

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： ELF文件与程序链接**

**院 系 ：计算机科学与技术**

**专业班级 ： CS2204**

**学 号 ： U202215458**

**姓 名 ： 柯晓豪**

**指导教师 ： 金良海**

**2024 年 5 月 29 日**

**一、实验目的与要求**

通过修改给定的可重定位的目标文件（链接炸弹），加深对可重定位目标文件格式、目标文件的生成、以及链接的理论知识的理解。

实验环境：Ubuntu

工具：GCC、GDB、readelf、hexdump、hexedit、od等。

**二、实验内容**

**任务** 链接炸弹的拆除

**在二进制层面，逐步修改构成目标程序“linkbomb”的多个二进制模块（“.o文件”），然后链接生成可执行程序，要求可执行程序运行能得到指定的效果。修改目标包括可重定位目标文件中的数据、机器指令、重定位记录等。**

**1、第1关 静态数据与ELF数据节**

修改二进制可重定位目标文件 phase1.o 的数据节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序，可以输出自己的学号。

**2、第2关 简单的机器指令修改**

修改二进制可重定位目标文件 phase2.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_2.c 中，有一个静态函数 static void myfunc( ) ，要求在 do\_phase 函数中调用myfunc( )，显示信息myfunc is called. Good!。

**3、第3关 有参数的函数调用的机器指令修改**

修改二进制可重定位目标文件 phase3.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_3.c 中，有一个静态函数 static void myfunc(int offset) ，要求在 do\_phase函数中调用myfunc(pos )，将do\_phase的参数pos直接传递myfunc，显示相应的信息。

**4、第4关 有局部变量的机器指令修改**

修改二进制可重定位目标文件 phase4.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_4.c 中，有一个静态函数 static void myfunc(char \*s) ，要求在 do\_phase 函数中调用myfunc(s )，显示出自己的学号。

**5、第5关 重定位表的修改**

修改二进制可重定位目标文件 phase5.o 的重定位节中的内容（不允许修改代码节和数据节），使其与main.o链接后，生成的执行程序运行时，显示Class Name : Computer Foundation. Teacher Name : Xu Xiangyang。

**6、第6关 强弱符号**

不准修改 main.c 和phase6.o，通过增补一个文件，使得程序链接后，能够输出自己的学号。

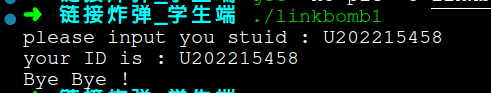
#gcc -no-pie -o linkbomb6 main.o phase6.o phase6\_patch.o

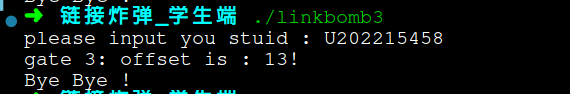
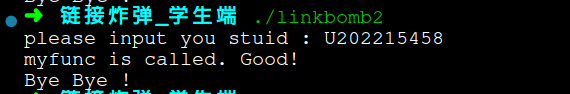
**7、第7关 只读数据节的修改**

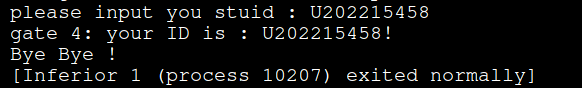
修改 phase7.o 中只读数据节（不准修改代码节），使其与main.o链接后，能够输出自己的学号。

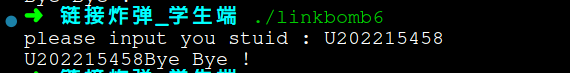
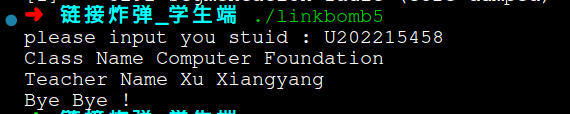
**三、实验记录及问题回答**

**（1）实验任务的实验记录**









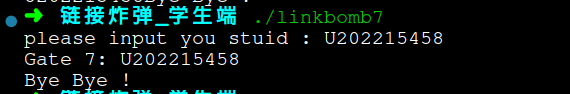


图1运行结果截图

**（2） 描述修改各个文件的基本思想**

**phase1:**

先对main.o和phase1.o进行链接，然后运行linkbomb1,结果如图2。然后使用hexedit编辑查看phase1.o,可以找到那一段字符串，然后查看ascii码表，从n开始修改字符串，如图3所示。最后运行结果符合预期。

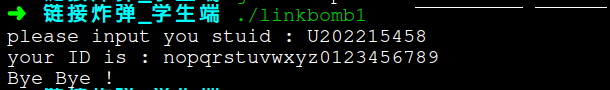
图2 未修改时链接后的运行结果

图3 修改后的结果

**phase2:**

反汇编do\_phase函数，如图4,我们可以在do\_phase中用新的指令替代nop指令。反汇编myfunc得到其入口地址为0x40131d，如图5。我们可以在nop处将其修改为call myfunc,这时候其机器码为e8 0x40131d-0x401344-5=0xffffffd9.修改phase2.o,修改后如图6.



图4 do\_phase反汇编截图

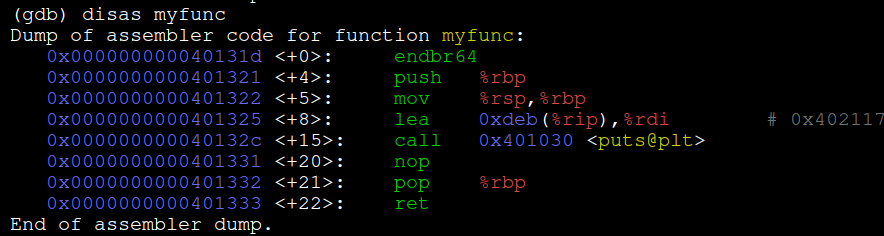


图5 myfunc反汇编截图

图6 修改后的结果

**phase3:**

反汇编myfunc如图7,发现edi中保存着offset,rdi保存着输出格式。故在do\_phase中加入call myfunc指令。与phase2类似，对应着0x40131d-0x401350-5=0xffffffc8,修改后的结果如图8.

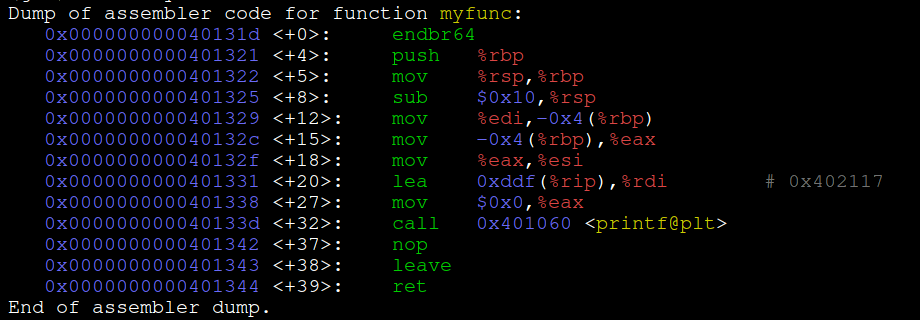
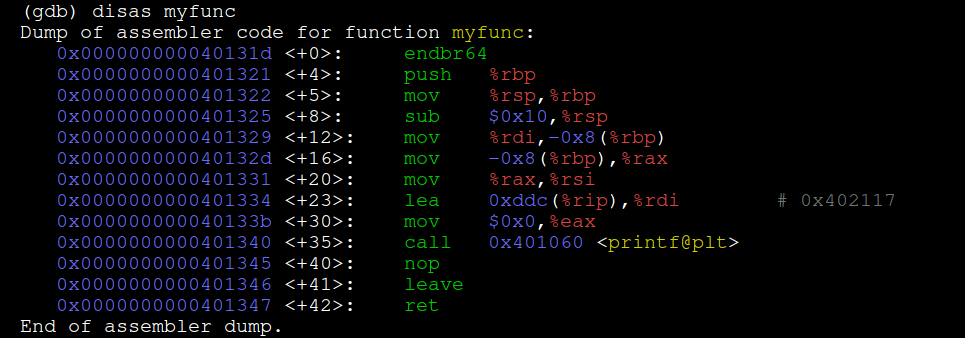
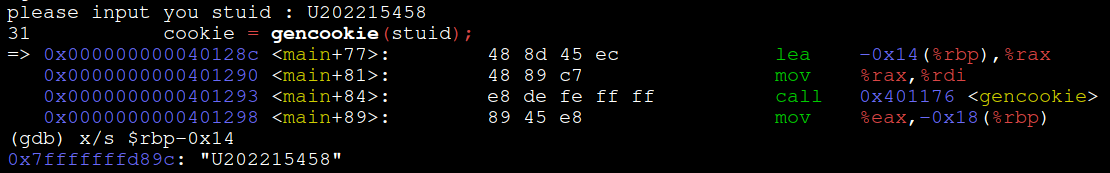
图7 myfunc反汇编截图

图8 修改后结果

**phase4:**

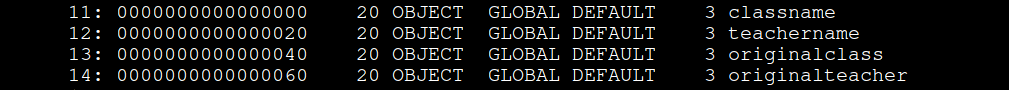
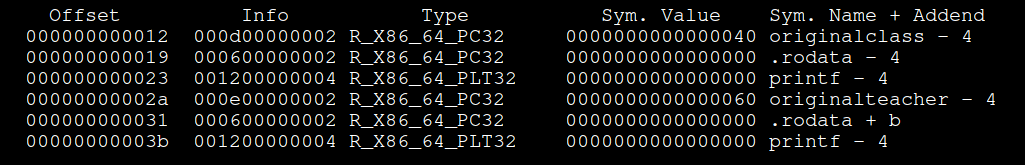
反汇编myfunc,发现rdi为学号的首地址，如图9.对main函数进行调试得知学号以0x7fffffffd89c为首地址，如图10,所以需要用movabs $0x7fffffffd89c,%rax, mov %rax,%rdi和call myfunc来替换nop.call myfunc的机器码计算方式同phase3.对应的机器指令为48 b8 9c d8 ff ff ff 7f 00 00 48 89 c7 e8 8d ff ff ff.

图9 myfunc反汇编截图

图10 gdb调试过程

**phase5:**

若不对phase5.o进行任何修改，其输出的课程名和教师名为originalclass和originalteacher.而我们要将originalclass改成classname，将originalteacher改成teachername.使用readelf -a phase5.o,查看elf文件，这里给出对解题有帮助的部分，如图11.可以发现，.rela.text的偏移量为0x3f8,现在需要将originalclass和originalteacher的Info部分改变，而Info的前两个字节记录的是符号在符号表中的序号，如originalclass位于.symtab的第13个条目，那么其info的前两个字节便是000d.观察.symtab,不难发现应将originalclass的info 前两个字节改成000b,将originalteacher的改成000c.然后再重新链接，程序如预期运行。

图11 elf头文件部分截图

**phase6:**

添加一个文件phase6\_patch.c,编写如图12所示的代码。这样链接之后便能够输出自己的学号了。

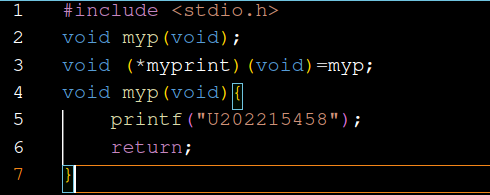
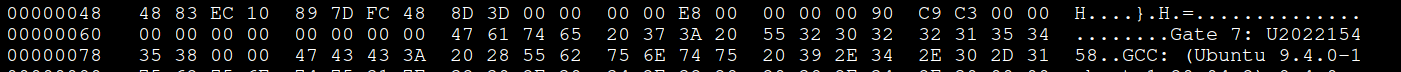


图12 phase6\_patch.c代码截图

phase7:

先不进行修改，可以发现其输出的字符串为U202212345,然后通过readelf工具查看.rodata的偏移量0x68，如图13，并使用hexedit定位到.rodata节，将U202212345改为自己的学号，如图14。这样链接之后便能输出自己的学号了。

图13 查看.rodata偏移量截图

图14 修改后截图

**四、体会**

经过链接炸弹的破解，我对节的概念和如何修改相应节的内容有了更深刻的认识，掌握了 根据偏移地址计算实际地址的方法。明白了函数压栈的内容在其运行中的作用。总而言之， 在破解炸弹的过程中我对程序的链接有了更好的认识。