目录

可执行文件格式

- 最新版本： `v0.8.3`
- 更新时间： `20231007`

简介

介绍常用的可执行文件格式，先是概览；然后是通用内容；接着介绍常见格式，包括早期通用的COFF、Windows的PE、Linux/Android的ELF、macOS/iOS的Mach-O，以及相关的工具，比如查看格式的file，和解析格式的LIEF，且给出具体用法实例，包括解析Mach-O、解析Android的OAT、解析ELF；以及相关子教程。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/executable_file_format: 可执行文件格式

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- 可执行文件格式 book.crifan.org
- 可执行文件格式 crifan.github.io

离线下载阅读

- 可执行文件格式 PDF
- 可执行文件格式 ePub
- 可执行文件格式 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

可执行文件格式概览

可执行文件格式，常见的有很多种。

此处只至少其中相对常见的：

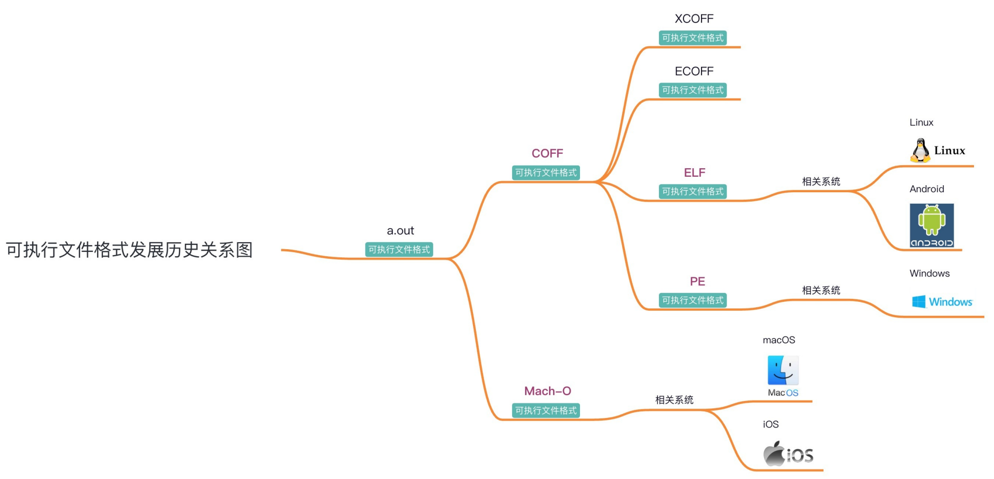
背景知识

文件后缀和格式

- 文件后缀和格式
  - 库文件
    - 动态库文件
      - Win: `.dll`
      - Linux/Android: `.so`
        - 相关：目标文件： `.o`
      - Mac: `.dylib`
    - 静态库文件
      - Win: `.lib`
      - Linux: `.a`
      - Mac: `.a`
  - 可执行文件
    - Win:.exe
    - Linux/Android: 无后缀
    - Mac:.app 或 无后缀

可执行文件格式和文件后缀

- 可执行文件格式和文件后缀
  - 可执行文件格式发展历史关系图
    - 离线查看



      -
    - 在线浏览

        - 可执行文件格式发展历史关系图| ProcessOn免费在线作图
    - 概述
        - 早期的： `a.out` 、通用的 `COFF`
        - 目前主流的：
            - `Win` ： `PE`
                - 相关文件： `.exe` 可执行文件、 `.dll` 动态库
            - `Linux` 和 `Android` ： `ELF`
                - 相关文件： `无后缀` 可执行文件、 `.so` 动态库、 `.o` 目标文件
            - `macOS` / `iOS` ： `Mach-O`
                - 相关文件： `无后缀` 可执行文件、 `.dylib` 动态库、 `.o` 目标文件
    - 详解
        - 早期的
            - Microsoft
                - `MZ` ： `DOS` 中的 `.exe`
                    - `MZ` = `Mark Zbikowski` ： MS-DOS主要开发者之一
            - UNIX系统
                - 最早的
                    - `a.out`
                        - = `assembler output`
                - 后来通用的
                    - `COFF`
        - 目前主流的
            - Microsoft系的： `PE`
                - 常见系统： `Windows`
                - 常见后缀： `.exe`
                    - 相关的：动态链接库的 `.dll`
            - Linux系的： `ELF`
                - 常见系统： `Linux` 各个发行版（ `Ubuntu` 、 `CentOS` 等）、（基于Linux改造的） `Android`
                - 常见后缀： `无`
                    - 相关的：动态链接库的 `.so` 、目标文件的 `.o`
            - Apple系的： `Mach-O`
                - 常见系统： `macOS` / `iOS` / `watchOS` / `tvOS`
                - 常见后缀： `无`
                    - 相关的：动态链接库的 `.dylib`

通用内容

核心点和通用点

不同可执行文件格式的内部，最核心的是：

- 编译和链接的过程
    - 涉及到编译器和链接器
        - 如何生成代码？
        - 把生成的代码和数据，放到什么地方？
        - 运行时如何加载这些代码和数据？
            - 才能确保程序能正常执行

不用文件格式的通用内容

- 各种的
    - Segment段
        - 代码段
            - `.text`
        - 数据段
            - `.data`
        - 等等
    - section区
        - `.bss`
        - `.data`
        - `.rodata`
        - 等等

函数调用逻辑

When a program wants to call a function, it actually does following flow:

- 1.It made a jump to relevant entry in PLT (Procedure Linkage Table)
- 2.In PLT, there is another jump to an address mentioned in related entry in GOT (Global Offset Table)
- 3.If this is the first the function is called, follow step #4. If this isn't, follow step #5.
- 4.The related GOT entry contains an address that points back to next instruction in PLT. Program will jump to this address and then calls the dynamic linker to resolve the function's address. If the function is found, its address is put in related GOT entry and then the function itself is executed.
    - So, another time the function is called, GOT already holds its address and PLT can jump directly to the address. This procedure is called lazy binding; all external symbols are not resolved until the time it is really needed (in this case, when a function is called). Jump to step #6
- 5.Jump to the address mentioned in GOT. It is the address of the function thus PLT is no longer used
- 6.Execution of the function is finished. Jump back to the next instruction in the main program.

-》

- 1.跳转到PLT=Procedure Linkage Table中的对应入口
- 2.PLT中还有另外一个跳转，跳转到GOT=Global Offset Table中相关的入口
- 3.如果函数找到了被调用了，则继续步骤4，否则继续步骤5
-
    1. GOT入口中包括了一个地址，指向了PLT中的下一个指令
    2. 程序会调到转该指令，谈话调用动态链接器去继续函数地址。
        - 如果函数找到了，则该函数地址被放到GOT入口中，然后函数自己被执行
    3. 如果下一次函数被执行时，GOT中已有其地址，所以PLT可以直接跳转到对应地址
        - 此过程叫做懒绑定lazy binding
            - 所有外部符号都不会去解析，直到去真正要执行之前，需要解析，才去解析
- 5.跳转到GOT中提到的函数的地址。所以就不用PLT了。
- 执行函数，直到结束。跳转会主程序中的下一个指令。

常见格式

此处介绍常见的可执行文件格式：

- 早期通用的： `COFF`
- 目前主流的：
  - `Win` 的： `PE`
  - `Linux` / `Android` 的： `ELF`
  - `macOS` / `iOS` 的： `Mach-O`

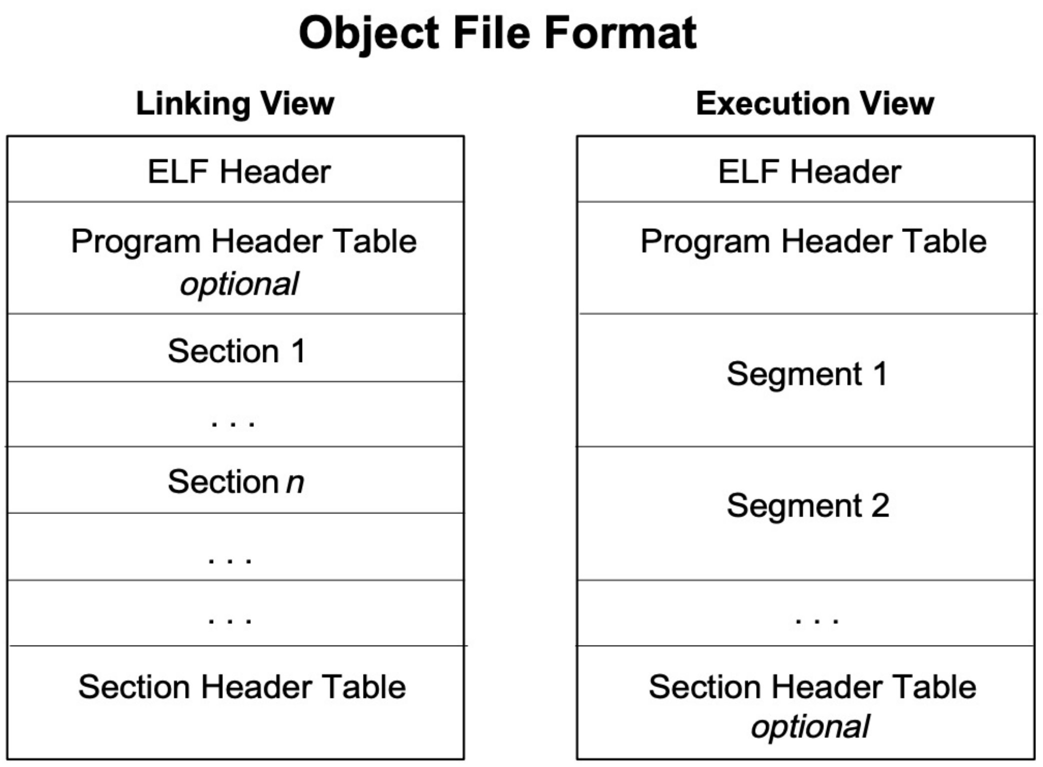
COFF

- `COFF` = `Common Object File Format` = `通用对象文件格式`

PE

- `PE` = `Portable Executable`

ELF

- `ELF = Executable and Linking Format = 可执行和链接格式`
  - 按类型分
    - `Relocatable File = 可重定位文件`
      - A relocatable file holds code and data suitable for linking with other object files to create an executable or a shared object file.
    - `Executable File = 可执行文件`
      - An executable file holds a program suitable for execution
    - `Shared Object File = 共享对象文件`
      - A shared object file holds code and data suitable for linking in two contexts. First, the link editor may process it with other relocatable and shared object files to create another object file.
      - Second, the dynamic linker combines it with an executable file and other shared objects to create a process image
  - ELF格式概览
    - 图
      - 

  - 详见子教程
    - 可执行文件格式：ELF

Mach-O

- `Mach-O` = `Mach Object`
    - `Mach-O` 是 `Mac` 和 `iOS` 等Apple平台中的可执行文件格式
    - 详见子教程
        - 可执行文件格式：Mach-O

相关工具

file

- `file`
    - 是个命令行工具，可以在 `Linux` / `Android` 和 `macOS` / `iOS` 中查看文件的格式

举例

Mac

- rsync

```
➜  bin pwd
/usr/local/Cellar/rsync/3.2.7_1/bin

➜  bin file rsync
rsync: Mach-O 64-bit executable x86_64
```

- readelf

```
➜  ~ file /opt/homebrew/opt/binutils/bin/readelf
/opt/homebrew/opt/binutils/bin/readelf: Mach-O 64-bit executable arm64
```

- python64.dylib

```
➜  plugins pwd
/Applications/IDA Pro 7.0/ida.app/Contents/MacOS/plugins
➜  plugins ll
total 33776
...
-rw-r--r--@  1 crifan  admin   113K  9 14  2017 python.dylib
-rw-r--r--@  1 crifan  admin   113K  9 14  2017 python64.dylib
...

➜  plugins file python64.dylib
python64.dylib: Mach-O 64-bit dynamically linked shared library x86_64
```

FAT格式

- FAT格式的rsync

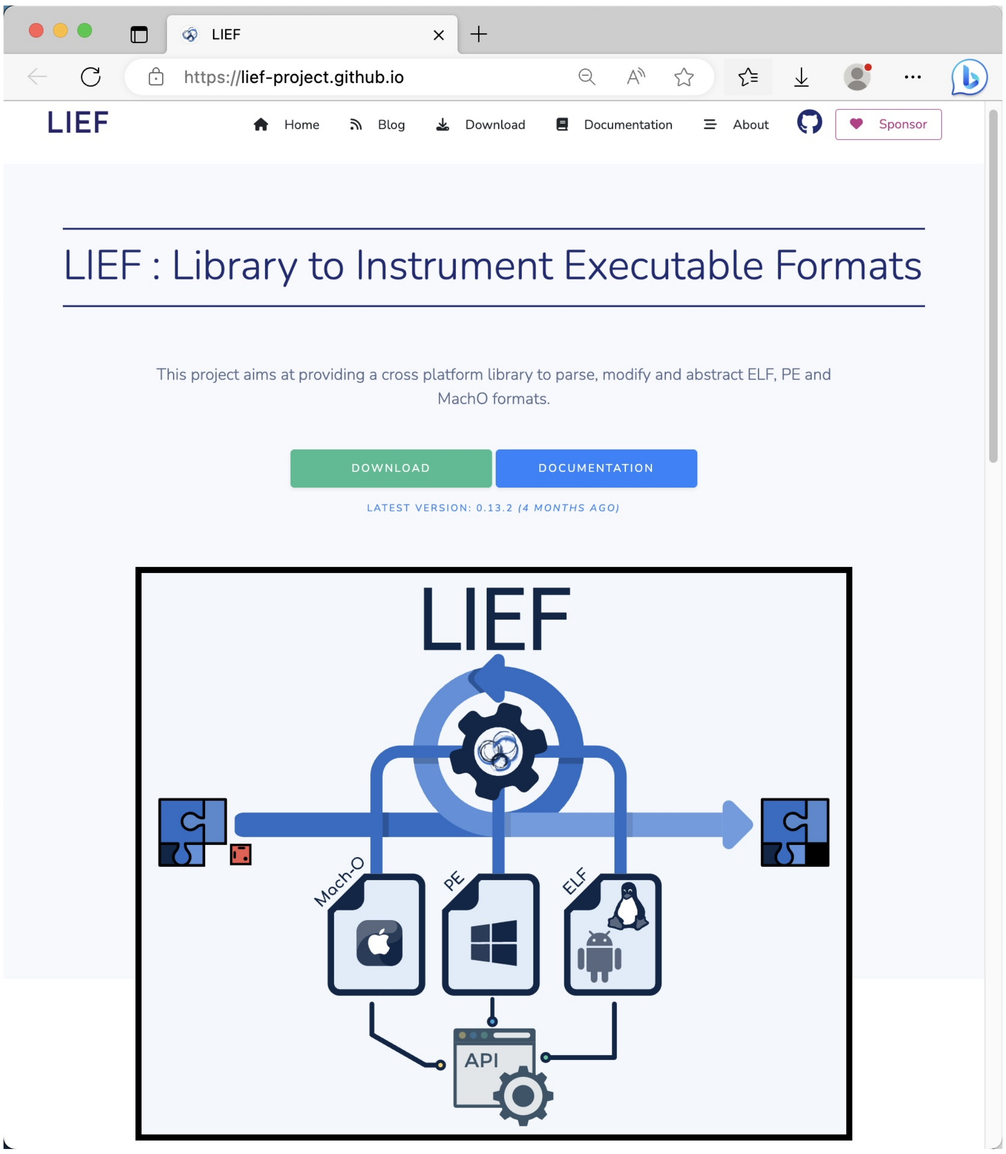
```
➜  ~ which rsync
/usr/bin/rsync
➜  ~ file /usr/bin/rsync
/usr/bin/rsync: Mach-O universal binary with 2 architectures: [x86_64:Mach-O 64-bit
 executable x86_64] [arm64e:Mach-O 64-bit executable arm64e]
/usr/bin/rsync (for architecture x86_64):	Mach-O 64-bit executable x86_64
/usr/bin/rsync (for architecture arm64e):	Mach-O 64-bit executable arm64e
```

更多例子详见：

FAT举例 · 可执行文件格式：Mach-O (crifan.org)

LIEF

- `LIEF`
  - 介绍：用于查看和解析（`ELF` / `MachO` / `PE` / `Android` 等）各种通用的可执行文件格式的库
  - 一句话描述：Library to Instrument Executable Formats
  - 支持格式
    - `ELF`
    - `PE`
    - `MachO`
    - `Android`
      - `DEX`
      - `OAT`
      - `ART`
      - `VDEX`
  - 主页
    - LIEF
      - 图



LIEF : Library to Instrument Executable Formats

This project aims at providing a cross platform library to parse, modify and abstract ELF, PE and MachO formats.

- 文档
  - Welcome to LIEF's documentation! — LIEF Documentation
- 下载
  - LIEF

安装

- Mac

  ```
  pip install lief
  ```

LIEF用法举例

LIEF解析Mach-O

从 `Mach-O` 中解析 `__text` 、 `__data` 等section的内容：

```
import lief

# --- Extract __text and __data section content from the binary ---
binary = lief.parse("uncrackable.arm64")
text_section = binary.get_section("__text")
text_content = text_section.content

data_section = binary.get_section("__data")
data_content = data_section.content
```

Android

此处举例说明，如何用LIEF去解析Android的相关格式。

LIEF解析Android的OAT

此处用例子说明，如何用LIEF解析Android的OAT格式文件：

代码：

```
import lief

oat = lief.parse("SomeOAT")
for s in oat.dynamic_symbols:
  print(s)
```

输出：

```
oatdata                         OBJECT    GLOBAL    1000       1262000
oatexec                         OBJECT    GLOBAL    1263000    10d4060
oatlastword                     OBJECT    GLOBAL    233705c    4
oatbss                          OBJECT    GLOBAL    2338000    f5050
oatbsslastword                  OBJECT    GLOBAL    242d04c    4
```

LIEF解析ELF

LIEF修改ELF的libtacker.so

此处举例说明LIEF的用法：

- 用LIEF解析修改ELF格式的so库文件：`libtacker.so`

基本用法：打印解析后对象的信息

代码：

```
import lief

libtackerObj = lief.parse('modifyElf/input/libtacker_arm64_v8a.so')
print("libtackerObj=", libtackerObj)
```

输出：

```
➜  libtacker_so python modifyElf/modifyLibtackerSo.py
intputElfFile= modifyElf/input/libtacker_armeabi_v7a.so
inputFileName= libtacker_armeabi_v7a.so
pureFileName=%s, fileExt=%s ('libtacker_armeabi_v7a', '.so')
Loaded file:  modifyElf/input/libtacker_armeabi_v7a.so
libtackerElf= Header
======
Magic:                           7f 45 4c 46
Class:                           CLASS32
Endianness:                      LSB
Version:                         CURRENT
OS/ABI:                          SYSTEMV
ABI Version:                     0
Machine type:                    ARM
File type:                       DYNAMIC
Object file version:             CURRENT
Entry Point:                     0xb9c8
Program header offset:           0x34
Section header offset:           815008
Processor Flag:                  83886592 SOFT_FLOAT EABI_VER5
Header size:                     52
Size of program header:          32
Number of program header:        9
Size of section header:          40
Number of section headers:       27
Section Name Table idx:          26

Sections
========
                    NULL           0         0         0         0
.note.android.ident NOTE           154       98        154       1.09412    ALLOC
                  LOAD NOTE
.note.gnu.build-id  NOTE           1ec       24        1ec       4.13606    ALLOC
                  LOAD NOTE
```

```
.dynsym            DYNSYM       210      6d0      210      3.54702  ALLOC
                  LOAD
.gnu.version       HIOS         8e0      da       8e0      1.48975  ALLOC
                  LOAD
.gnu.version_r     GNU_VERNEED  9bc      40       9bc      2.16157  ALLOC
                  LOAD
.gnu.hash          GNU_HASH     9fc      1ec      9fc      6.3602   ALLOC
                  LOAD
.hash              HASH         be8      370      be8      1.38896  ALLOC
                  LOAD
.dynstr            STRTAB       f58      b8f      f58      4.66594  ALLOC
                  LOAD
.rel.dyn           REL          1ae8     2dd0     1ae8     3.36673  ALLOC
                  LOAD
.ARM.exidx         ARM_EXIDX    48b8     1ad8     48b8     5.18866  ALLOC LINK_O
RDER              LOAD ARM_UNWIND
.rel.plt           REL          6390     120      6390     3.54485  ALLOC INFO_L
INK               LOAD
.ARM.extab         PROGBITS     64b0     334c     64b0     4.80098  ALLOC
                  LOAD
.rodata            PROGBITS     97fc     21c7     97fc     5.26982  ALLOC MERGE
STRINGS           LOAD
.text              PROGBITS     b9c8     b72c4    b9c8     6.55022  ALLOC EXECIN
STR               LOAD
.plt               PROGBITS     c2c90    260      c2c90    3.78829  ALLOC EXECIN
STR               LOAD
.data.rel.ro       PROGBITS     c3ef0    1790     c2ef0    4.42226  WRITE ALLOC
                  LOAD GNU_RELRO
.fini_array        FINI_ARRAY   c5680    8        c4680    1.75     WRITE ALLOC
                  LOAD GNU_RELRO
.init_array        INIT_ARRAY   c5688    114      c4688    -0       WRITE ALLOC
                  LOAD GNU_RELRO
.dynamic           DYNAMIC      c579c    e8       c479c    3.17459  WRITE ALLOC
                  LOAD DYNAMIC GNU_RELRO
.got               PROGBITS     c5884    70       c4884    3.63662  WRITE ALLOC
                  LOAD GNU_RELRO
.got.plt           PROGBITS     c58f4    9c       c48f4    1.98777  WRITE ALLOC
                  LOAD GNU_RELRO
.data              PROGBITS     c6990    2410     c4990    7.22773  WRITE ALLOC
                  LOAD
.bss               NOBITS       c8da0    22d      c6da0    5.11609  WRITE ALLOC
                  LOAD
.comment           PROGBITS     0        c6       c6da0    5.05236  MERGE STRING
S
.ARM.attributes    ARM_ATTRIBUTES 0      3c       c6e66    4.587
.shstrtab          STRTAB       0        fe       c6ea2    4.2632

Segments
========
PHDR              r--      34       34       34       120      120      4

LOAD              r-x      0        0        0        c2ef0    c2ef0    1000
Sections in this segment :
    .note.android.ident
    .note.gnu.build-id
    .dynsym
    .gnu.version
    .gnu.version_r
    .gnu.hash
    .hash
    .dynstr
    .rel.dyn
    .ARM.exidx
    .rel.plt
    .ARM.extab
    .rodata
    .text
    .plt

LOAD                rw-       c2ef0      c3ef0      c3ef0      1aa0       1aa0       1000
Sections in this segment :
    .data.rel.ro
    .fini_array
    .init_array
    .dynamic
    .got
    .got.plt

LOAD                rw-       c4990      c6990      c6990      2410       263d       1000
Sections in this segment :
    .data
    .bss

DYNAMIC             rw-       c479c      c579c      c579c      e8         e8         4
Sections in this segment :
    .dynamic

GNU_RELRO           r--       c2ef0      c3ef0      c3ef0      1aa0       2110       1
Sections in this segment :
    .data.rel.ro
    .fini_array
    .init_array
    .dynamic
    .got
    .got.plt

GNU_STACK           rw-       0          0          0          0          0          0

NOTE                r--       154        154        154        bc         bc         4
Sections in this segment :
    .note.android.ident
    .note.gnu.build-id

ARM_UNWIND          r--       48b8       48b8       48b8       1ad8       1ad8       4
Sections in this segment :
    .ARM.exidx


Dynamic entries
===============
NEEDED                b71       liblog.so
NEEDED                b7b       libm.so
NEEDED              b5b       libdl.so
NEEDED              b69       libc.so
SONAME              b83       libforce.so
FLAGS               8          BIND_NOW
FLAGS_1             1          NOW
REL                 1ae8
RELSZ               2dd0
RELENT              8
RELCOUNT            573
JMPREL              6390
PLTRELSZ            120
PLTGOT              c58f4
PLTREL              11
SYMTAB              210
SYMENT              10
STRTAB              f58
STRSZ               b8f
GNU_HASH            9fc
HASH                be8
INIT_ARRAY          c5688     [0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0
x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x
0, 0x0, 0x0, 0x0, 0x0, 0x0]
INIT_ARRAYSZ        114
FINI_ARRAY          c5680     [0xb9dc, 0xb9c8]
FINI_ARRAYSZ        8
VERSYM              8e0
VERNEEDED           9bc
VERNEEDNUM          2
NULL                0

Dynamic symbols
===============
                              NOTYPE    LOCAL     0         0         * Local *
__cxa_finalize                FUNC      GLOBAL    0         0         LIBC      (2)
__cxa_atexit                  FUNC      GLOBAL    0         0         LIBC      (2)
__android_log_print           FUNC      GLOBAL    0         0         * Global *
__stack_chk_fail              FUNC      GLOBAL    0         0         LIBC      (2)
__stack_chk_guard             OBJECT    GLOBAL    0         0         LIBC      (2)
__strncpy_chk2                FUNC      GLOBAL    0         0         LIBC      (2)
strncpy                       FUNC      GLOBAL    0         0         LIBC      (2)
strncat                       FUNC      GLOBAL    0         0         LIBC      (2)
pthread_self                  FUNC      GLOBAL    0         0         LIBC      (2)
malloc                        FUNC      GLOBAL    0         0         LIBC      (2)
free                          FUNC      GLOBAL    0         0         LIBC      (2)
posix_memalign                FUNC      GLOBAL    0         0         LIBC      (2)
abort                         FUNC      GLOBAL    0         0         LIBC      (2)
vfprintf                      FUNC      GLOBAL    0         0         LIBC      (2)
fputc                         FUNC      GLOBAL    0         0         LIBC      (2)
vasprintf                     FUNC      GLOBAL    0         0         LIBC      (2)
__assert2                     FUNC      GLOBAL    0         0         LIBC      (2)
__sF                          OBJECT    GLOBAL    0         0         LIBC      (2)
strlen                        FUNC      GLOBAL    0         0         LIBC      (2)
realloc                       FUNC      GLOBAL    0         0         LIBC      (2)
__strlen_chk                  FUNC      GLOBAL    0         0         LIBC      (2)
```

```
memchr                        FUNC      GLOBAL    0         0         LIBC      (2)
__vsnprintf_chk               FUNC      GLOBAL    0         0         LIBC      (2)
pthread_mutex_lock            FUNC      GLOBAL    0         0         LIBC      (2)
calloc                        FUNC      GLOBAL    0         0         LIBC      (2)
pthread_mutex_unlock          FUNC      GLOBAL    0         0         LIBC      (2)
strcmp                        FUNC      GLOBAL    0         0         LIBC      (2)
pthread_getspecific           FUNC      GLOBAL    0         0         LIBC      (2)
memset                        FUNC      GLOBAL    0         0         LIBC      (2)
pthread_once                  FUNC      GLOBAL    0         0         LIBC      (2)
pthread_setspecific           FUNC      GLOBAL    0         0         LIBC      (2)
memcpy                        FUNC      GLOBAL    0         0         LIBC      (2)
pthread_key_delete            FUNC      GLOBAL    0         0         LIBC      (2)
pthread_key_create            FUNC      GLOBAL    0         0         LIBC      (2)
memmove                       FUNC      GLOBAL    0         0         LIBC      (2)
fprintf                       FUNC      GLOBAL    0         0         LIBC      (2)
fflush                        FUNC      GLOBAL    0         0         LIBC      (2)
dl_unwind_find_exidx          FUNC      GLOBAL    0         0         LIBC      (3)
.datadiv_decode8345671930506918460FUNC      GLOBAL    28ec9     2384      * Global *
.datadiv_decode17898716361002246085FUNC      GLOBAL    69a59     eee       * Global *
...
.datadiv_decode18430821605022316503FUNC      GLOBAL    9c47d     1044      * Global *
JNI_OnLoad                    FUNC      GLOBAL    b2bcd     48c       * Global *
.datadiv_decode9810505568447265400FUNC      GLOBAL    18c81     33e8      * Global *
...
.datadiv_decode3623259086264565478FUNC      GLOBAL    89201     8e6       * Global *

Static symbols
==============

Symbol versions
===============
* Local *
LIBC(2)
LIBC(2)
* Global *
LIBC(2)
LIBC(2)
...
LIBC(2)
LIBC(3)
* Global *
...
* Global *

Symbol versions definition
==========================

Symbol version requirement
==========================
1 libdl.so
1 libc.so

Dynamic relocations
===================
c3ef0     RELATIVE  32  0         0         DYNAMIC
c3ef4     RELATIVE  32  0         0         DYNAMIC
```

```
c3ef8     RELATIVE  32  0          0         DYNAMIC
...
c8d9c     RELATIVE  32  0          0         DYNAMIC
c5884     GLOB_DAT  32  0          5         DYNAMIC   __stack_chk_guard
c58a8     GLOB_DAT  32  0          12        DYNAMIC   __sF
c56ac     ABS32     32  0          27        DYNAMIC   .datadiv_decode834567193050691846
0
c56fc     ABS32     32  0          28        DYNAMIC   .datadiv_decode178987163610022460
85
c573c     ABS32     32  0          29        DYNAMIC   .datadiv_decode185723924818675064
6
...
c5738     ABS32     32  0          6c        DYNAMIC   .datadiv_decode362325908626456547
8

.plt.got relocations
====================
c5900     JUMP_SLOT 32  0          1         PLTGOT    __cxa_finalize
c5904     JUMP_SLOT 32  0          2         PLTGOT    __cxa_atexit
c5908     JUMP_SLOT 32  0          3         PLTGOT    __android_log_print
c590c     JUMP_SLOT 32  0          4         PLTGOT    __stack_chk_fail
c5910     JUMP_SLOT 32  0          6         PLTGOT    __strncpy_chk2
c5914     JUMP_SLOT 32  0          7         PLTGOT    strncpy
c5918     JUMP_SLOT 32  0          8         PLTGOT    strncat
c591c     JUMP_SLOT 32  0          9         PLTGOT    pthread_self
c5920     JUMP_SLOT 32  0          a         PLTGOT    malloc
c5924     JUMP_SLOT 32  0          b         PLTGOT    free
c5928     JUMP_SLOT 32  0          c         PLTGOT    posix_memalign
c592c     JUMP_SLOT 32  0          d         PLTGOT    abort
c5930     JUMP_SLOT 32  0          e         PLTGOT    vfprintf
c5934     JUMP_SLOT 32  0          f         PLTGOT    fputc
c5938     JUMP_SLOT 32  0          10        PLTGOT    vasprintf
c593c     JUMP_SLOT 32  0          11        PLTGOT    __assert2
c5940     JUMP_SLOT 32  0          13        PLTGOT    strlen
c5944     JUMP_SLOT 32  0          14        PLTGOT    realloc
c5948     JUMP_SLOT 32  0          15        PLTGOT    __strlen_chk
c594c     JUMP_SLOT 32  0          16        PLTGOT    memchr
c5950     JUMP_SLOT 32  0          17        PLTGOT    __vsnprintf_chk
c5954     JUMP_SLOT 32  0          18        PLTGOT    pthread_mutex_lock
c5958     JUMP_SLOT 32  0          19        PLTGOT    calloc
c595c     JUMP_SLOT 32  0          1a        PLTGOT    pthread_mutex_unlock
c5960     JUMP_SLOT 32  0          1b        PLTGOT    strcmp
c5964     JUMP_SLOT 32  0          1c        PLTGOT    pthread_getspecific
c5968     JUMP_SLOT 32  0          1d        PLTGOT    memset
c596c     JUMP_SLOT 32  0          1e        PLTGOT    pthread_once
c5970     JUMP_SLOT 32  0          1f        PLTGOT    pthread_setspecific
c5974     JUMP_SLOT 32  0          20        PLTGOT    memcpy
c5978     JUMP_SLOT 32  0          21        PLTGOT    pthread_key_delete
c597c     JUMP_SLOT 32  0          22        PLTGOT    pthread_key_create
c5980     JUMP_SLOT 32  0          23        PLTGOT    memmove
c5984     JUMP_SLOT 32  0          24        PLTGOT    fprintf
c5988     JUMP_SLOT 32  0          25        PLTGOT    fflush
c598c     JUMP_SLOT 32  0          26        PLTGOT    dl_unwind_find_exidx

Notes
=====
Note #0
-------
Name:                             Android
Type:                             ABI_TAG
Description:                      [15 00 00 00 72 32 34 00 00 00 00 00 00 00 00 00 ...]

Note #1
-------
Name:                             GNU
Type:                             BUILD_ID
Description:                      [fa 8a 30 55 6b db 44 9a 4a 2f 1f 83 46 6d 93 44 ...]
ID Hash:                          fa8a30556bdb449a4a2f1f83466d9344fdef3fb5



GNU Hash Table
==============
0x6000022d2e80

SYSV Hash Table
===============
0x600001381b40
```

复杂用法：修改 `.comment` 内容为随机乱码

代码：

- `modifyElf/modifyLibtackerSo.py`

```
# Function: mofidy libtacker so (arm64_v8a, armeabi_v7a) comments section content the output to new one
# Author: Crifan Li
# Update: 20230912

import os
import random
import string
import lief

################################################################################
# input & config
################################################################################

# intputElfFile = "modifyElf/input/libtacker_arm64_v8a.so"
intputElfFile = "modifyElf/input/libtacker_armeabi_v7a.so"
print("intputElfFile=", intputElfFile)

outputFoler = "modifyElf/output"

TotalRandomTime = 40
OutputSuffixStartNum = 20

################################################################################
# generate output file name
```

```
################################################################################

inputFileName = os.path.basename(intputElfFile)
print("inputFileName=", inputFileName)
pureFileName, fileExt = os.path.splitext(inputFileName)
print("pureFileName=%s, fileExt=%s", (pureFileName, fileExt)) # pureFileName=%s, fileEx
t=%s ('libtacker_arm64_v8a', '.so')

################################################################################
# Main
################################################################################

libtackerElf = lief.parse(intputElfFile)
print("Loaded file: ", intputElfFile)
# print("libtackerElf=", libtackerElf)
print("header=", libtackerElf.header)
print("imported_functions=", libtackerElf.imported_functions)
print("exported_functions=", libtackerElf.exported_functions)

for section in libtackerElf.sections:
  print("-"*20) # section's name
  print("name=", section.name) # section's name
  print("size=", section.size) # section's size
  contentLen = len(section.content)
  print("contentLen=", contentLen) # Should match the previous print

commentSection = libtackerElf.get_section(".comment")

print("\n\n")


print("+++ Start total %d round overwrite .comment section content with random char" %
TotalRandomTime)

ContentCharNum = commentSection.size
print("ContentCharNum=", ContentCharNum)

RamdonCharRange = string.ascii_letters + string.digits
print("RamdonCharRange=", RamdonCharRange)

for curRandomTimeIdx in range(TotalRandomTime):
  curRandomTime = curRandomTimeIdx + 1
  print("%s Random Time [%d] %s" % ("-"*30, curRandomTime, "-"*30))

  # randomCharList = random.sample(RamdonCharRange, 198)

  randomCharList = []
  for randomCharIdx in range(ContentCharNum):
    curRandomChar = random.choice(RamdonCharRange)
    randomCharList.append(curRandomChar)
  print("randomCharList=", randomCharList)

  # randomCharBytes = bytes(randomCharList)
  randomStr = "".join(randomCharList)
  print("randomStr=", randomStr)
  randomCharBytes = bytes(randomStr, "utf-8")
  print("randomCharBytes=", randomCharBytes)
  newContentBytes = randomCharBytes

  print("before: commentSection.content=", commentSection.content)

  # num0 = 0x30
  # print("commentSection contentType=", type(commentSection.content))
  # newContentBytes = bytes([num0] * commentSection.size)

  newContentMemoryView = memoryview(newContentBytes)
  # print("newContentMemoryView type=", type(newContentMemoryView))
  commentSection.content = newContentMemoryView
  print("after: commentSection.content=", commentSection.content)

  # outputElfFile = "modifyElf/output/libtacker_arm64_v8a_changedComment.so"
  # outputElfFile = "modifyElf/output/libtacker_arm64_v8a_changedComment_%d.so" % curRa
ndomTime
  suffixNum = OutputSuffixStartNum + curRandomTime
  outputFileName = "%s_changedComment_%d%s" % (pureFileName, suffixNum, fileExt)
  outputElfFile = os.path.join(outputFoler, outputFileName)

  libtackerElf.write(outputElfFile)
  print("Saved to", outputElfFile)
```

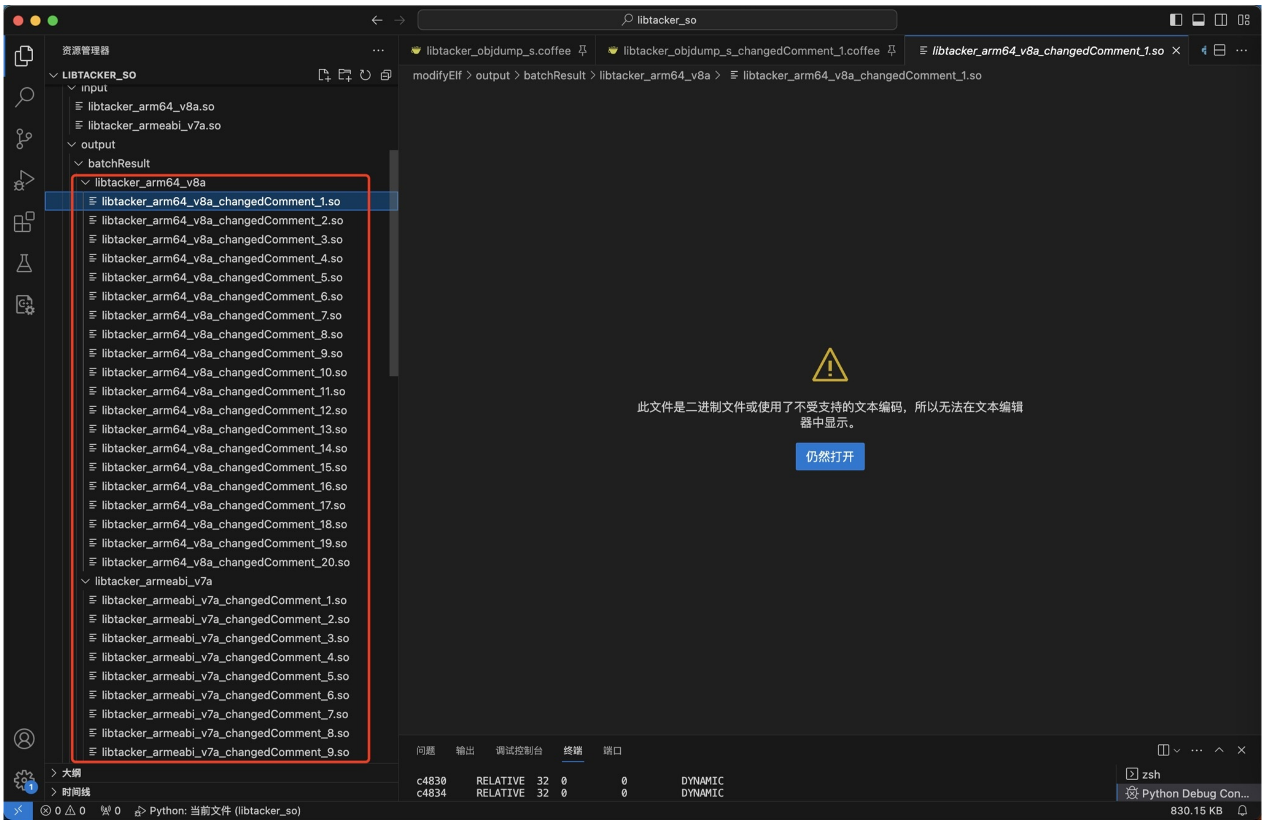
- 输出log日志

```
Loaded file:  modifyElf/input/libtacker_arm64_v8a.so
header= Magic:                         7f 45 4c 46
Class:                         CLASS64
Endianness:                    LSB
Version:                       CURRENT
OS/ABI:                        SYSTEMV
ABI Version:                   0
Machine type:                  AARCH64
File type:                     DYNAMIC
Object file version:           CURRENT
Entry Point:                   0x1a5c0
Program header offset:         0x40
Section header offset:         848344
Processor Flag:                0
Header size:                   64
Size of program header:        56
Number of program header:      9
Size of section header:        64
Number of section headers:     27
Section Name Table idx:        26


imported_functions= [<lief._lief.Function object at 0x104876f70>, <lief._lief.Function
object at 0x104877230>, <lief._lief.Function object at 0x104877370>, <lief._lief.Functi
on object at 0x1048772f0>, <lief._lief.Function object at 0x1048a99b0>, <lief._lief.Fun
ction object at 0x104967230>, <lief._lief.Function object at 0x1049668f0>, <lief._lief.
Function object at 0x10487cc30>, <lief._lief.Function object at 0x10487cbb0>, <lief._li
ef.Function object at 0x10487cc70>, <lief._lief.Function object at 0x10487ccf0>, <lief.
_lief.Function object at 0x10487ccb0>, <lief._lief.Function object at 0x10497a7f0>, <li
ef._lief.Function object at 0x104a45bf0> , <lief._lief.Function object at 0x104a47c30> , <
lief._lief.Function object at 0x104a474b0> , <lief._lief.Function object at 0x104a44470>
, <lief._lief.Function object at 0x1049fd8b0> , <lief._lief.Function object at 0x1049fcab
0> , <lief._lief.Function object at 0x104a7d6f0> , <lief._lief.Function object at 0x104a7
ffb0> , <lief._lief.Function object at 0x104a7fbf0> , <lief._lief.Function object at 0x10
4a7fb70> , <lief._lief.Function object at 0x104a7ebb0> , <lief._lief.Function object at 0
x104a7f5b0> , <lief._lief.Function object at 0x104a7efb0> , <lief._lief.Function object a
t 0x104a7c7f0> , <lief._lief.Function object at 0x1048e16b0> , <lief._lief.Function objec
t at 0x1048e21f0> , <lief._lief.Function object at 0x10492bd70> , <lief._lief.Function ob
ject at 0x104929cf0> , <lief._lief.Function object at 0x104a14430> , <lief._lief.Function
 object at 0x104a166f0> , <lief._lief.Function object at 0x1049d86f0> , <lief._lief.Funct
ion object at 0x1049db6f0> , <lief._lief.Function object at 0x10495c6f0> , <lief._lief.Fu
nction object at 0x10495f1f0> , <lief._lief.Function object at 0x10495ebb0> , <lief._lief
.Function object at 0x104a218b0> , <lief._lief.Function object at 0x104a211f0> , <lief._l
ief.Function object at 0x1048f98f0> , <lief._lief.Function object at 0x1049c20b0> , <lief
._lief.Function object at 0x1048cfeb0> , <lief._lief.Function object at 0x104890370> , <l
ief._lief.Function object at 0x104890f70> , <lief._lief.Function object at 0x104a0f030>]
exported_functions= [<lief._lief.Function object at 0x104876f70> , <lief._lief.Function
object at 0x104877230> , <lief._lief.Function object at 0x104877370> , <lief._lief.Functi
on object at 0x1048772f0> , <lief._lief.Function object at 0x1048a99b0> , <lief._lief.Fun
ction object at 0x104967230> , <lief._lief.Function object at 0x1049668f0> , <lief._lief.
Function object at 0x10487cc30> , <lief._lief.Function object at 0x10487cbb0> , <lief._li
ef.Function object at 0x10487cc70> , <lief._lief.Function object at 0x10487ccf0> , <lief.
_lief.Function object at 0x10487ccb0> , <lief._lief.Function object at 0x10497a7f0> , <li
ef._lief.Function object at 0x104a45bf0> , <lief._lief.Function object at 0x104a47c30> , <
lief._lief.Function object at 0x104a474b0> , <lief._lief.Function object at 0x104a44470>
, <lief._lief.Function object at 0x1049fd8b0> , <lief._lief.Function object at 0x1049fcab
0> , <lief._lief.Function object at 0x104a7d6f0> , <lief._lief.Function object at 0x104a7
ffb0> , <lief._lief.Function object at 0x104a7fbf0> , <lief._lief.Function object at 0x10
4a7fb70> , <lief._lief.Function object at 0x104a7ebb0> , <lief._lief.Function object at 0
x104a7f5b0> , <lief._lief.Function object at 0x104a7efb0> , <lief._lief.Function object a
t 0x104a7c7f0> , <lief._lief.Function object at 0x1048e16b0> , <lief._lief.Function objec
t at 0x1048e21f0> , <lief._lief.Function object at 0x10492bd70> , <lief._lief.Function ob
ject at 0x104929cf0> , <lief._lief.Function object at 0x104a14430> , <lief._lief.Function
 object at 0x104a166f0> , <lief._lief.Function object at 0x1049d86f0> , <lief._lief.Funct
ion object at 0x1049db6f0> , <lief._lief.Function object at 0x10495c6f0> , <lief._lief.Fu
nction object at 0x10495f1f0> , <lief._lief.Function object at 0x10495ebb0> , <lief._lief
.Function object at 0x104a218b0> , <lief._lief.Function object at 0x104a211f0> , <lief._l
ief.Function object at 0x1048f98f0> , <lief._lief.Function object at 0x1049c20b0> , <lief
._lief.Function object at 0x1048cfeb0> , <lief._lief.Function object at 0x104890370> , <l
ief._lief.Function object at 0x104890f70> , <lief._lief.Function object at 0x104a0f030> ,
 <lief._lief.Function object at 0x104a0db30> , <lief._lief.Function object at 0x104896770>
, <lief._lief.Function object at 0x1049144b0> , <lief._lief.Function object at 0x1048e553
0> , <lief._lief.Function object at 0x1048ac730> , <lief._lief.Function object at 0x1049a
4ef0> , <lief._lief.Function object at 0x1049a5670> , <lief._lief.Function object at 0x10
488b1b0> , <lief._lief.Function object at 0x104993570> , <lief._lief.Function object at 0
x104993fb0> , <lief._lief.Function object at 0x104992f30> , <lief._lief.Function object a
t 0x104a900f0> , <lief._lief.Function object at 0x104a90130> , <lief._lief.Function objec
t at 0x104a90170> , <lief._lief.Function object at 0x104a901b0> , <lief._lief.Function ob
ject at 0x104a901f0> , <lief._lief.Function object at 0x104a90230> , <lief._lief.Function
 object at 0x104a90270> , <lief._lief.Function object at 0x104a902b0> , <lief._lief.Funct
ion object at 0x104a902f0> , <lief._lief.Function object at 0x104a90330> , <lief._lief.Fu
nction object at 0x104a90370> , <lief._lief.Function object at 0x104a903b0> , <lief._lief
.Function object at 0x104a903f0>]
--------------------
name=
size= 0
contentLen= 0
-------------------
name= .note.android.ident
size= 152
contentLen= 152
-------------------
name= .note.gnu.build-id
size= 36
contentLen= 36
-------------------
name= .dynsym
size= 2832
contentLen= 2832
-------------------
name= .gnu.version
size= 236
contentLen= 236
-------------------
name= .gnu.version_r
size= 64
contentLen= 64
-------------------
name= .gnu.hash
size= 492
contentLen= 492
-------------------
name= .hash
size= 952
contentLen= 952
-------------------
name= .dynstr
size= 3097
contentLen= 3097
-------------------
name= .rela.dyn
size= 34896
contentLen= 34896
-------------------
name= .rela.plt
size= 1104
contentLen= 1104
-------------------
name= .gcc_except_table
size= 6496
contentLen= 6496
-------------------
name= .rodata
size= 13364
contentLen= 13364
-------------------
name= .eh_frame_hdr
size= 7612
contentLen= 7612
-------------------
name= .eh_frame
```

```
size= 36052
contentLen= 36052
--------------------
name= .text
size= 715872
contentLen= 715872
--------------------
name= .plt
size= 768
contentLen= 768
--------------------
name= .data.rel.ro
size= 11960
contentLen= 11960
--------------------
name= .fini_array
size= 16
contentLen= 16
--------------------
name= .init_array
size= 560
contentLen= 560
--------------------
name= .dynamic
size= 464
contentLen= 464
--------------------
name= .got
size= 192
contentLen= 192
--------------------
name= .got.plt
size= 392
contentLen= 392
--------------------
name= .data
size= 9688
contentLen= 9688
--------------------
name= .bss
size= 2768
contentLen= 0
--------------------
name= .comment
size= 198
contentLen= 198
--------------------
name= .shstrtab
size= 260
contentLen= 260
before: commentSection.content= <memory at 0x1049b5480>
commentSection contentType= <class 'memoryview'>
newContentMemoryView type= <class 'memoryview'>
after: commentSection.content= <memory at 0x1049b5540>
Saved to modifyElf/output/libtacker_arm64_v8a_changedComment.so
```

- 输出文件：（一次性的批量的多个）修改后的ELF文件：


modifyElf > output > batchResult > libtacker_arm64_v8a > libtacker_arm64_v8a_changedComment_1.so

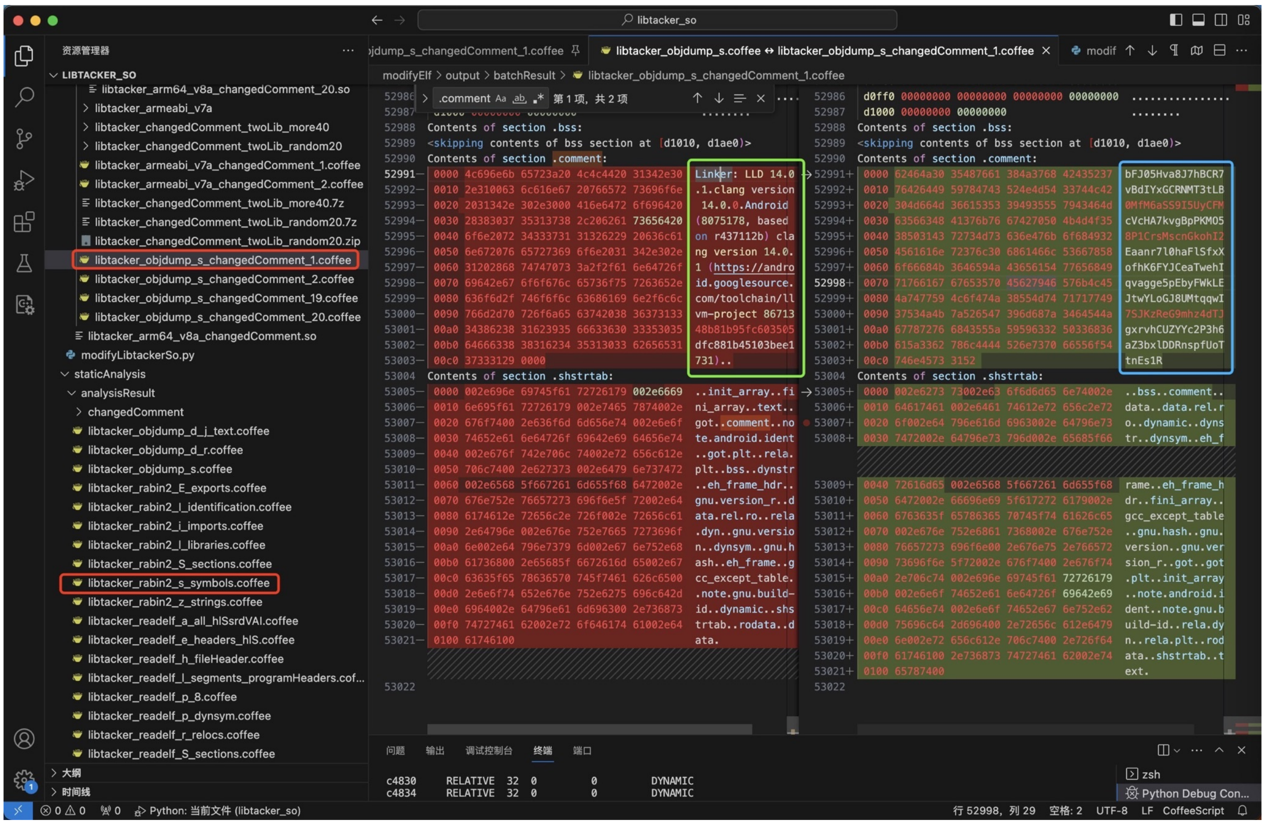
```
LIBTACKER_SO
  input
    libtacker_arm64_v8a.so
    libtacker_armeabi_v7a.so
  output
    batchResult
      libtacker_arm64_v8a
        libtacker_arm64_v8a_changedComment_1.so
        libtacker_arm64_v8a_changedComment_2.so
        libtacker_arm64_v8a_changedComment_3.so
        libtacker_arm64_v8a_changedComment_4.so
        libtacker_arm64_v8a_changedComment_5.so
        libtacker_arm64_v8a_changedComment_6.so
        libtacker_arm64_v8a_changedComment_7.so
        libtacker_arm64_v8a_changedComment_8.so
        libtacker_arm64_v8a_changedComment_9.so
        libtacker_arm64_v8a_changedComment_10.so
        libtacker_arm64_v8a_changedComment_11.so
        libtacker_arm64_v8a_changedComment_12.so
        libtacker_arm64_v8a_changedComment_13.so
        libtacker_arm64_v8a_changedComment_14.so
        libtacker_arm64_v8a_changedComment_15.so
        libtacker_arm64_v8a_changedComment_16.so
        libtacker_arm64_v8a_changedComment_17.so
        libtacker_arm64_v8a_changedComment_18.so
        libtacker_arm64_v8a_changedComment_19.so
        libtacker_arm64_v8a_changedComment_20.so
      libtacker_armeabi_v7a
        libtacker_armeabi_v7a_changedComment_1.so
        libtacker_armeabi_v7a_changedComment_2.so
        libtacker_armeabi_v7a_changedComment_3.so
        libtacker_armeabi_v7a_changedComment_4.so
        libtacker_armeabi_v7a_changedComment_5.so
        libtacker_armeabi_v7a_changedComment_6.so
        libtacker_armeabi_v7a_changedComment_7.so
        libtacker_armeabi_v7a_changedComment_8.so
        libtacker_armeabi_v7a_changedComment_9.so
```

此文件是二进制文件或使用了不受支持的文本编码，所以无法在文本编辑器中显示。

仍然打开

```
c4830       RELATIVE  32  0         0          DYNAMIC
c4834       RELATIVE  32  0         0          DYNAMIC
```

修改前后的内容对比


libtacker_objdump_s.coffee ↔ libtacker_objdump_s_changedComment_1.coffee

modifyElf > output > batchResult > libtacker_objdump_s_changedComment_1.coffee

修改前：

```
Contents of section .bss:
<skipping contents of bss section at [d1010, d1ae0)>
Contents of section .comment:
 0000 4c696e6b 65723a20 4c4c4420 31342e30  Linker: LLD 14.0
 0010 2e310063 6c616e67 20766572 73696f6e  .1.clang version
 0020 2031342e 302e3000 416e6472 6f696420   14.0.0.Android 
 0030 28383037 35313738 2c206261 73656420  (8075178, based 
 0040 6f6e2072 34333731 31326229 20636c61  on r437112b) cla
 0050 6e672076 65727369 6f6e2031 342e302e  ng version 14.0.
 0060 31202868 74747073 3a2f2f61 6e64726f  1 (https://andro
 0070 69642e67 6f6f676c 65736f75 7263652e  id.googlesource.
 0080 636f6d2f 746f6f6c 63686169 6e2f6c6c  com/toolchain/ll
 0090 766d2d70 726f6a65 63742038 36373133  vm-project 86713
 00a0 34386238 31623935 66633630 33353035  48b81b95fc603505
 00b0 64666338 38316234 35313033 62656531  dfc881b45103bee1
 00c0 37333129 0000                        731)..
Contents of section .shstrtab:
 0000 002e696e 69745f61 72726179 002e6669  ..init_array..fi
 0010 6e695f61 72726179 002e7465 7874002e  ni_array..text..
 0020 676f7400 2e636f6d 6d656e74 002e6e6f  got..comment..no
 0030 74652e61 6e64726f 69642e69 64656e74  te.android.ident
 0040 002e676f 742e706c 74002e72 656c612e  ..got.plt..rela.
 0050 706c7400 2e627373 002e6479 6e737472  plt..bss..dynstr
 0060 002e6568 5f667261 6d655f68 6472002e  ..eh_frame_hdr..
 0070 676e752e 76657273 696f6e5f 72002e64  gnu.version_r..d
 0080 6174612e 72656c2e 726f002e 72656c61  ata.rel.ro..rela
 0090 2e64796e 002e676e 752e7665 7273696f  .dyn..gnu.versio
 00a0 6e002e64 796e7379 6d002e67 6e752e68  n..dynsym..gnu.h
 00b0 61736800 2e65685f 6672616d 65002e67  ash..eh_frame..g
 00c0 63635f65 78636570 745f7461 626c6500  cc_except_table.
 00d0 2e6e6f74 652e676e 752e6275 696c642d  .note.gnu.build-
 00e0 6964002e 64796e61 6d696300 2e736873  id..dynamic..shs
 00f0 74727461 62002e72 6f646174 61002e64  trtab..rodata..d
 0100 61746100                             ata.
```

修改后：

```
 d0ff0 00000000 00000000 00000000 00000000  ................
 d1000 00000000 00000000                    ........
Contents of section .bss:
<skipping contents of bss section at [d1010, d1ae0)>
Contents of section .comment:
 0000 62464a30 35487661 384a3768 42435237  bFJ05Hva8J7hBCR7
 0010 76426449 59784743 524e4d54 33744c42  vBdIYxGCRNMT3tLB
 0020 304d664d 36615353 39493555 7943464d  0MfM6aSS9I5UyCFM
 0030 63566348 41376b76 67427050 4b4d4f35  cVcHA7kvgBpPKMO5
 0040 38503143 72734d73 636e476b 6f684932  8P1CrsMscnGkohI2
 0050 4561616e 72376c30 6861466c 53667858  Eaanr7l0haFlSfxX
 0060 6f66684b 3646594a 43656154 77656849  ofhK6FYJCeaTwehI
 0070 71766167 67653570 45627946 576b4c45  qvagge5pEbyFWkLE
 0080 4a747759 4c6f474a 38554d74 71717749  JtwYLoGJ8UMtqqwI
 0090 37534a4b 7a526547 396d687a 3464544a  7SJKzReG9mhz4dTJ
 00a0 67787276 6843555a 59596332 50336836  gxrvhCUZYYc2P3h6
 00b0 615a3362 786c4444 526e7370 66556f54  aZ3bxlDDRnspfUoT
 00c0 746e4573 3152                        tnEs1R
Contents of section .shstrtab:
 0000 002e6273 73002e63 6f6d6d65 6e74002e  ..bss..comment..
 0010 64617461 002e6461 74612e72 656c2e72  data..data.rel.r
 0020 6f002e64 796e616d 6963002e 64796e73  o..dynamic..dyns
 0030 7472002e 64796e73 796d002e 65685f66  tr..dynsym..eh_f
 0040 72616d65 002e6568 5f667261 6d655f68  rame..eh_frame_h
 0050 6472002e 66696e69 5f617272 6179002e  dr..fini_array..
 0060 6763635f 65786365 70745f74 61626c65  gcc_except_table
 0070 002e676e 752e6861 7368002e 676e752e  ..gnu.hash..gnu.
 0080 76657273 696f6e00 2e676e75 2e766572  version..gnu.ver
 0090 73696f6e 5f72002e 676f7400 2e676f74  sion_r..got..got
 00a0 2e706c74 002e696e 69745f61 72726179  .plt..init_array
 00b0 002e6e6f 74652e61 6e64726f 69642e69  ..note.android.i
 00c0 64656e74 002e6e6f 74652e67 6e752e62  dent..note.gnu.b
 00d0 75696c64 2d696400 2e72656c 612e6479  uild-id..rela.dy
 00e0 6e002e72 656c612e 706c7400 2e726f64  n..rela.plt..rod
 00f0 61746100 2e736873 74727461 62002e74  ata..shstrtab..t
 0100 65787400                             ext.
```

```
c4830       RELATIVE  32  0         0          DYNAMIC
c4834       RELATIVE  32  0         0          DYNAMIC
```

- `.comment` 修改前：

```
Contents of section .comment:
0000 4c696e6b 65723a20 4c4c4420 31342e30  Linker: LLD 14.0
0010 2e310063 6c616e67 20766572 73696f6e  .1.clang version
0020 2031342e 302e3000 416e6472 6f696420   14.0.0.Android
0030 28383037 35313738 2c206261 73656420  (8075178, based
0040 6f6e2072 34333731 31326229 20636c61  on r437112b) cla
0050 6e672076 65727369 6f6e2031 342e302e  ng version 14.0.
0060 31202868 74747073 3a2f2f61 6e64726f  1 (https://andro
0070 69642e67 6f6f676c 65736f75 7263652e  id.googlesource.
0080 636f6d2f 746f6f6c 63686169 6e2f6c6c  com/toolchain/ll
0090 766d2d70 726f6a65 63742038 36373133  vm-project 86713
00a0 34386238 31623935 66633630 33353035  48b81b95fc603505
00b0 64666338 38316234 35313033 62656531  dfc881b45103bee1
00c0 37333129 0000                        731)..
```

- `.comment` 修改后：

```
Contents of section .comment:
0000 62464a30 35487661 384a3768 42435237  bFJ05Hva8J7hBCR7
0010 76426449 59784743 524e4d54 33744c42  vBdIYxGCRNMT3tLB
0020 304d664d 36615353 39493555 7943464d  0MfM6aSS9I5UyCFM
0030 63566348 41376b76 67427050 4b4d4f35  cVcHA7kvgBpPKMO5
0040 38503143 72734d73 636e476b 6f684932  8P1CrsMscnGkohI2
0050 4561616e 72376c30 6861466c 53667858  Eaanr7l0haFlSfxX
0060 6f66684b 3646594a 43656154 77656849  ofhK6FYJCeaTwehI
0070 71766167 67653570 45627946 576b4c45  qvagge5pEbyFWkLE
0080 4a747759 4c6f474a 38554d74 71717749  JtwYLoGJ8UMtqqwI
0090 37534a4b 7a526547 396d687a 3464544a  7SJKzReG9mhz4dTJ
00a0 67787276 6843555a 59596332 50336836  gxrvhCUZYYc2P3h6
00b0 615a3362 786c4444 526e7370 66556f54  aZ3bxlDDRnspfUoT
00c0 746e4573 3152                        tnEs1R
```

子教程

此处列出相关子教程：

- 可执行文件格式
  - 可执行文件格式：ELF
  - 可执行文件格式：Mach-O

附录

下面列出相关参考资料。

参考资料

- 【已解决】给已有libtacker.so去改动信息
- 【已解决】用LIEF去修改ELF的so中的部分信息
- 【整理】Mach-O格式分析工具：LIEF
- 【整理】安卓相关：OTA格式
- 【已解决】新Mac M2 Max中如何安装x86_64版本rsync
-
- Comparison of executable file formats - Wikipedia
- DOS MZ executable - Wikipedia
- a.out - Wikipedia
- Portable Executable - Wikipedia
- COFF - Wikipedia
- Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2
- Creating and using static libraries in Linux ~ codingfreak
- Better understanding Linux secondary dependencies solving with examples
- 动态库与静态库的区别（linux vs windows vs mac）*51CTO*博客动态库和静态库的区别
- C/C++ 动态库(dll)与静态库(lib)*c++*编译出来的*lib*库和*c*库有什么区别—叶知秋@qqy的博客-CSDN博客
- 以.a(a为后缀)的文件类型是啥鸭？*.a*是什么文件子燕若水的博客-CSDN博客
- 静态库的格式*there is no dynamic section in this file*广敏的博客-CSDN博客
- 静态库的管理与文件格式分析 - 知乎
- 静态库（.a）和共享库（.so）的文件格式差异?
- Welcome to LIEF’s documentation! — LIEF Documentation (lief-project.github.io)
- Installation and Integration — LIEF Documentation (lief-project.github.io)
- Tutorials — LIEF Documentation (lief-project.github.io)
- 03 - Play with ELF symbols — LIEF Documentation (lief-project.github.io)
- 10 - Android formats — LIEF Documentation (archive.is)
- Hiding Behind ART - Black Hat 2015
- Dalvik and ART
- OAT internal structures
-