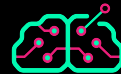
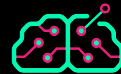


A Arte da Engenharia Reversa

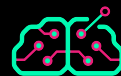


O binário ELF (Extra)



ELF

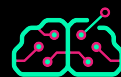
- Nos binários ELF, os cabeçalhos mudam, mas seguem uma lógica similar ao PE.
- Os trechos mapeados em memória são os **segmentos**. Também conhecidos como Program Headers, ou simplesmente PHDRS, eles contém as seções, que são opcionais no ELF.



Tipos de dados

32-Bit Data Types

Name	Size	Alignment	Purpose
Elf32_Addr	4	4	Unsigned program address
Elf32_Half	2	2	Unsigned medium integer
Elf32_Off	4	4	Unsigned file offset
Elf32_Sword	4	4	Signed large integer
Elf32_Word	4	4	Unsigned large integer
unsigned char	1	1	Unsigned small integer

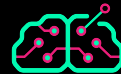


Cabeçalho ELF (32-bits)

Figure 1-3. ELF Header

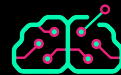
```
#define EI_NIDENT          16

typedef struct {
    unsigned char    e_ident[EI_NIDENT];
    Elf32_Half       e_type;
    Elf32_Half       e_machine;
    Elf32_Word       e_version;
    Elf32_Addr       e_entry;
    Elf32_Off        e_phoff;
    Elf32_Off        e_shoff;
    Elf32_Word       e_flags;
    Elf32_Half       e_ehsize;
    Elf32_Half       e_phentsize;
    Elf32_Half       e_phnum;
    Elf32_Half       e_shentsize;
    Elf32_Half       e_shnum;
    Elf32_Half       e_shstrndx;
} Elf32_Ehdr;
```



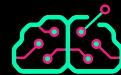
unsigned char **e_ident[16]**

- Bytes 0-4 - Do EI_MAG0 ao EI_MAG3, temos “\x7fELF”.
- Byte 5 - EI_CLASS pode ser 0 (nenhuma), 1 (32-bits) ou 2 (64-bits).
- Byte 6 - EI_DATA pode ser 0 (nenhuma), 1 (LSB) ou 2 (MSB).



Elf32_Halt **e_type**

Name	Value	Meaning
ET_NONE	0	No file type
ET_REL	1	Relocatable file
ET_EXEC	2	Executable file
ET_DYN	3	Shared object file
ET_CORE	4	Core file
ET_LOPROC	0xff00	Processor-specific
ET_HIPROC	0xffff	Processor-specific

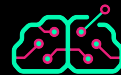


Elf32_Half **e_machine**

- Valores comuns:

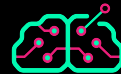
Macro	Valor	Arquitetura
EM_386	3	32-bit Intel (i386)
EM_MIPS	8	MIPS R3000 BE
EM_ARM	40 (0x28)	32-bit ARM
EM_X86_64	62 (0x3e)	x86-64 Intel (amd64)

- Lista completa em `/usr/include/linux/elf-em.h`



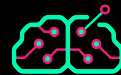
Outros campos importantes

- Elf32_Addr **e_entry**
 - VA do entriypoint.
- Elf32_Off **e_phoff**
 - Offset no arquivo da tabela de cabeçalhos de segmentos (Program Headers table).
- Elf32_Off **e_shoff**
 - Offset no arquivo da tabela de cabeçalhos de seções.
- Elf32_Half **e_phnum** e **e_shnum** (número de cabeçalhos de segmentos e seções, respectivamente).



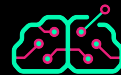
Tipos de Segmentos

- PT_PHDR
 - Contém os cabeçalhos
- PT_INTERP
 - Contém o interpretador requisitado.
- PT_LOAD
 - Será carregado em memória. Contém as seções correspondentes (se existirem).



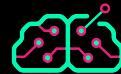
Seções

- `.text`
 - Contém código. Leitura e execução (em seu segmento).
- `.data`
 - Dados não inicializados. Leitura e escrita (em seu segmento).
- `.bss`
 - Dados não inicializados. Leitura e escrita (em seu segmento).
- `.rodata`
 - Dados inicializados. Somente leitura (em seu segmento).



Lab 09*

Inspeccionando o ELF



Lab 09 - Inspeccionando o ELF

1. Abra o arquivo *linux.elf* no 010 Editor e responda à seguintes perguntas:

- a) É um arquivo de que tipo (e_type)?
- b) Ele foi compilado para qual arquitetura?
- c) Quantos segmentos o programa possui?
- d) Quantas seções o programa possui?

Para conferir suas respostas, abra o arquivo no elfparser-ng e também no XELFViewer, mas não antes de responder as perguntas através do 010 Editor. ;)