

# 同时使用多种版本的libc && 编译libc



pu1p (/u/82052c0e70a9) + 关注

1.6 2019.04.30 23:20\* 字数 940 阅读 132 评论 0 喜欢 4

(/u/82052c0e70a9)

参考: <https://bbs.pediy.com/thread-225849.htm> (<https://links.jianshu.com/go?to=https%3A%2F%2Fbbs.pediy.com%2Fthread-225849.htm>)

做pwn题的时候经常会遇到不同的libc, 最简单的解决方法就是LD\_PRELOAD了, 不过这个方法有一个局限. 就是如果libc的版本差太多的话就没办法使用了. 毕竟LD\_PRELOAD只是告诉ld.so该去哪儿加载libc, 可是ld.so也没有办法去加载不同版本的libc 于是我就只能傻乎乎地配了三个ubuntu的pwn环境, 虽然有docker可以用, 可是感觉在虚拟机里面又装一层docker实在有点蠢.... 总之既浪费时间又浪费空间 通过上面的帖子我才发现其实是可以在一个ubuntu中使用多个版本的libc的, 只要修改程序使得其使用对应版本的ld.so即可

elf文件有个段 PT\_INTERP, 其中存储了程序使用的ld.so的路径, 默认是这样

```
00pwn_scripts git:(master) x ldd test.out
linux-vdso.so.1 => (0x00007ffff7ffa000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7a0d000)
/lib64/ld-linux-x86-64.so.2 (0x00007ffff7dd7000)
```

默认使用 /lib64/ld-linux-x86-64.so.2

我们通过上面链接里面给出的脚本可以修改程序的这个段的内容, 使得其启动的时候使用我们提供的ld.so, 然后我们就可以修改其为特定版本的ld.so从而达到使用不同版本libc的目的了

下面就是修改为2.24版本的ld.so其使用 libc2.24

```
00pwn_scripts git:(master) x ldd test.out_ld_2.24.so
linux-vdso.so.1 => (0x00007ffff7ffa000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7a0d000)
./ld_2.24.so => /lib64/ld-linux-x86-64.so.2 (0x00007ffff7dd7000)
```

使用2.24版本的ld.so

我发现如果使用自己编译的libc的话即使不使用LD\_PRELOAD它也会找到对应的libc, 而不是默认的 /lib/x86\_64-linux-gnu/libc.so.6, 应该是在ld.so里面记录了libc的路径.

```
pwndbg> vmap
add vmap to history
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
0x400000 0x401000 r-xp 1000 0 /mnt/hgfs/ctf/ctf_games/00pwn_scripts/test.out_ld_2.24.so
0x600000 0x601000 r--p 1000 0 /mnt/hgfs/ctf/ctf_games/00pwn_scripts/test.out_ld_2.24.so
0x602000 0x603000 rw-p 1000 1000 /mnt/hgfs/ctf/ctf_games/00pwn_scripts/test.out_ld_2.24.so
0x7ffff7a3b000 0x7ffff7bd0000 r-xp 195000 0 /home/pulp/glibcs/glibc-2.24_out/lib/libc-2.24.so
0x7ffff7bd0000 0x7ffff7dd0000 ---p 200000 195000 /home/pulp/glibcs/glibc-2.24_out/lib/libc-2.24.so
0x7ffff7dd0000 0x7ffff7dd4000 r--p 4000 195000 /home/pulp/glibcs/glibc-2.24_out/lib/libc-2.24.so
0x7ffff7dd4000 0x7ffff7dd6000 rw-p 2000 199000 /home/pulp/glibcs/glibc-2.24_out/lib/libc-2.24.so
0x7ffff7dd6000 0x7ffff7dda000 rw-p 4000 0
```

没有使用LD\_PRELOAD也会找到对应的libc

而且我们还可以下载libc 2.24的源码从而实现源码级调试的目的. (具体操作参考这个SO  
([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F29955609%2Finclude-source-code-of-malloc-c-in-gdb)

[to=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F29955609%2Finclude-source-code-of-malloc-c-in-gdb](https://stackoverflow.com/questions/29955609/include-source-code-of-malloc-c-in-gdb)))

(/apps/redi  
utm\_source  
banner-lic

```

0x7ffff7ab62a6 <malloc+6>      mov     rax, qword ptr [rip + 0x31dc43]
0x7ffff7ab62ad <malloc+13>     mov     rax, qword ptr [rax]
0x7ffff7ab62b0 <malloc+16>     test    rax, rax
0x7ffff7ab62b3 <malloc+19>     jne     malloc+360 <0x7ffff7ab6408>

0x7ffff7ab62b9 <malloc+25>     mov     rax, qword ptr [rip + 0x31dad0]
0x7ffff7ab62c0 <malloc+32>     mov     rbp, rdi
0x7ffff7ab62c3 <malloc+35>     mov     rbx, qword ptr fs:[rax]

In file: /home/pulp/glibc/glibc-2.24/malloc/malloc.c
2909
2910 /*----- Public wrappers. -----*/
2911
2912 void *
2913 __libc_malloc (size_t bytes)
2914 {
2915     mstate ar_ptr;
2916     void *victim;
2917
2918     void *(*hook) (size_t, const void *)
2919         = atomic_forced_read (&__malloc_hook);
2920
00:0000| rsp 0x7fffffff1d8 -> 0x40053e (main+24) <- mov     eax, 0
01:0008| rbp 0x7fffffff1e0 -> 0x400550 (__libc_csu_init) <- push    r15
02:0010|      0x7fffffff1e8 -> 0x7ffff7a5b300 (__libc_start_main+240) <- mov     edi, eax
03:0018|      0x7fffffff1f0 <- 0x0
04:0020|      0x7fffffff1f8 -> 0x7ffff7fe2c8 -> 0x7ffff7fe55e <- 0x6667682f746e6d2f ('/mnt/h
[BA
> f 0      7ffff7ab62a0 malloc
f 1      40053e main+24
f 2      7ffff7a5b300 __libc_start_main+240
nwndbg>

```

源码调试malloc

贴一下我自己的脚本, 相比原帖修改了2点:

1. 将修改后的文件就放在当前文件夹而不是/tmp/pwn/下
2. 修改后的文件添加一个后缀, 表示使用的是哪个版本的ld.so, 因为现在经常有些题目不给libc.... 可能得试好几个



```
#coding:utf-8
from pwn import *
import os
def change_ld(binary, ld):
    if not os.access(ld, os.R_OK):
        log.failure("Invalid path {} to ld".format(ld))
        return None
    if not os.access(binary, os.R_OK):
        log.failure("Invalid path {} to binary".format(binary))
        return None
    binary = ELF(binary)
    path = './{}_{}'.format(os.path.basename(binary.path), ld.split('.')[-2])
    if os.access(path, os.F_OK):
        os.remove(path)
        print("remove exist file.....")
        return ELF(path)
    for segment in binary.segments:
        if segment.header['p_type'] == 'PT_INTERP':
            size = segment.header['p_memsz']
            addr = segment.header['p_paddr']
            data = segment.data()
            if size <= len(ld):
                log.failure("Failed to change PT_INTERP from {} to {}".format(data, ld))
                return None
            binary.write(addr, ld.ljust(size, '\x00'))
            break
    binary.save(path)
    os.chmod(path, 0b111000000) #rwx-----
    success("PT_INTERP has changed from {} to {}. Using temp file {}".format(data, ld, path))
    return ELF(path)
```

(/apps/redi  
utm\_source  
banner-clc

这个仓库 ([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fmatrix1001%2Fwelpwn%2Ftree%2Fmaster%2FPwnContext%2Flibs)

[to=https%3A%2F%2Fgithub.com%2Fmatrix1001%2Fwelpwn%2Ftree%2Fmaster%2FPwnContext%2Flibs](https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fmatrix1001%2Fwelpwn%2Ftree%2Fmaster%2FPwnContext%2Flibs))里面有编译好的各个版本的libc.so和ld.so

不过我还是建议自己编译libc, 一个版本也就20分钟不到就编译好了. 之后调试的时候会很方便

下面大概简单说了一下我编译libc的过程. 大家可以参考一下

## Appendix: 编译libc

官方编译教程 ([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fgnu.org%2Fsoftware%2Flibc%2Fmanual%2Fhtml_mono%2Flibc.html%23toc-Installing-the-GNU-C-Library)

[to=https%3A%2F%2Fgnu.org%2Fsoftware%2Flibc%2Fmanual%2Fhtml\\_mono%2Flibc.html%23toc-Installing-the-GNU-C-Library](https://links.jianshu.com/go?to=https%3A%2F%2Fgnu.org%2Fsoftware%2Flibc%2Fmanual%2Fhtml_mono%2Flibc.html%23toc-Installing-the-GNU-C-Library))

注意:

本教程没有考虑到编译过程中可能需要某些诸如gcc之类软件依赖的情况.

环境:

Ubuntu 16.04 server版

### 1. 下载glibc:

我选择的是glibc\_\*.tar.gz版本

#### 1. 官网 ([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fftp.gnu.org%2Fgnu%2Fglibc%2F)

[to=https%3A%2F%2Fftp.gnu.org%2Fgnu%2Fglibc%2F](https://links.jianshu.com/go?to=https%3A%2F%2Fftp.gnu.org%2Fgnu%2Fglibc%2F))



2. 教育网可以选择清华的镜像站 ([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fmirrors.tuna.tsinghua.edu.cn%2Fgnu%2Fglibc%2F)

[to=https%3A%2F%2Fmirrors.tuna.tsinghua.edu.cn%2Fgnu%2Fglibc%2F](https://links.jianshu.com/go?to=https%3A%2F%2Fmirrors.tuna.tsinghua.edu.cn%2Fgnu%2Fglibc%2F))

## 2. 解压

尽量不要再虚拟机的共享文件夹下解压, 可能会出现莫名其妙的问题, 速度可能也会很慢

a. `tar -xvf glibc_*.tar.gz`

b. 可以看到一个新的文件夹, 然后我们再新建两个文件夹用来存储编译结果:

(/apps/redi  
utm\_source  
banner-clic

image.png

第一文件夹是解压得到的, 后两个是新建的

## 3. 配置编译参数

a. 进入 `*_build` 文件夹, 执行

```
../glibc-2.24/configure '--prefix=/home/pu1p/glibcs/glibc-2.24_out'
```

这儿解释一下, glibc编译的时候需要在一个新的文件夹下编译, 不允许在源代码的目录中编译.

编译完成后还会产生一些诸如ldd等二进制文件, 最好使用--prefix声明一个文件夹来存储这些产生的文件, 否则如果存储到默认路径/usr/local可能会影响系统的正常运行

## 4. make:


a. make (大概10min)

b. make install (大概2min)

## 5. 寻找libc.so 和 ld.so

a. libc.so 就在glibc-2.24\_build目录下

b. ld.so 就在glibc-2.24\_build/elf 目录下

 pwn&RE (/nb/24012649)

举报文章 © 著作权归作者所有



pu1p (/u/82052c0e70a9)

写了 21261 字, 被 32 人关注, 获得了 34 个喜欢  
(/u/82052c0e70a9)

+ 关注



learn and share

喜欢 | 4




更多分享

(/apps/redi  
utm\_source  
banner-clic



(/p/7dd2ad568a69)

被以下专题收入，发现更多相似内容

 CTF Re... (/c/c7680c1e9407?  
utm\_source=desktop&utm\_medium=notes-included-collection)

读书笔记 - 《程序员的自我修养》 (/p/78a1e8d85e5f?utm\_campaign=...

一、温故而知新 1. 内存不够怎么办 内存简单分配策略的问题地址空间不隔离内存使用效率低程序运行的地址不确定 关于隔离：分为 虚拟地址空间和 物理地址空间 分段：把一段程序所需要的内存空间大小映射...

 SeanCST (/u/7283ea6ff5b6?  
utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

dll和so文件区别与构成 (/p/3fa8e813441a?utm\_campaign=maleskin...

动态链接，在可执行文件装载时或运行时，由操作系统的装载程序加载库。大多数操作系统将解析外部引用（比如库）作为加载过程的一部分。在这些系统上，可执行文件包含一个叫做import directory的表，该表...

 小5筒 (/u/5f3c4a5b736d?  
utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

强制升级glibc2.14到2.17问题 (/p/e7d43e39df0b?utm\_campaign=m...

源码编译安装，关键是恢复、比对。备份系统文件，用Live系统恢复原系统状态；比对发行版包管理器软件包的安装路径，移除冲突，比对发行版编译软件包时的编译选项，如果函数库没编译某些特性，可能会导...

 MrChenyz (/u/144d91c6f9ac?  
utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

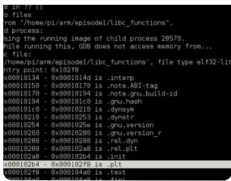
Linux动态链接库的创建与使用 (/p/f4f1804f38ac?utm\_campaign=mal...

1. 介绍 使用GNU的工具我们如何在Linux下创建自己的程序函数库?一个“程序函数库”简单的说就是一个文件包含了一些编译好的代码和数据，这些编译好的代码和数据可以在事后供其他的程序使用。程序函数...

 逍遥\_9353 (/u/c9e32c16e1a5?  
utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm



(/p/7ff3ff26dba3?



(/apps/redi  
utm\_source  
utm\_source=recomm  
banner-clic

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recomm  
**物联网上的 ARM 漏洞利用 (/p/7ff3ff26dba3?utm\_campaign=maleski...**

原文出自看雪论坛 发文动机 几周前我参加某个会议的时候，有个“物联网上的 ARM 漏洞利用课程”的议题我觉得很多干货，我也决定自己写一篇，给去不了现场的同学发些福利。我打算分为三个部分来写。当然...

看雪学院 (/u/fc7a20500211?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

**我们“本来的样子” (/p/be35e1fb50ab?utm\_campaign=maleskine&...**

今天，猴年就要结束了。这一年，匆匆赶路，有收获、有失去。这一年，不曾开心的笑，也不曾纵情地哭。相遇一些人，共过一些事，阴霾是否已经褪去？幸福是否正在到来？想说：“谢谢你！那些冰凉的时刻，...

董董23 (/u/46bf2945eff9?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

**变形金刚5.一个自挂东南枝的笑话 (/p/bcc9eb69a51c?utm\_campaign=...**

导演迈克尔贝以为抓住了中国观众的口味就认为掌控了致胜的法门，实在有点太可笑了。中国观众之所以喜欢这个系列，是因为国产烂片太多了，不如看个特效大片来的实在一些。而迈克尔贝却以为中国观众就只...

王廿 (/u/cffc38945524?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

(/p/127c24a5c765?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recomm  
**80㎡舒适北欧2室2厅，清新自然的小资格调【沈阳方林装饰】 (/p/127c24...**

沈阳方林装饰官方联系人：李经理（咨询电话17046101918） 欢迎关注小编博客账号  
http://blog.sina.com.cn/u/7017017887 设计理念 Idea 美好的事物固然值得赞美，但透过假想去构思未...

方林装饰李经理 (/u/6ccdb420a1ef?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

**有酒性，才是真男人！ (/p/aa4e38d9ab5c?utm\_campaign=maleskine...**

什么是福？福，就是心大！不为无聊的人，烦恼；不为无聊的事，计较！什么是乐？乐，就是心态好！不管穷富，吃饱睡好；不管咋活，舒坦重要！谁的人生敢说十全十美，谁的生活不是酸甜苦辣，谁敢保...

GZJYT (/u/66880c3f61a6?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm

**期中已至， (/p/48aba82b40ce?utm\_campaign=maleskine&utm\_co...**

哈哈了

欣均 (/u/eb2ef986ce08?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recomm



(/apps/redi  
utm\_source  
banner-clic

