计算机系统详解课程实验与心得体会

古宜民 PB1700002

2018年6月

第一部分：二进制炸弹实验报告

1. 介绍及准备

二进制炸弹为一个Linux下可执行C语言程序，包含6个阶段，每个阶段要输入特定字符串才能解除炸弹，不同阶段需要不同方法破解，通过反汇编及gdb指令级调试来获得所需内容。

运行环境：Linux 4.9.0-6-amd64 Debian 9, gcc 6.3.0, gdb 7.12.0

建造炸弹：按照README中指示，创建单机bomb，

./makebomb.pl -s ./src -b ./bombs

生成的文件中bomb（可执行）和bomb.c为主文件，其中bomb.c描述了6个阶段分别为phase\_1到phase\_6共6个函数。

1. 逐个阶段破解

**Phase\_1**

gdb加断点到phase\_1并运行，disas反汇编查看代码：

Dump of assembler code for function phase\_1:

=> 0x00005555555551b0 <+0>: sub $0x8,%rsp

0x00005555555551b4 <+4>: lea 0x1555(%rip),%rsi # 0x555555556710

0x00005555555551bb <+11>: callq 0x555555555693 <strings\_not\_equal>

0x00005555555551c0 <+16>: test %eax,%eax

0x00005555555551c2 <+18>: jne 0x5555555551c9 <phase\_1+25>

0x00005555555551c4 <+20>: add $0x8,%rsp

0x00005555555551c8 <+24>: retq

0x00005555555551c9 <+25>: callq 0x55555555579f <explode\_bomb>

0x00005555555551ce <+30>: jmp 0x5555555551c4 <phase\_1+20>

End of assembler dump.

可见以rsi、rip为参数调用了 strings\_not\_equal，之后查看返回值%eax，为0则返回，否则explode\_bomb

进入strings\_not\_equal进一步查看，发现分别直接或间接以%rsi和%rdi为参数调用了 string\_length：

0x0000555555555697 <+4>: mov %rdi,%rbx

0x000055555555569a <+7>: mov %rsi,%rbp

0x000055555555569d <+10>: callq 0x555555555675 <string\_length>

0x00005555555556a2 <+15>: mov %eax,%r12d

0x00005555555556a5 <+18>: mov %rbp,%rdi

0x00005555555556a8 <+21>: callq 0x555555555675 <string\_length>

看汇编知string\_length为求字符串长度的函数。之后比较两个string\_length的返回值：

0x00005555555556b2 <+31>: cmp %eax,%r12d

如果相等则继续，否则返回0x1，进而炸弹爆炸。之后一段代码相对较难理解，但可以看出包含循环和比较，可猜测为两字符串长度相等时依次比较每个字符。由于两个字符串的首地址是函数的参数，只要在gdb中看对应内存值即可。

(gdb) x/1s $rsi

0x555555556710: "Border relations with Canada have never been better."

(gdb) x/1s $rdi

0x555555758780 <input\_strings>: 'a' <repeats 17 times>

(gdb)

其中17个a是输入的字串。

于是phase\_1的答案就是Border relations with Canada have never been better.

**Phase\_2**

第二部分：课程心得体会