

**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 ： 计卓 2201**

**学 号 ： U202215322**

**姓 名 ： 濮澍**

**指导教师 ： 许向阳**

**2024年 5月 27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。

**三、实验记录及问题回答**

**（1）实验任务的实验记录**

**通过重写Buf越过当前子程序的堆栈区，smoke通过直接修改保存的EIP**

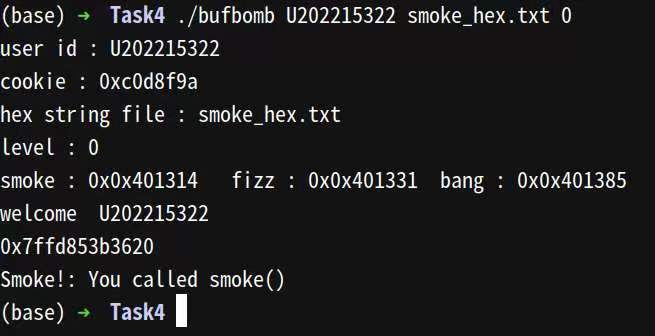
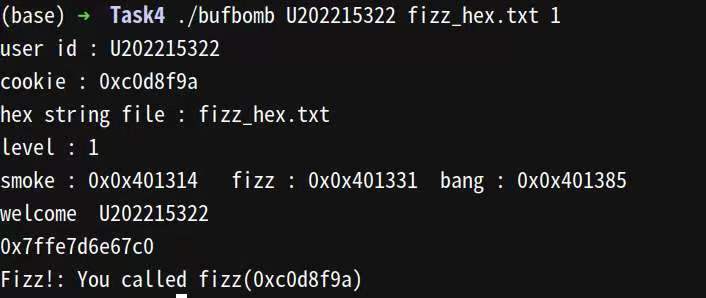


图1 第0级 smoke



**图2 第一级fizz**

**fizz通过修改全局变量值加修改eip为fizz程序地址来攻击**

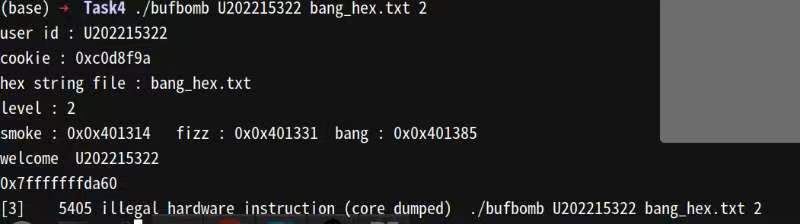


图3 第二级bang

这里重新运行的时候不知道为什么报了个没见过的错误，之前都能完美运行的； 之前遇到了许多 如进入printf错误，gdb运行时和正常运行堆栈段不同等错误，还需通过写两个不同的rbp地址的文件来分别测试

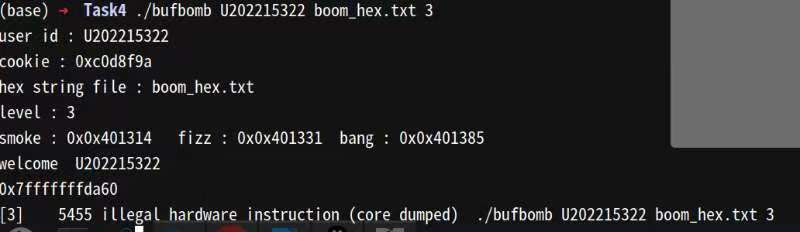


图4 第三级 boom

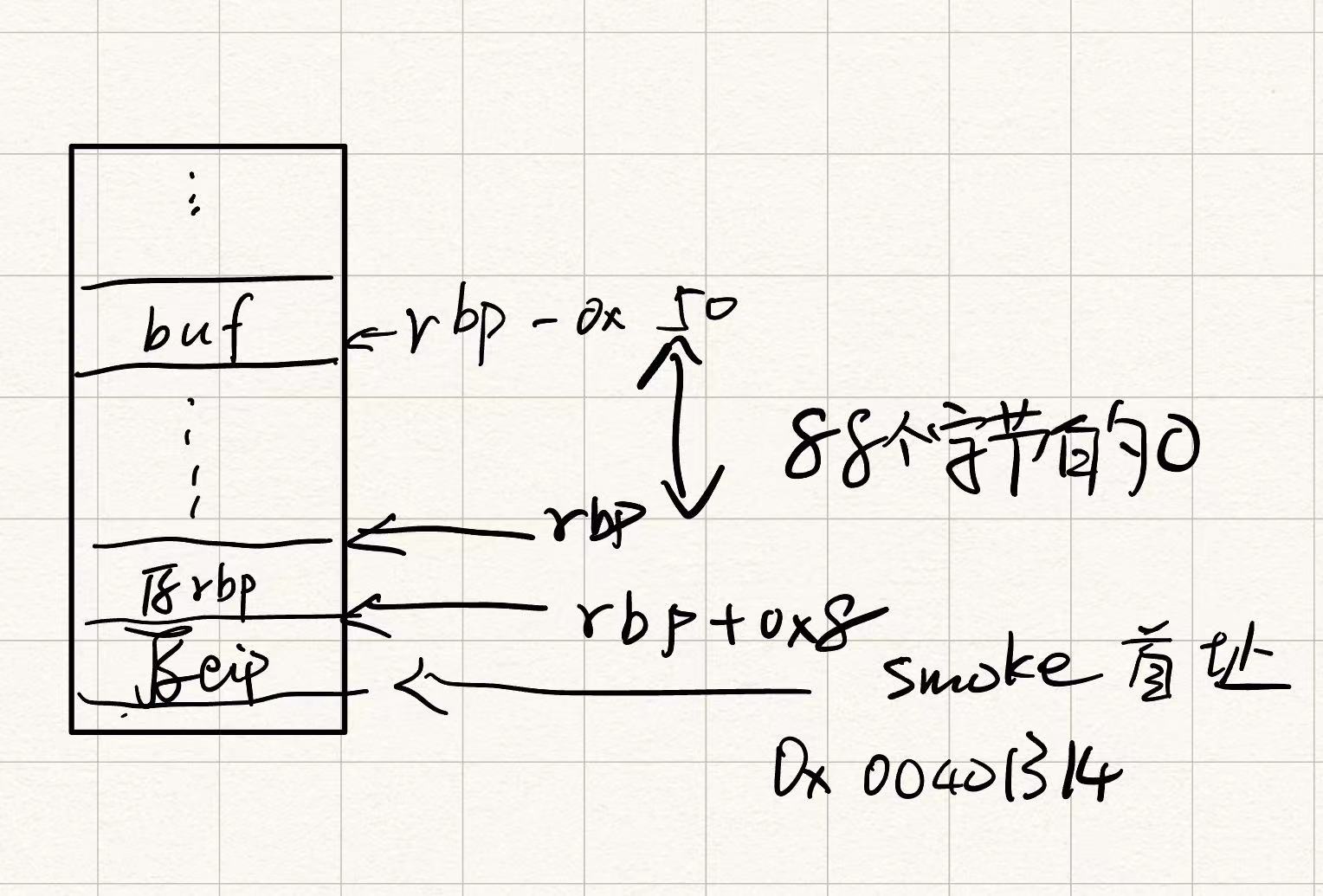
出现了和第二关一样的问题，但我当时做的时候这块是能过的，不知道发生了什么问题

**bang与boom类似 通过获得汇编程序的十六进制代码，对程序代码进行攻击**

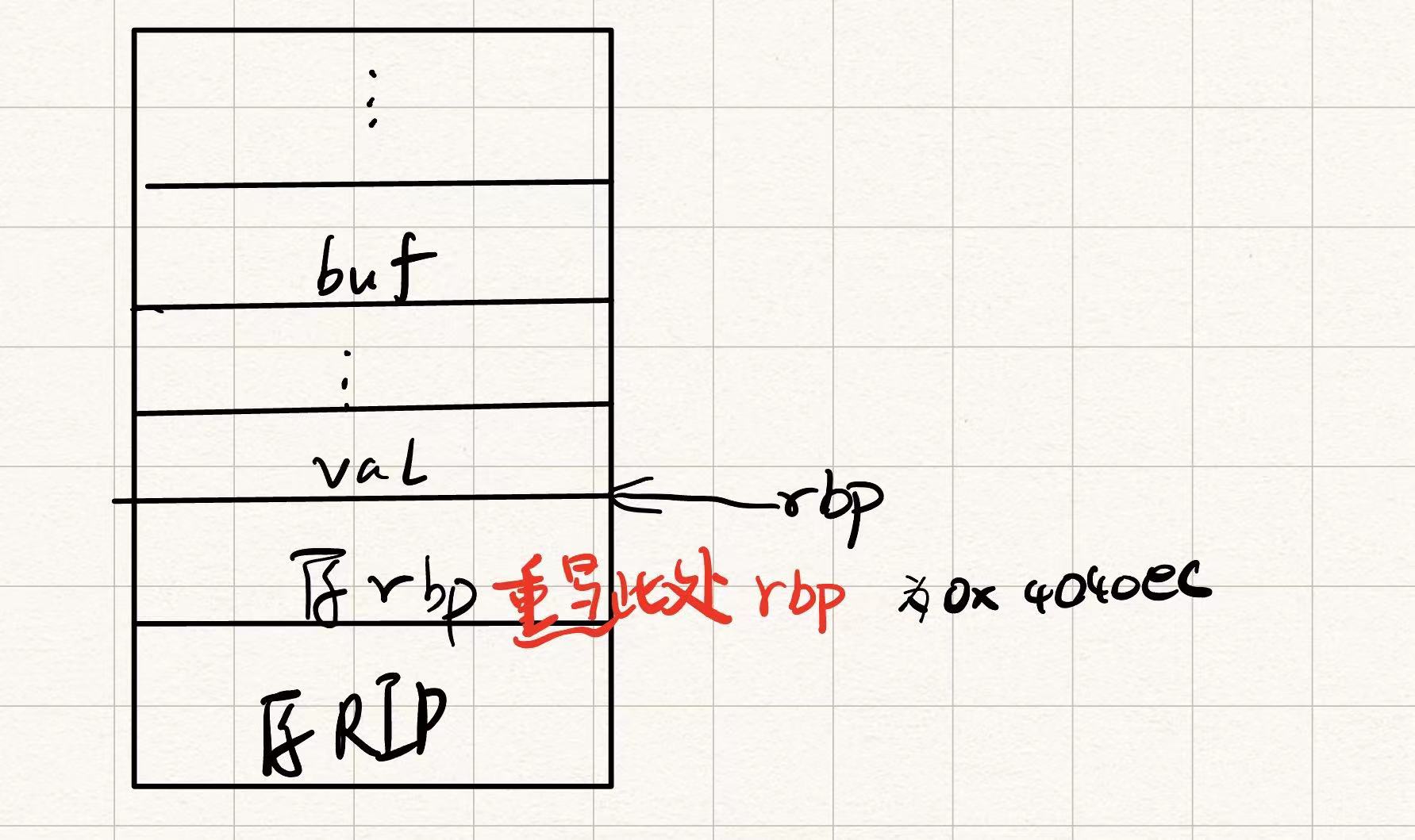
**（2）缓冲区溢出攻击中字符串产生的方法描述**

**要求：一定要画出栈帧结构 （包括断点的存放位置，保存ebp的位置，局部变量的位置等等）**

1. **第0关 smoke**



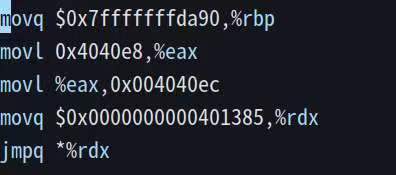
**通过观察运行时寄存器的值，发现buf在栈中距离原ebp的距离为0x50，因此重写buf时，用足够长度00填充，并在原EIP位置上（原ebp＋8bytes）填写smoke函数首地址即可**

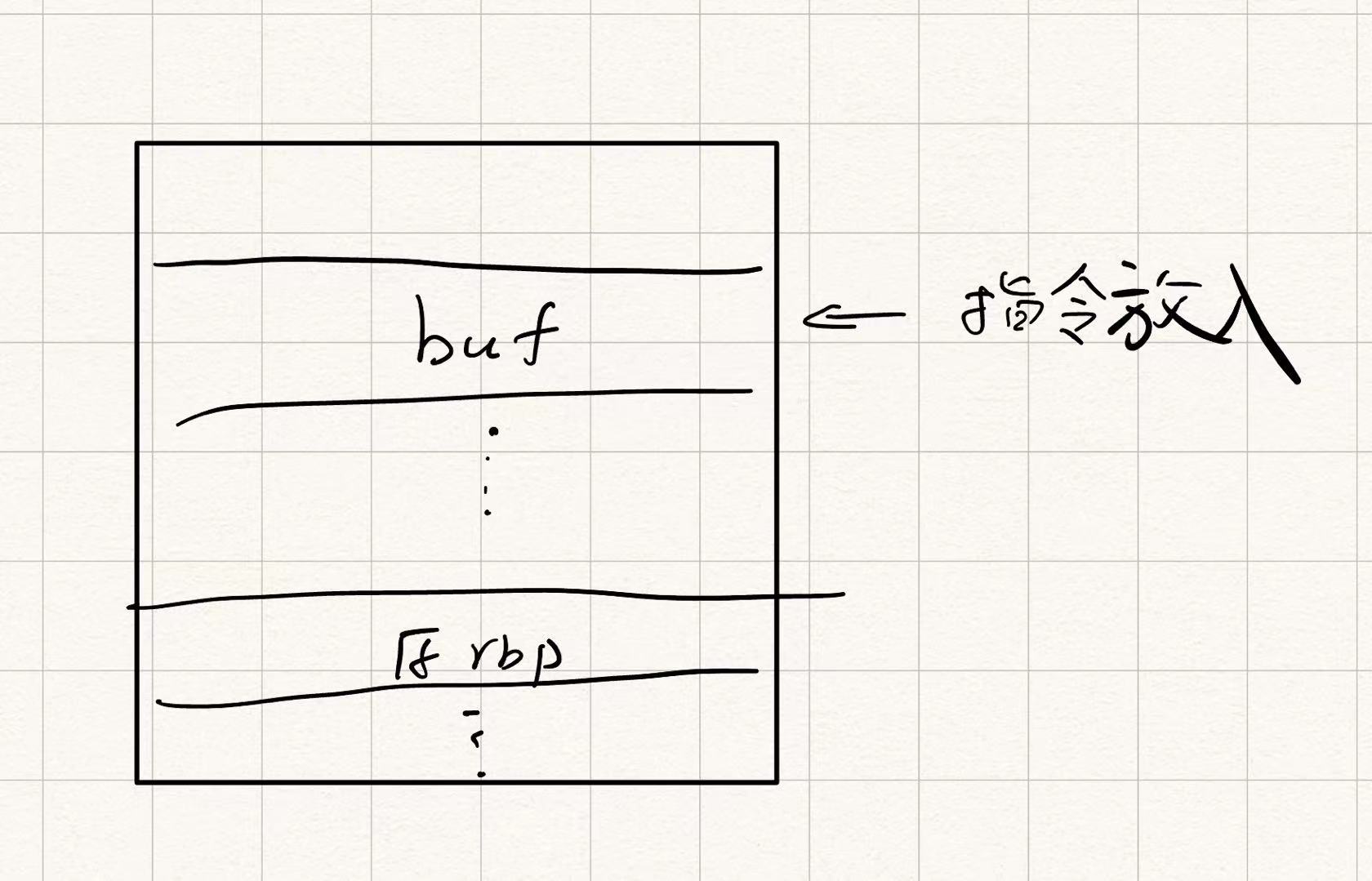
1. **第1关 fizz**

**这关是要通过将val的地址改成和cookie相同，来规避传参工作，即 通过修改rbp地址 使得对应val的地址和cookie相同。 cookie : 0x4040e8; val : rbp-0x4 所以原rbp**

**改为4040ec0x**

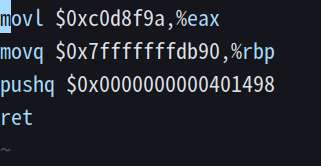
1. **第2关 bang**

**这关需要自己写汇编代码并转化为机器代码，使得可以更改全局变量cookie的值，并通过jmp跳转到对应bang函数地址中去**

第一句是恢复rbp为test原本的rbp地址，避免环境出错，第二句将cookie的地址传入eax,并在第三句通过寄存器寻址修改global\_value的值。 第四句和第五句联合实现了跳转到bang.s的功能

1. **第3关 boom**

**写汇编程序1. 恢复rbp值 2. 修改eax中值为cookie值 3.通过ret跳转到boom.s**



以上程序实现了上述三个功能

1. **体会**

**由于系统环境不同，可参考的东西不够，在debug的时候变得极为复杂，很多时候不知道为什么就报错了。 所以在研究为什么会报seg fault等错误时遇到了很多问题**

**因此在解决这些问题的过程中，增强了我对堆栈段的变化的理解，尤其是rbp和rsp的跳转问题。对电脑安全的防护的意识加深。**

**学习了简单的汇编代码怎么写**