哈爾濱工業大學

实验报告

实 验（五）

题　　目 LinkLab

链接

专　　业 计算机类

学　　号 1170300817

班　　级 1703008

学　　生 林之浩

指 导 教 师 郑贵滨

实 验 地 点 G712

实 验 日 期 2018.11.19

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 实验目的

理解链接的作用与工作步骤
掌握 ELF 结构、符号解析与重定位的工作过程
熟练使用 Linux 工具完成 ELF 分析与修改

1.2 实验环境与工具

1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

1.2.2 软件环境

Windows7 64 位以上；VirtualBox/Vmware 11 以上；Ubuntu 16.04 LTS 64 位/优麒麟 64 位；

1.2.3 开发工具

Visual Studio 2010 64 位以上；GDB/OBJDUMP；DDD/EDB 等

1.3 实验预习

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
请按顺序写出 ELF 格式的可执行目标文件的各类信息。
请按照内存地址从低到高的顺序，写出 Linux 下 X64 内存映像。
请运行“LinkAddress -u 学号 姓名” 按地址顺序写出各符号的地址、空间。并按照 Linux 下 X64 内存映像标出其所属各区。
gcc -o LinkAddress linkaddress.c
请按顺序写出 LinkAddress 从开始执行到 main 前/后执行的子程序的名字。(gcc 与 objdump/GDB/EDB)

第 2 章 实验预习

2.1 请按顺序写出 ELF 格式的可执行目标文件的各类信息（5 分）

ELF 头
段头部表
.init
.text
.rodata
.data
.bss
.symtab
.debug
.line
.strtab
节头部表

ELF 头：描述文件的总体格式。还包括程序的入口点，及程序要运行时所要执行的第一条指令的地址。
段头部表：用于将连续的文件节映射到运行时的内存段
.init:定义了一个叫做_init 的小程序，在程序初始化代码会用到。
.text:已编译程序的机器代码
.rodata:只读数据。
.data:已初始化的全局和静态 C 变量。
.bss:未初始化的全局和静态 C 变量。
.symtab:符号表，存放程序中定义和引用的函数和全局变量的信息。
.debug:调试符号表，只有在以-g 选项编译时才会有，其条目是程序中定义的局部变量和类型定义。
.line:原始 c 源程序中的行号和.text 机器指令之间的映射，只有在以-g 选项编译时才会有。
.strtab:字符串表.
节头部表:描述目标文件的节

2.2 请按照内存地址从低到高的顺序，写出 Linux 下 X64 内存映像

（5 分）

内核内存	
用户栈 （运行时 创建）	%rsp 栈指针
（栈-向下） . . . （映射区域-向上）	
共享库的内存映射区域	
. . . （堆-向上）	Brk
运行时堆 （由 malloc 创建）	
读/写段 （.data,.bss）	从可执行文件中加载
只读代码段 （.init,.text,.rodata）	
. . .	

2.3 请运行“LinkAddress -u 学号 姓名” 按地址顺序写出各符号的地址、空间。并按照 Linux 下 X64 内存映像标出其所属各区（5 分）

所属	符号、地址、空间（从小到大）
用户栈 （运行时 创建）	argv[0] 0x7fffce7ad22e 140736657543726 ./linkaddress argv[1] 0x7fffce7ad23c 140736657543740 -u argv[2] 0x7fffce7ad23f 140736657543743 1170300817 argv[3] 0x7fffce7ad24a 140736657543754 linzhihao local 0x7fffce7abe30 140736657538608

	argc0x7fffce7abe2c 140736657538604 argv 0x7fffce7abf58 140736657538904
共享库的内存映射区域	printf 0x7f71f874be80 140127476432512 malloc 0x7f71f877e070 140127476637808 free 0x7f71f877e950 140127476640080 p3 0x7f71f8cc4010 140127482167312 exit 0x7f71f872a120 140127476293920 p1 0x7f71e86e6010 140127207579664 p4 0x7f71a86e5010 140126133833744 p5 0x7f71286e4010 140123986346000
只读代码段（.init,.text,.rodata）	big array 0x556e9bacf040 93933546565696 huge array 0x556e5bacf040 93932472823872 global 0x556e5bacf02c 93932472823852 p2 0x556e9dd0567093933582440048 show_pointer 0x556e5b8cd81a 93932470720538 useless 0x556e5b8cd84d 93932470720589 main 0x556e5b8cd858 93932470720600

2.4 请按顺序写出 LinkAddress 从开始执行到 main 前/后执行的子程序的名字(使用 gcc 与 objdump/GDB/EDB)（5 分）

在 main 函数之前执行

_dl_init
_init
_dl_vdso_vsym
_dl_lookup_symbol_x
do_lookup_x
strcmp
_dl_vdso_vsym
__strrchr_avx2
__init_misc
__GI___ctype_init
init_cacheinfo

intel_check_word
_start
__libc_start_main
__GI___cxa_atexit
__new_exitfn
__GI___cxa_atexit
__libc_csu_init
__libc_csu_init
frame_dummy
register_tm_clones
_setjmp
__sigsetjmp
__sigjmp_save

在 main 函数之后执行：

__libc_start_main
__GI_exit
__run_exit_handlers
__GI___call_tls_dtors
_dl_fini
rtld_lock_default_lock_recursive
_dl_sort_maps
memset
__do_global_dtors_aux
__cxa_finalize@plt
__unregister_atfork
deregister_tm_clones
__do_global_dtors_aux
_IO_cleanup
_IO_flush_all_lockp
_IO_new_file_setbuf
_IO_new_file_sync
_IO_default_setbuf

第 3 章 各阶段的原理与方法

每阶段 40 分，phasex.o 20 分，分析 20 分，总分不超过 80 分

3.1 阶段 1 的分析

程序运行结果截图：

```
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ gcc -m32 -o linkbomb1 main
.o phase1.o
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ ./linkbomb1
1170300817
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$
```

分析与设计的过程：先将未被修改的 phase1.o 与 main 链接输出结果

```
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ ./linkbomb1
Ny2n7DNC3QP7 29XR FZCrvl9LdzExC7YMBOtOdYEiTlCZhhvpScv1PncJr     t808lGlUZG
```

用 hexedit 打开 phase1.o，复制该字符串并用 hexedit 搜索

```
0x0FF0 0000 0000 0000 0000 2D05 0000 0000 0000  ........-.......
0x1000 0000 0000 0420 0000 0000 0000 0000 0000  ..... ..........
0x1010 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x1020 6F7A 5137 6E66 4B6D 6E49 5366 6E64 4E79  ozQ7nfKmnISfndNy
0x1030 326E 3744 4E43 3351 5037 2032 3958 5220  2n7DNC3QP7 29XR
0x1040 465A 4372 766C 394C 647A 4578 4337 594D  FZCrvl9LdzExC7YM
0x1050 424F 744F 6459 4569 546C 435A 6868 7670  BOtOdYEiTlCZhhvp
0x1060 5363 7631 506E 634A 7209 7438 3038 6C47  Scv1PncJr.t8081G
0x1070 6C55 5A47 0900 0000 8505 0000 4743 433A  lUZG....?...GCC:
0x1080 2028 5562 756E 7475 2037 2E33 2E30 2D32   (Ubuntu 7.3.0-2
0x1090 3775 6275 6E74 7531 7E31 382E 3034 2920  7ubuntu1~18.04)
0x10A0 372E 332E 3000 4743 433A 2028 5562 756E  7.3.0.GCC: (Ubun
```

就找到了.data 节中的字符串内容，对其修改。

```
0 0000 0000 0000 0000 0000 0000 0000 0000  ................
0 6F7A 5137 6E66 4B6D 6E49 5366 6E64 3131  ozQ7nfKmnISfnd11
0 3730 3330 3038 3137 0000 2032 3958 5220  70300817.. 29XR
0 465A 4372 766C 394C 647A 4578 4337 594D  FZCrvl9LdzExC7YM
0 424F 744F 6459 4569 546C 435A 6868 7670  BOtOdYEiTlCZhhvp
0 5363 7631 506E 634A 7209 7438 3038 6C47  Scv1PncJr.t8081G
0 6C55 5A47 0900 0000 0000 0000 8B04 24C3  lUZG........?.$?
0 0047 4343 3A20 2855 6275 6E74 7520 372E  .GCC: (Ubuntu 7.
0 332E 302D 3136 7562 756E 7475 3329 2037  3.0-16ubuntu3) 7
```

改成学号 1170300817，并在末尾加上 00 表示字符串结束。

重新链接编译运行完成要求。

```
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ gcc -m32 -o linkbomb1 main
.o phase1.o
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ ./linkbomb1
1170300817
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$
```

3.2 阶段 2 的分析

程序运行结果截图：

```
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ ./linkbomb2
1170300817
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$
```

分析与设计的过程：

根据 ppt 的指导，先对 phase.o 进行反汇编

```
00000041 <do_phase>:
  41:   55                      push   %ebp
  42:   89 e5                   mov    %esp,%ebp
  44:   e8 fc ff ff ff          call   45 <do_phase+0x4>
                45: R_386_PC32  __x86.get_pc_thunk.ax
  49:   05 01 00 00 00          add    $0x1,%eax
                4a: R_386_GOTPC _GLOBAL_OFFSET_TABLE_
  4e:   90                      nop
  4f:   90                      nop
  50:   90                      nop
```

本题需要在 do_phase 中 call 如下函数，并且利用这个函数调用 puts 来完成输出。

```
00000000 <pHbfHxVh>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
```

观察 strcmp 函数，发现它在被调用前 pushl 两个变量，

```
add     $0x1a13,%ebx
sub     $0x8,%esp
lea     -0x18b0(%ebx),%eax
```

pushl　0x8(%ebp)是调用者函数保存的参数，

```
add     $0x1a13,%ebx
sub     $0x8,%esp
lea     -0x18b0(%ebx),%eax
```

eax 则来自 add　　$0x1a13,%ebx 和 lea　　-0x18b0(%ebx),%eax 两句

```
lea     -0x18b0(%ebx),%eax
push    %eax
```

为了观察这个位置存的是什么数据我们用-dx 参数运行 objdump

objdump -dx phase2.o > phase2asm.txt

```
c:    81 c3 02 00 00 00        add     $0x2,%ebx
        e: R_386_GOTPC  _GLOBAL_OFFSET_TABLE_
```

结合 elf 文件的 neir 我们发现程序试图访问 rodata 节的内容

```
00000017  00000509 R_386_GOTOFF       00000000    .rodata
```

查看 elf 发现 rodata 节偏移量为 b5

```
[ 7] .rodata           PROGBITS        00000000 0000b5 00000b 00
```

然后用 hexedit 查看 b5

```
0x0B0 9090 905D C331 3137 3033 3030 3831 3700  ???]?1170300817.
```

发现存的内容就是我的学号 1170300817。既然以及有我的学号在内存里了，只要把 push 如栈的参数指向已有的我的学号，这样相当于就是让 strcmp 函数比较了两个一摸一样的东西。

我们发现函数通过

```
5ca:    8d 83 50 e7 ff ff        lea     -0x18b0(%ebx),%eax
```

来获得存在 rodata 节的里的我的学号于是我们照搬这一句，

然后把 eax 入栈，之后再 call 这个特殊的函数

```
55:    e8 a6 ff ff ff
5a:    c9
```

Call 的偏移地址计算 call 完之后 pc 是来到了 5a 地址，`00000000 <pHbfHxVh>:` 特殊函数啊地址 00,5a-00=5a

5a 取补码是 ffffffa6，最后加上 leave 和 ret

所以最后我们写入的代码应该如下

```
8d 98 50 e7 ff ff      lea    -0x18b0(%eax),%ebx
53                     push   %ebx
e8 a6 ff ff ff         call   0 <pHbfHxVh>
c9                     leave
c3                     ret
```

用 hexedit 编辑 phase2.o

```
0x080 8B5D FCC9 C355 89E5 E8FC FFFF FF05 0100  ?]???U????   ...
0x090 0000 8D98 50E7 FFFF 53E8 A6FF FFFF C9C3  ..??P?  S??   ??
0x0A0 9090 9090 9090 9090 9090 9090 9090 9090  ????????????????
```

链接编译运行得到了正确答案。

```
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ gcc -m32 -o linkbomb2 main
.o phase2.o
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$ ./linkbomb2
1170300817
linleo@ubuntu:~/Desktop/sharefile/linklab-1170300817$
```

3.3 阶段 3 的分析

程序运行结果截图：

分析与设计的过程：

3.4 阶段 4 的分析

程序运行结果截图：

分析与设计的过程：

3.5 阶段 5 的分析

程序运行结果截图：

分析与设计的过程：

第 4 章 总结

4.1 请总结本次实验的收获

4.2 请给出对本次实验内容的建议

注：本章为酌情加分项。

参考文献

为完成本次实验你翻阅的书籍与网站等

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/collection/anatmorp.