

**课 程 实 验 报 告**

**课程名称： 计算机系统基础实验**

**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 ： CS2205**

**学 号 ： U202215510**

**姓 名 ： 徐新飏**

**指导教师 ： 李专**

**2024 年 5 月 6 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**

**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。**

**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。

**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!。

**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。

**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。

**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。

**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

**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**

phase1:

直接运行./linkbomb1的结果如下，要求要将其修改为自己的学号。

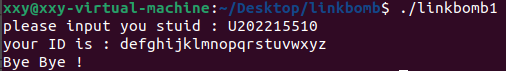


图1.1

使用readelf -S phase1.o查看.data节的位置，发现其偏移量offset为0x80。



图1.2

使用hexedit打开文件后，查找与该字符串相对应的位置。填入学号对应的ASCII码，修改学号为U202215510，最后记得补上‘\0’，保存并退出。



图1.3

重新使用gcc生成执行程序，输入学号，发现程序会输出自己的学号。

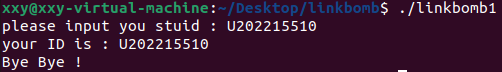


图1.4

phase2:

修改二进制可重定位目标文件phase2.o的代码节内容，使其与main.o链接后能够在do\_phase函数中调用myfunc( )。

查看phase2.o的段头表和符号表，可以确定do\_phase函数在phase2.o中的位置为0x40+0x17=0x47。

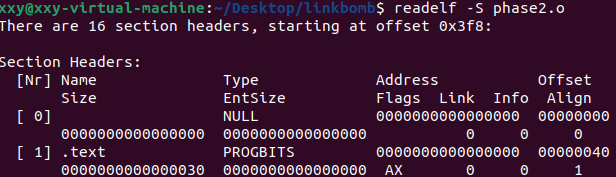




图2.1

objdump查看phase2.o中do\_phase函数，发现其中全是空操作，故需要从phase2.o中开始添加反汇编代码，添加地址应当在0x40+0x22=0x62到0x6d之间。

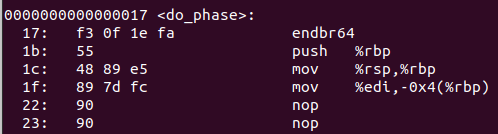




图2.2

添加call指令，机器码为e8,由于偏移量 = 跳转到的地址 - call指令后一条指令长度的起始地址。故本题偏移量机器码为0x00-(0x22+0x5)=-39，其补码为0xffffffd9,按小端存储的原则将其修改为d9 ff ff ff即可，如图所示。



图2.3

链接后运行。

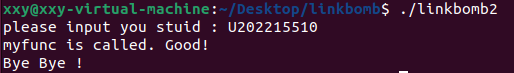


图2.4

phase3:

重点在调用函数时传入参数，反汇编查看phase3.o，其存在很多空操作。其中参数由edi传入到-0x4(%rbp)中。

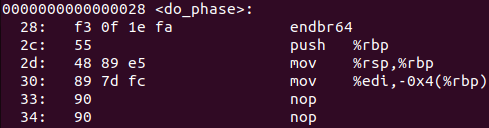




图3.1

在myfunc上发现参数保存在edi并传给eax，手动编写反汇编代码查看机器码。

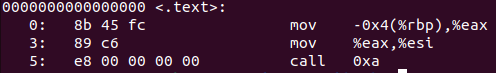


图3.2

修改二进制文件,由于是在0x38处插入call语句，调用地址为0x00-(0x38+0x5)=-61=0xffffffc3,如图修改。



图3.3

链接后通过测试

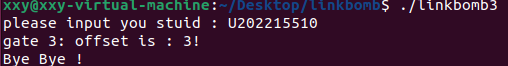


图3.4

phase4:

objdump观察myfunc，发现其参数由rdi通过rbp-0x8传到rax中。

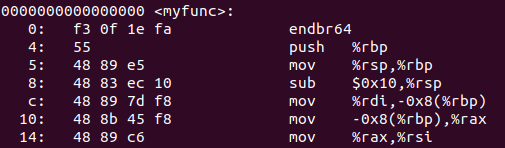


图4.1

观察do\_phase函数，rbp-0x13到rbp-0xa一共十个字节按小端序存储学号,rbp-0x9置零表示字符串结束，只需将字符串的首地址rbp-0x13传给edi。

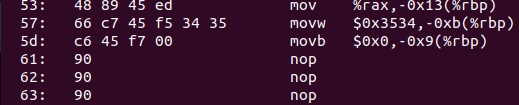


图4.2

编写反汇编代码查看机器码。



图4.3

由此，call函数调用myfunc的偏移量为0x00-(0x65+0x5)=-106=0xffffff96，根据readelf得到的text位置更改二进制编码。注意需要同时修改学号代码赋值部分和调用的代码部分。







图4.4

链接程序后运行。

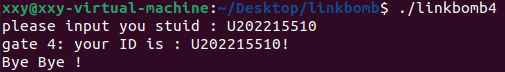


图4.5

phase5:

readelf查看phase5.o的符号表，发现classname比originalclass小2，teachername比originalteachername小2，在重定位节，应当用classname替换originalclassname,应当用teachername替换originalteachername。

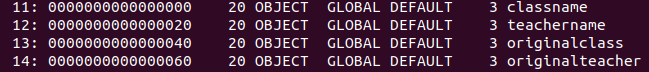


图5.1

查看节头表，找到.rela.text和.data.rel.local的描述信息。

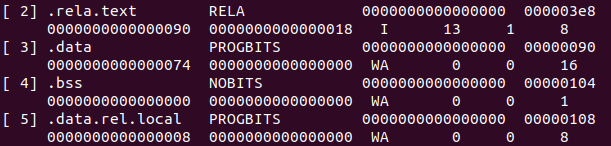


图5.2

查看.rela.text的详细信息，info字段用来指定符号表中的符号索引，因此需要修改info的值。

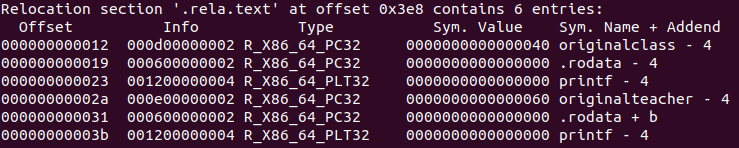


图5.3

Hexedit打开二进制文件找到相应部分，将0d变为0b，将0e变为0c，修改二进制文件。



图5.4

链接后运行。

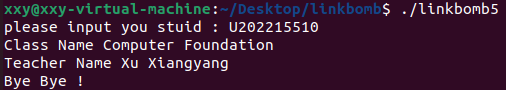


图5.5

phase6：

直接链接运行后发现，需要编写自己的文件phase6\_patch.c。

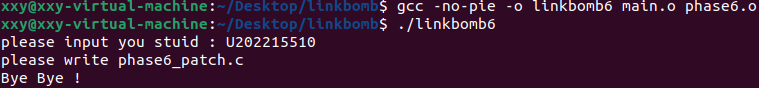
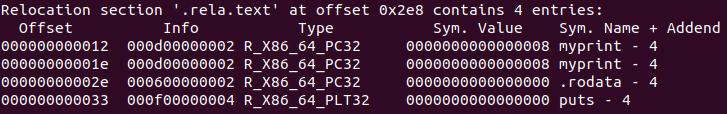


图6.1

查看phase6重定位节表，符号表等信息。



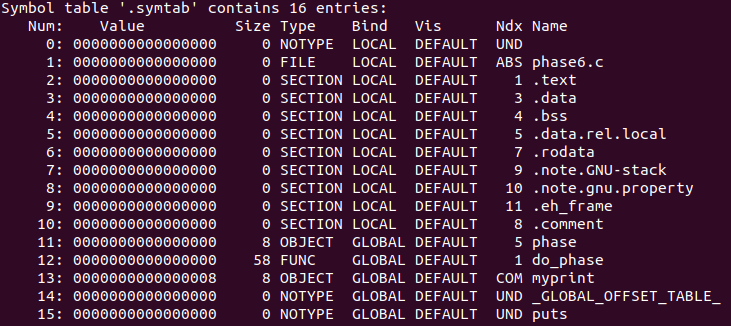


图6.2

这表面了程序中存在myprint的全局对象，其大小为8个字节而非一个函数的大小，COM意味着Common Symbol，表示是一个未初始化的全局变量，由于64位系统的函数指针通常大小为8个字节，合理猜测其为一个函数指针，编写相应函数来完成学号的输出。

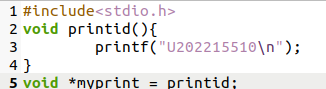


图6.3

链接程序后运行通过。

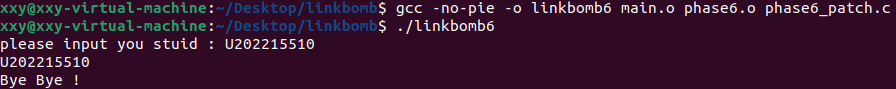


图6.4

phase7:

要求修改只读数据节来输出自己的学号，用readelf查看节头表，偏移量为0x68。



图7.1

Hexedit找到.rodata对应位置进行修改。



图7.2

修改后链接运行通过。

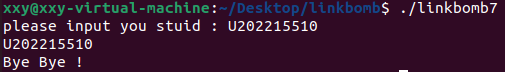


图7.3

**（2）描述修改各个文件的基本思想**

phase1.o：查看节头表找到.data节的位置，使用hexedit修改.data节的数据为自己的学号。

phase2.o：查看节头表找到.text节的位置，根据call指令的偏移量 = 跳转到的地址 - call指令后一条指令长度的起始地址来计算call指令需要跳转到的地址为0x00-(0x22+0x5)=-39，其补码为0xffffffd9。编写反汇编代码并转为机器码，用hexedit将其填入.text节对应位置。

phase3.o：思路同上，只是这回需要观察反汇编代码，来确定参数是由edi传入到-0x4(%rbp)中，由此利用esi重新传参，重新编写反汇编代码并计算call的跳转地址，用hexedit将其填入.text节对应位置。

phase4.o：观察反汇编代码注意到字符串的首地址是rbp-0x13，将其作为函数参数传入rdi，重新编写反汇编代码并计算call的跳转地址，将修改后的代码和学号用hexedit将其填入.text节对应位置。

phase5.o：查看符号表和重定位表，发现要修改的变量比原变量的info值小2，从节头表获取到.rela.text的偏移量，根据.rela.text的详细信息，用hexedit找到对应info值的位置，将其索引部分改小2即可。

phase6.o：查看重定位节表和符号表，发现myprint是一个弱符号，是一个未初始化的全局变量，根据其大小为8推测出其为一个函数指针，编写phase6\_patch.c在其中实现打印自己学号的函数和函数指针的连接。

phase7.o：查看节头表来确定.rodata节的位置，用hexedit在.rodata节中找到学号的16进制码，将其修改为自己的学号。

**四、体会**

通过此次实验，我在很多方面都取得了显著的收获和提高。

我更加熟悉了由于在每一关卡中，我都需要直接使用hexedit操作ELF文件的不同部分，这使我对ELF文件的结构和各个节的作用有了更深入的了解，特别是通过修改数据节、代码节和重定位节等，我熟悉了这些节在程序链接和执行过程中的具体作用。

我对于linux相关工具使用熟练度提高，实验过程中，我大量使用了GCC、GDB、readelf、hexdump、hexedit等工具来帮助我查看和修改二进制文件，分析反汇编代码，调试程序等。

此次实验让我系统性地学习和掌握了可重定位目标文件的格式和操作技巧，深入理解了目标文件的生成和链接过程。通过对多个二进制模块的逐步修改，不仅提升了实际操作能力，还加深了对底层理论知识的理解，使我在二进制文件操作、调试和程序链接方面的能力得到了全面提升。

这些知识和技能不仅对我今后的课程学习有重要帮助，也为我未来从事相关领域的工作打下了坚实基础，这次实验是一次非常有价值的学习和实践经历。