

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： 缓冲区溢出攻击**

**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**

通过分析一个程序（称为“缓冲区炸弹”）的构成和运行逻辑，加深对理论课中关于程序的机器级表示、函数调用规则、栈结构等方面知识点的理解，增强反汇编、跟踪、分析、调试等能力，加深对缓冲区溢出攻击原理、方法与防范等方面知识的理解和掌握；

实验环境：Ubuntu，GCC，GDB等

**二、实验内容**

**任务** 缓冲区溢出攻击

**程序运行过程中，需要输入特定的字符串，使得程序达到期望的运行效果。**

对一个可执行程序“bufbomb” 实施一系列缓冲区溢出攻击(buffer overflow attacks)，也就是设法通过造成缓冲区溢出来改变该程序的运行内存映像(例如将专门设计的字节序列插入到栈中特定内存位置)和行为，以实现实验预定的目标。bufbomb 目标程序在运行时使用函数 getbuf读入一个字符串。根据不同的任务，学生生成相应的攻击字符串。

实验中需要针对目标可执行程序bufbomb,分别完成多个难度递增的缓冲区溢出攻击(完成的顺序没有固定要求)。按从易到难的顺序，这些难度级分别命名为smoke (level 0)、fizz (level 1)、bang (level 2)、boom (level 3)和kaboom (level 4)。

**1、第0级 smoke**

正常情况下，getbuf函数运行结束，执行最后的ret指令时，将取出保存于栈帧中的返回（断点）地址并跳转至它继续执行（test函数中调用getbuf处）。要求将返回地址的值改为本级别实验的目标smoke函数的首条指令的地址， getbuf函数返回时，跳转到smoke函数执行，即达到了实验的目标。

**2、第1级 fizz**

要求getbuf函数运行结束后，转到 fizz函数处执行。与smoke的差别是，fizz函数有一个参数。 fizz函数中比较了参数val 与 全局变量cookie的值，只有两者相同（要正确打印val）才能达到目标。

**3、第2级 bang**

要求getbuf函数运行结束后，转到 bang 函数执行，并且让全局变量global\_value 与 cookie相同（要正确打印global\_value）。

**4、第3级 boom**

无感攻击，执行攻击代码后，程序仍然返回到原来的调用函数继续执行，使得调用函数（或者程序用户）感觉不到攻击行为。

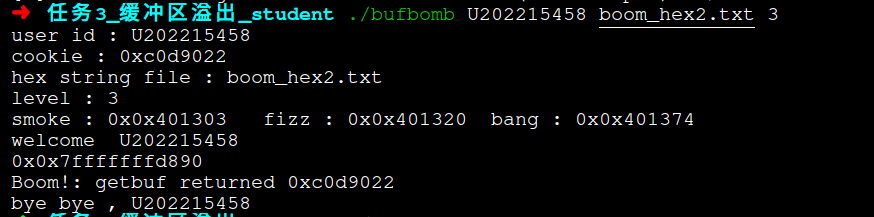
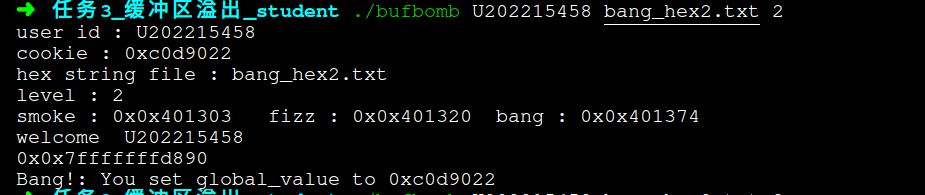
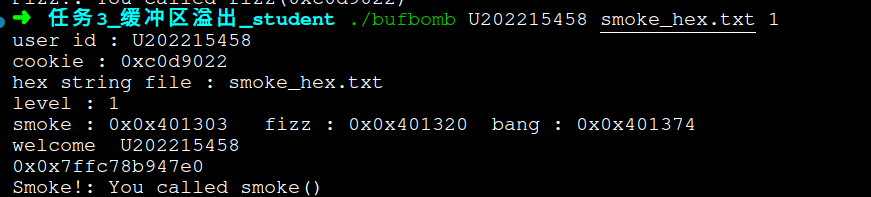
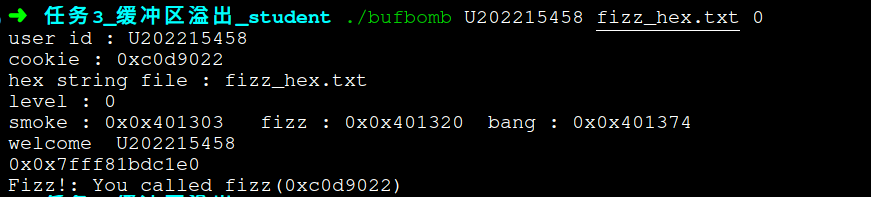
构造攻击字符串，让函数 getbuf将cookie值返回给 test函数，而不是返回值 1 。还原被破坏的栈帧状态，将正确的返回地址压入栈中，并且执行 ret 指令，从而返回到 test函数。

**5、第4级 kaboom**

**一个函数的栈帧的地址通常并不是固定的，随程序运行实例的不同而不同，即每次运行有一个随机的、不固定的值。在此种条件下，要求getbuf（getbufn）函数返回cookie的值，而不是返回值1，并且能正确回到调用函数处继续执行。**

**三、实验记录及问题回答**

**（1）实验任务的实验记录**

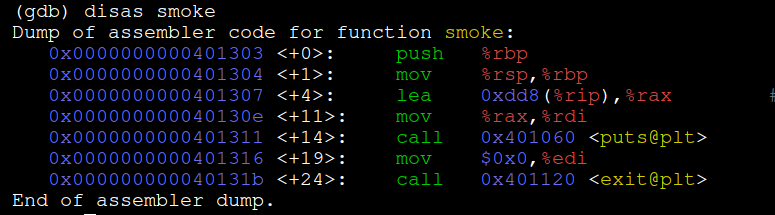
图1 实验通关记录

**（2）缓冲区溢出攻击中字符串产生的方法描述**

先根据自己的学号进行条件编译，对炸弹进行反汇编。

**smoke:**

通过填充缓冲区将缓冲区之外存放返回地址的部分用smoke函数的起始地址覆盖掉。在gdb调试中反汇编smoke函数可以得到其起始地址，如图2.最终用于通关的smoke\_hex.txt内容为00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 13 40 00 00 00 00 00。

图2 反汇编smoke截图

**fizz:**

反汇编fizz函数，如图3,可以知道它将value存在rbp-0x4里面了，且cookie是存在0x4040e8中，最后比较的也是eax寄存器和rbp-0x4,如图4。我们可以通过改变rbp的值，使得rbp-0x4指向cookie存储的位置，即修改rbp为0x4040ec。再将函数返回的断点地址修改为fizz函数中”mov 0x2db7(%rip), %eax”这一条指令的地址0x40132b。即fizz\_hex.txt文件的内容应为：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ec 40 40 00 00 00 00 00 2b 13 40 00 00 00 00 00。通过gdb调试窗口可以看到，rbp的值已经修改成功，且函数成功跳转到fizz函数，如图5。

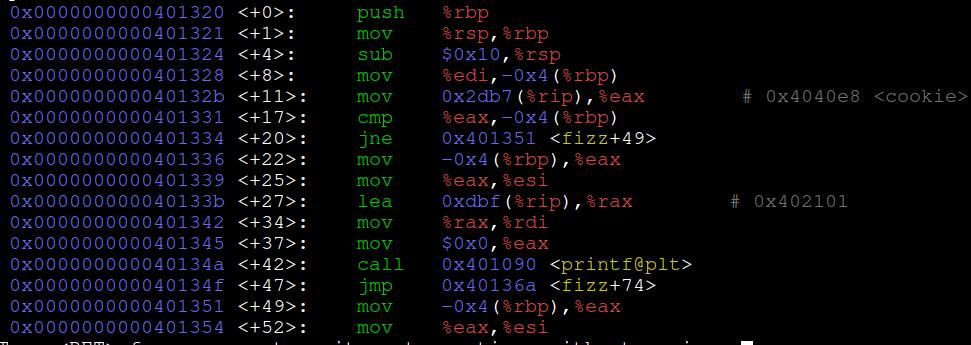


图3 fizz函数反汇编代码



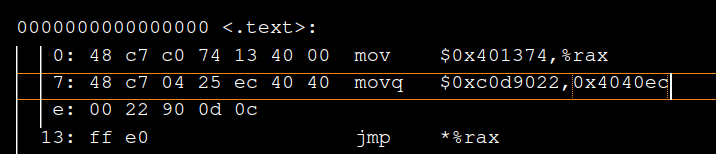
图4 比较指令截图



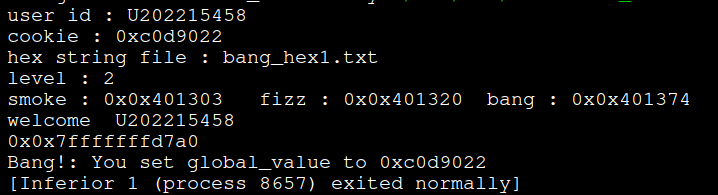
图5 查看rbp寄存器的值截图

**bang:**

本关应该在缓冲区开头写一段攻击代码，该攻击代码完成的主要工作是将value的值改为cookie的值，修改完毕后跳转到bang函数运行。然后将攻击代码转换为二进制文件，写到攻击字符串开头，攻击代码及其对应的二进制代码如图6.

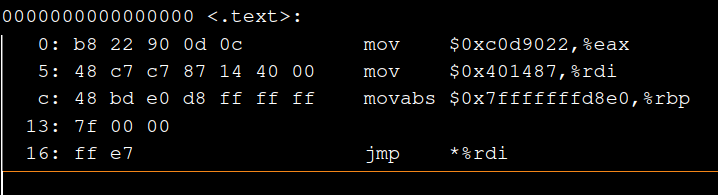
图6 编写攻击代码截图

要想让getbuf结束后跳转到缓冲区开头位置执行代码，应该修改断点地址。注意，即使关闭随机化设置，gdb调试环境和直接运行时的环境是不一样的，需要针对两个环境写两份攻击字符串，区别在于缓冲区的地址，在bufbomb.c文件中加入打印缓冲区地址的语句可以快速确定。gdb调试环境下的攻击字符串如下：48 c7 c0 74 13 40 00 48 c7 04 25 ec 40 40 00 22 90 0d 0c ff e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 a0 d7 ff ff ff 7f 00 00。运行截图如图7.

图7 bang运行截图

**boom:**

本关的大致解决方案如bang,差别在于要将原来的rbp恢复。填写到缓冲区的攻击代码的机器码如图8.其中第三行便是恢复rbp.攻击字符串如下： b8 22 90 0d 0c 48 c7 c7 87 14 40 00 48 bd 80 d8 ff ff ff 7f 00 00 ff e7 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ec 40 40 00 00 00 00 00 10 d8 ff ff ff 7f 00 00。

图8 攻击代码以及机器码截图

**四、体会**

通过对缓冲区炸弹的破解，我深刻领会到了不检查输入长度可能对堆栈产生的巨大影 响，同时熟悉了子程序的结构，和 rip 与堆栈中 rsp、rbp 的联系，对条件编译和传参函数 的调试有了更深入的掌握。