (rsi: p_vaddr)
mov rdi, 1          (rdi: STDOUT_FILENO)
mov rax, 1          (rax: SC_WRITE)
syscall             (system call)        write(STDOUT_FILENO, "Hello World!\n", len("Hello World!\n"));

mov rdi, 0          (rdi: return code)
mov rax, 0x3C       (rax: SC_EXIT)
syscall             (system call)        exit(0);
```

```
Offset:0xC0/Address:0x00000C0
48 65 6C 6C 6F 20 57 6F 72 6C 64 21 0A 00         Hello.World!..
```

STRINGS

"Hello World!\n", 0

```
Offset:0xD0
00 2E 73 68 73 74 72 74 61 62 00 2E 74 65 78 74   ..shstrtab..text
00 2E 72 6F 64 61 74 61 00                        ..rodata.
```

SECTION NAMES

"" .shstrtab .text .rodata

```
Offset:0xF0
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0B 00 00 00 01 00 00 00 06 00 00 00 00 00 00 00   ................
80 00 00 10 00 00 00 00 80 00 00 00 00 00 00 00   ................
31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   1...............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
11 00 00 00 01 00 00 00 02 00 00 00 00 00 00 00   ................
C0 00 00 10 00 00 00 00 C0 00 00 00 00 00 00 00   ................
0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
01 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 D0 00 00 00 00 00 00 00   ................
19 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

SECTION HEADER TABLE

INDEX	NAME (sh_name)	TYPE (sh_type)	FLAGS (sh_flags)	ADDRESS (sh_addr)	OFFSET (sh_offset)	SIZE (sh_size)
0	<null>	0 SHT_NULL (inactive)				
1	.text	1 SHT_PROGBITS (program)	6 SHF_ALLOC (allocated), SHF_EXECINSTR (executable)	0x10000080	0x80	0x31
2	.rodata	1 SHT_PROGBITS (program)	2 SHF_ALLOC (allocated)	0x100000C0	0xC0	0x0D
3*	.shstrtab	3 SHT_STRTAB (string table)			0xD0	0x19

* relative offsets in names' section

THIS IS THE WHOLE FILE, HOWEVER, MOST ELF FILES CONTAIN MANY MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED, FOR CONCISENESS.

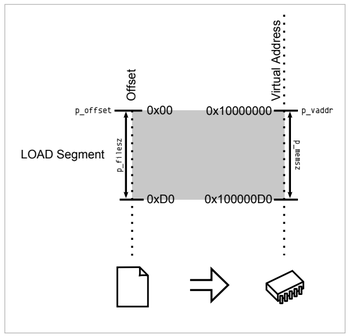
LOADING PROCESS

1 HEADER

THE ELF HEADER IS PARSED
THE PROGRAM HEADER IS PARSED
(SECTIONS ARE NOT USED)

2 MAPPING

THE FILE IS MAPPED IN MEMORY
ACCORDING TO ITS SEGMENT(S)



3 EXECUTION

ENTRY IS CALLED
SYSCALLS[KERNEL SERVICES] ARE ACCESSED VIA:
- SYSCALL NUMBER IN THE RAX REGISTER
- CALLING INSTRUCTION SYSCALL

TRIVIA

THE ELF WAS FIRST SPECIFIED BY U.S.L.[UNIX SYSTEM LABORATORIES] AND U.I.[UNIVERSAL INTERCONNECTION] FOR UNIX SYSTEM V, IN 1989

THE ELF IS USED, AMONG OTHERS, IN:
- LINUX, ANDROID, *BSD, SOLARIS, BEOS
- PSP, PLAYSTATION 2-4, DREAMCAST, GAMECUBE, WII
- VARIOUS OSES MADE BY SAMSUNG, ERICSSON, NOKIA,
- MICROCONTROLLERS FROM ATMEL, TEXAS INSTRUMENTS

VERSION 1.0C
2013/12/04
CC BY