Linux incident response - the readelf command

Taz Wake | Halkyn Consulting Ltd | Originally published 2023-11-06

What is readelf?

readelf is a command-line tool that is used to display information about ELF (Executable and Linkable Format) files. It is a part of the GNU Binutils package and is available on most Linux distributions. The readelf command can be used to extract information about the headers, sections, symbols, and other aspects of an ELF file. It is a powerful tool that can be used for general system administration purposes as well as for incident response and malware analysis.

One of the most common uses of the readelf command is to display the headers of an ELF file. The -h option can be used to display the ELF file header, which contains information about the file type, machine architecture, entry point address, and other details. For example, the command readelf -h /usr/bin/ls will display the ELF file header of the /usr/bin/ls binary.

Another useful feature of the readelf command is the ability to display information about the different sections of an ELF file. The -S option can be used to display the section headers of the file, which contain information about the various sections such as the code, data, and bss sections. For example, the command readelf -S /usr/bin/ls will display the section headers of the /usr/bin/ls binary.

The readelf command can also be used to display the symbols table of an ELF file. The -s option can be used to display the symbol table, which contains information about the functions and variables defined in the file. For example, the command readelf -s /usr/bin/ls will display the symbol table of the /usr/bin/ls binary.

In addition to these features, the readelf command can also be used to display information about the dynamic section of an ELF file (-d option), the relocation section (-r option), and the core notes (-n option). The readelf command can also be used to check whether a file is an ELF file (-a option) and to display the version of the readelf command (-v option).

The readelf command is a powerful tool that can be used for incident response and malware analysis. For example, it can be used to analyze the headers and sections of a binary file to determine whether it is malicious. The readelf command can also be used to extract information about the symbols and functions defined in a binary file, which can be useful for identifying malicious code.

Commonly used command line arguments

- To display the ELF file header of a binary file:

```
$ readelf -h /usr/bin/ls
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x6ab0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          136224 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```

example readelf -h output using the ls binary

- **Section Headers (**-S**):** This option displays detailed information about each section in the ELF file. Section headers include names, types, addresses, sizes, and other attributes.

```
$ readelf -S /usr/bin/ls
There are 31 section headers, starting at offset 0x21420:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS         0000000000000318  00000318
       000000000000001c  0000000000000000   A       0     0     1
  [ 2] .note.gnu.pr[...] NOTE             0000000000000338  00000338
       0000000000000030  0000000000000000   A       0     0     8
  [ 3] .note.gnu.bu[...] NOTE             0000000000000368  00000368
       0000000000000024  0000000000000000   A       0     0     4
  [ 4] .note.ABI-tag     NOTE             000000000000038c  0000038c
       0000000000000020  0000000000000000   A       0     0     4
  [ 5] .gnu.hash         GNU_HASH         00000000000003b0  000003b0
       000000000000004c  0000000000000000   A       6     0     8
  [ 6] .dynsym           DYNSYM           0000000000000400  00000400
       0000000000000b88  0000000000000018   A       7     1     8
  [ 7] .dynstr           STRTAB           0000000000000f88  00000f88
       00000000000005a6  0000000000000000   A       0     0     1
  [ 8] .gnu.version      VERSYM           000000000000152e  0000152e
       00000000000000f6  0000000000000002   A       6     0     2
  [ 9] .gnu.version_r    VERNEED          0000000000001628  00001628
       00000000000000c0  0000000000000000   A       7     2     8
  [10] .rela.dyn         RELA             00000000000016e8  000016e8
```

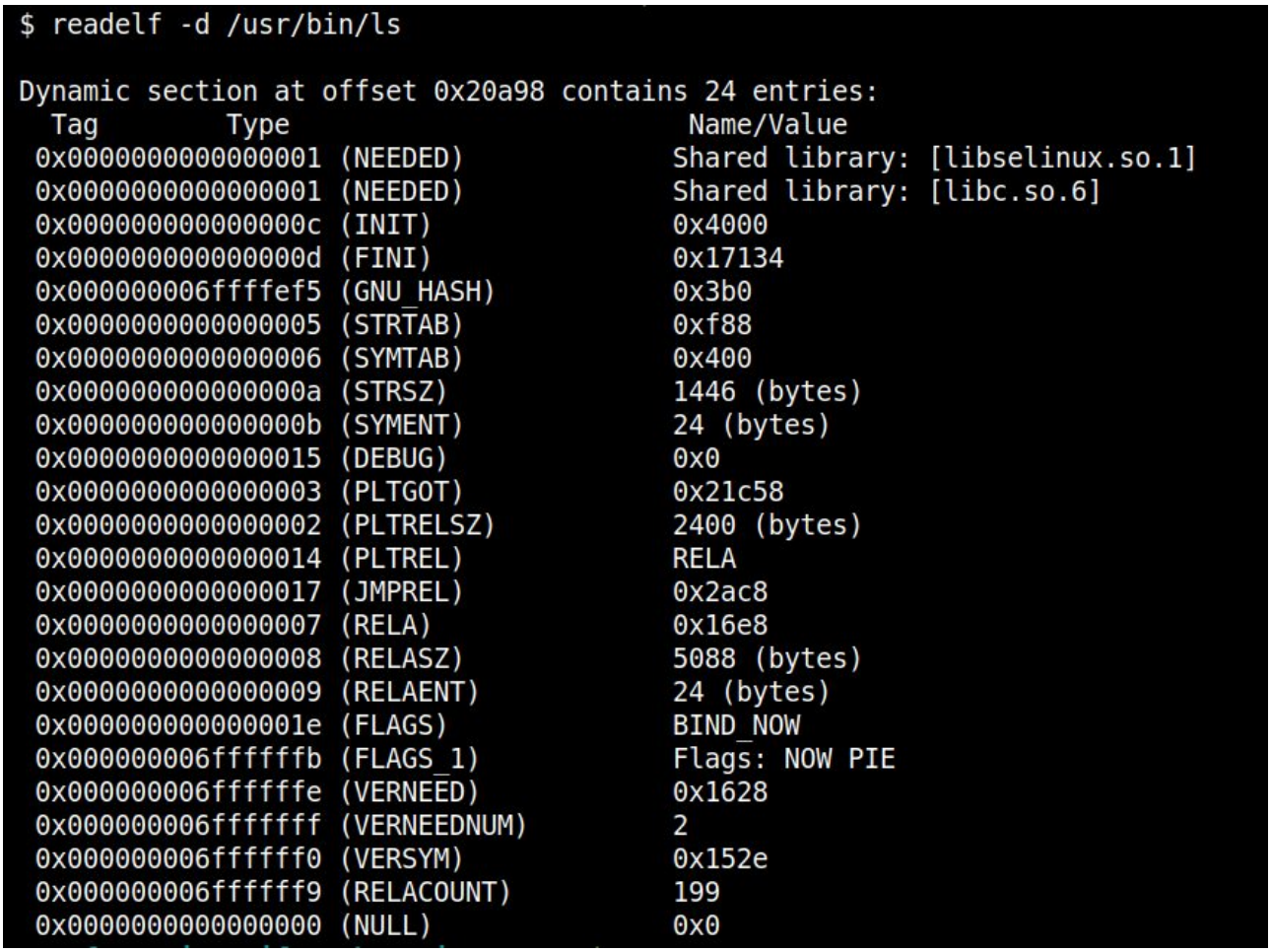
Truncated output from readelf -S looking at the /usr/bin/ls binary

- **Symbol Table (**-s**):** The symbol table contains information about symbols used in the binary, including functions and variables. This is invaluable for understanding program logic.

```
$ readelf -s /usr/bin/ls

Symbol table '.dynsym' contains 123 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __[...]@GLIBC_2.3 (2)
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     3: 0000000000000000     0 OBJECT  GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.3.4 (4)
     6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND raise@GLIBC_2.2.5 (3)
     7: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _[...]@GLIBC_2.34 (5)
     8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND abort@GLIBC_2.2.5 (3)
     9: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    10: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    11: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
    12: 0000000000000000     0 OBJECT  GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    13: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    14: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _exit@GLIBC_2.2.5 (3)
    15: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    16: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND   [...]@GLIBC_2.4 (6)
```

Truncated example output from readelf -s looking at /usr/bin/ls

- **Dynamic Section (**-d**):** This option reveals the dynamic linking information, including shared library dependencies and versioning details.



```
$ readelf -d /usr/bin/ls

Dynamic section at offset 0x20a98 contains 24 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libselinux.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000c (INIT)               0x4000
 0x000000000000000d (FINI)               0x17134
 0x000000006ffffef5 (GNU_HASH)           0x3b0
 0x0000000000000005 (STRTAB)             0xf88
 0x0000000000000006 (SYMTAB)             0x400
 0x000000000000000a (STRSZ)              1446 (bytes)
 0x000000000000000b (SYMENT)             24 (bytes)
 0x0000000000000015 (DEBUG)              0x0
 0x0000000000000003 (PLTGOT)             0x21c58
 0x0000000000000002 (PLTRELSZ)           2400 (bytes)
 0x0000000000000014 (PLTREL)             RELA
 0x0000000000000017 (JMPREL)             0x2ac8
 0x0000000000000007 (RELA)               0x16e8
 0x0000000000000008 (RELASZ)             5088 (bytes)
 0x0000000000000009 (RELAENT)            24 (bytes)
 0x000000000000001e (FLAGS)              BIND_NOW
 0x000000006ffffffb (FLAGS_1)            Flags: NOW PIE
 0x000000006ffffffe (VERNEED)            0x1628
 0x000000006fffffff (VERNEEDNUM)         2
 0x000000006ffffff0 (VERSYM)             0x152e
 0x000000006ffffff9 (RELACOUNT)          199
 0x0000000000000000 (NULL)               0x0
```

Output of readelf -d examining /usr/bin/ls

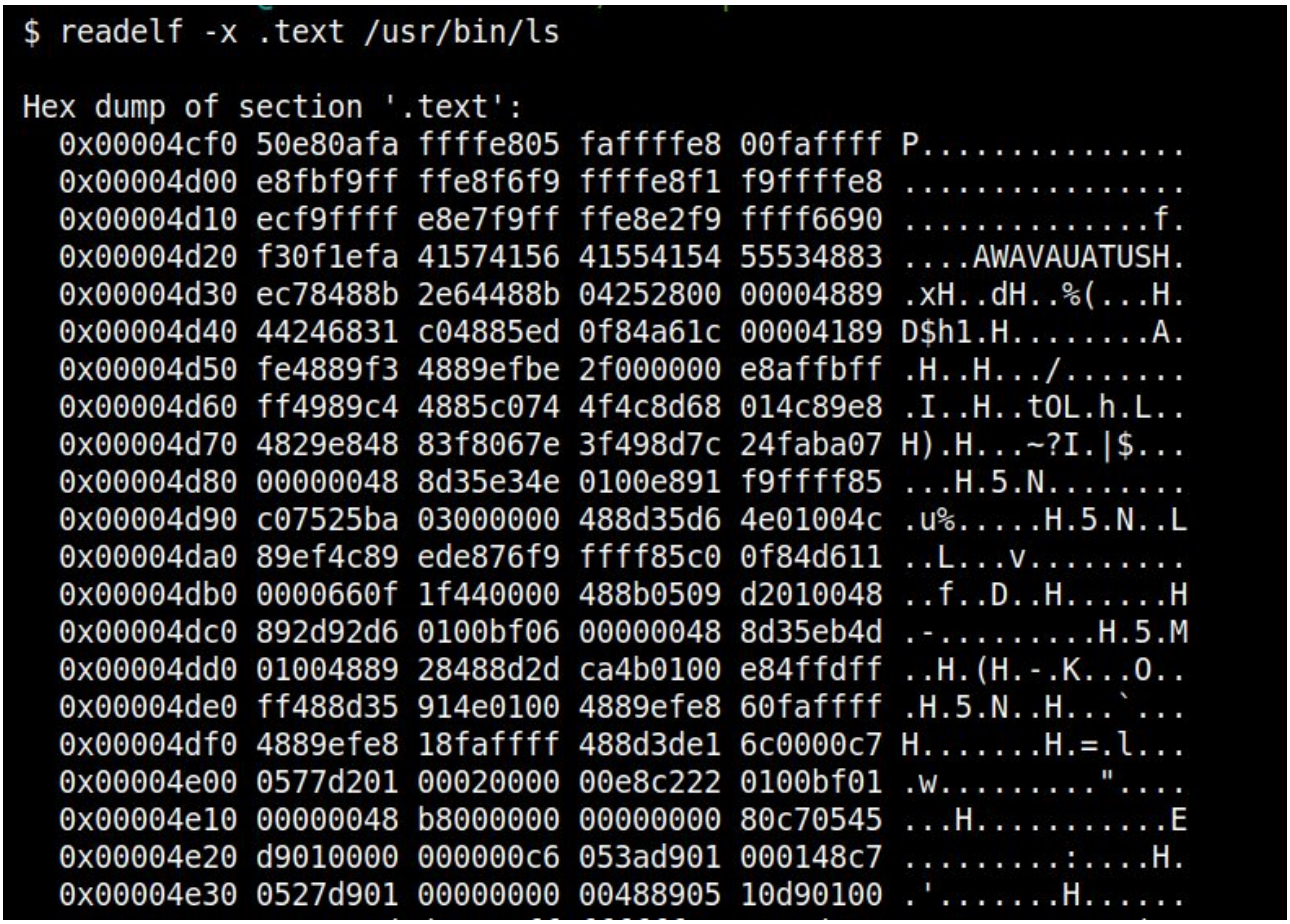
- **Program Headers (**-l**):** Program headers describe segments' layout in the binary. This information is vital for understanding memory mapping and the executable's load address.

```
$ readelf -l /usr/bin/ls

Elf file type is DYN (Position-Independent Executable file)
Entry point 0x6ab0
There are 13 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000000040 0x0000000000000040
                 0x00000000000002d8 0x00000000000002d8  R      0x8
  INTERP         0x0000000000000318 0x0000000000000318 0x0000000000000318
                 0x000000000000001c 0x000000000000001c  R      0x1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000003428 0x0000000000003428  R      0x1000
  LOAD           0x0000000000004000 0x0000000000004000 0x0000000000004000
                 0x0000000000013146 0x0000000000013146  R E    0x1000
  LOAD           0x0000000000018000 0x0000000000018000 0x0000000000018000
                 0x0000000000007458 0x0000000000007458  R      0x1000
  LOAD           0x0000000000020000 0x0000000000021000 0x0000000000021000
                 0x0000000000001278 0x0000000000002540  RW     0x1000
  DYNAMIC        0x0000000000020a98 0x0000000000021a98 0x0000000000021a98
                 0x00000000000001c0 0x00000000000001c0  RW     0x8
  NOTE           0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000030 0x0000000000000030  R      0x8
  NOTE           0x0000000000000368 0x0000000000000368 0x0000000000000368
                 0x0000000000000044 0x0000000000000044  R      0x4
```

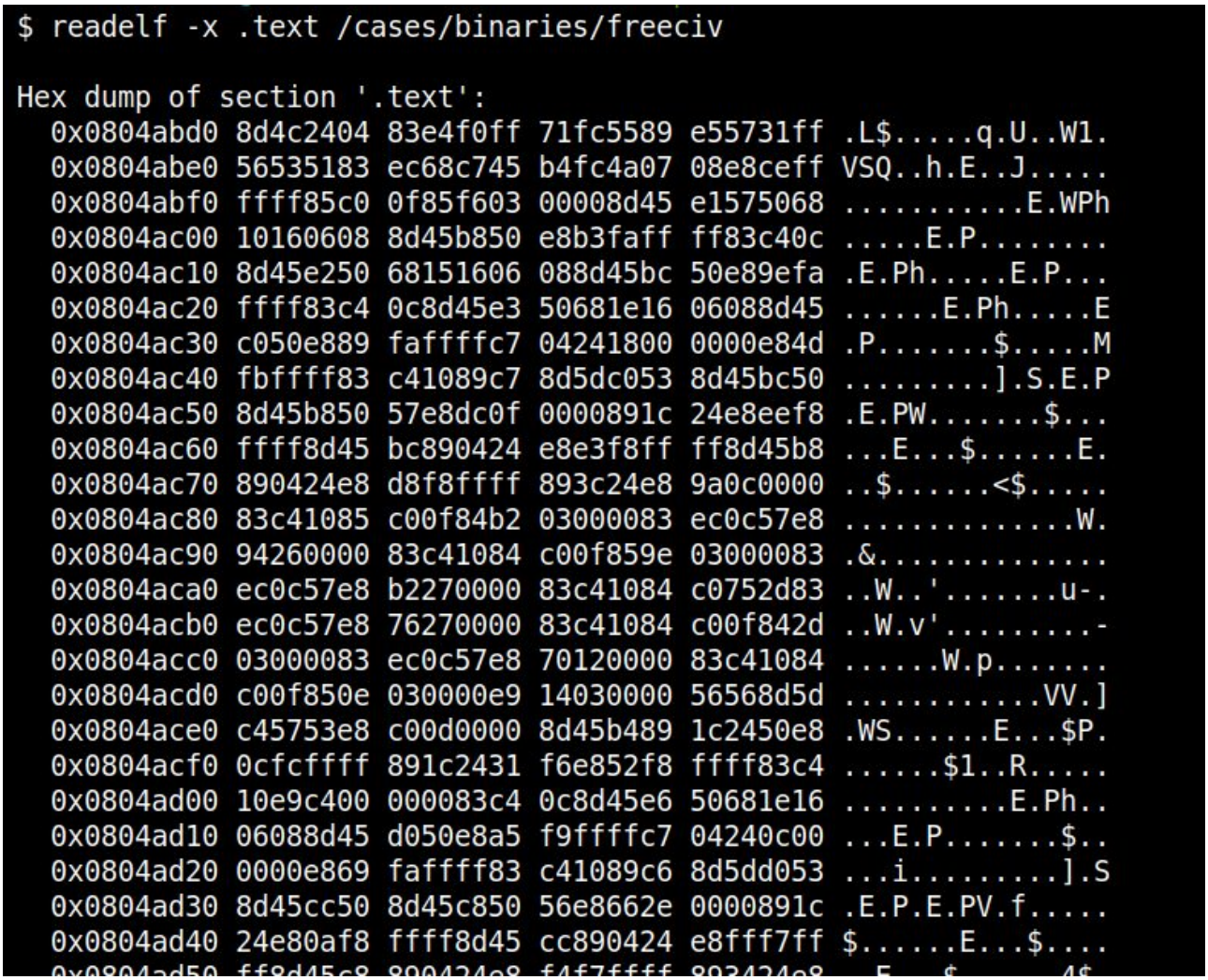
```
 GNU_PROPERTY   0x0000000000000338 0x0000000000000338 0x0000000000000338
                0x0000000000000030 0x0000000000000030  R      0x8
 GNU_EH_FRAME   0x000000000001cdcc 0x000000000001cdcc 0x000000000001cdcc
                0x000000000000056c 0x000000000000056c  R      0x4
 GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                0x0000000000000000 0x0000000000000000  RW     0x10
 GNU_RELRO      0x0000000000020000 0x0000000000021000 0x0000000000021000
                0x0000000000001000 0x0000000000001000  R      0x1

Section to Segment mapping:
 Segment Sections...
  00
  01     .interp
  02     .interp .note.gnu.property .note.gnu.build-id .note.ABI-tag .gnu.hash .dynsym .dynstr .gnu.version .gnu.version_r .re
  03     .init .plt .plt.got .plt.sec .text .fini
  04     .rodata .eh_frame_hdr .eh_frame
  05     .ctors .dtors .data.rel.ro .dynamic .got .data .bss
  06     .dynamic
  07     .note.gnu.property
  08     .note.gnu.build-id .note.ABI-tag
  09     .note.gnu.property
  10     .eh_frame_hdr
  11
  12     .ctors .dtors .data.rel.ro .dynamic .got
```

Two screenshots of output from readelf -l

- **Section Headers and Their Contents (**-x**):** This option displays the hexadecimal and ASCII representation of the content of a specific section. For instance, to examine the .text section:



Example showing the .text section of /usr/bin/ls



Example showing the .text section of a malware sample (fysbis)

- **Display Dynamic String Table (**--dyn-syms**):** This option prints dynamic symbols, which are essential for dynamic linking analysis.

```
$ readelf --dyn-syms /usr/bin/ls

Symbol table '.dynsym' contains 123 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __[...]@GLIBC_2.3 (2)
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     3: 0000000000000000     0 OBJECT  GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.3.4 (4)
     6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND raise@GLIBC_2.2.5 (3)
     7: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _[...]@GLIBC_2.34 (5)
     8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND abort@GLIBC_2.2.5 (3)
     9: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    10: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    11: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
    12: 0000000000000000     0 OBJECT  GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    13: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    14: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _exit@GLIBC_2.2.5 (3)
    15: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (3)
    16: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.4 (6)
```

Truncated output from readelf --dyn-syms run against the /usr/bin/ls binary

Understanding the output of the readelf command empowers professionals to dissect ELF binaries systematically, making it an indispensable tool for incident responders and security analysts dealing with Linux-based systems and potential security threats.

#linux #elf #binaryfiles #fileanalysis #forensics #incidentresponse #dfir #cybersecurity

---

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858