

**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 ： 本硕博202001**

**学 号 ： U202115674**

**姓 名 ： 姚晨炫**

**指导教师 ： 许向阳**

**2022 年 10 月 31 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6.1** 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。

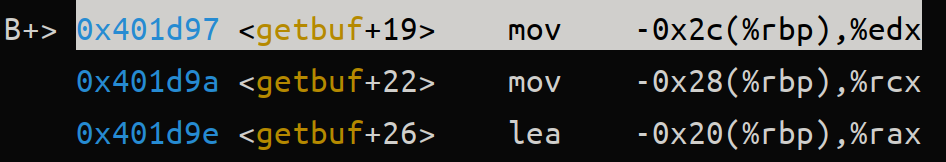
构造攻击字符串，让函数 getbuf将cookie值返回给 test函数，而不是返回值 1 。还原被破坏的栈帧状态，将正确的返回地址压入栈中，并且执行 ret 指令，从而返回到 test函数。

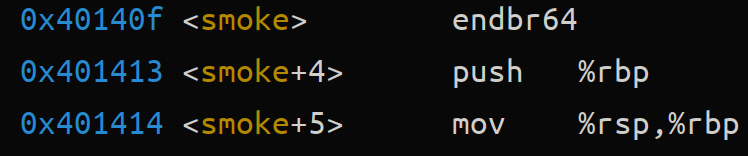
**5、第4级 kaboom**

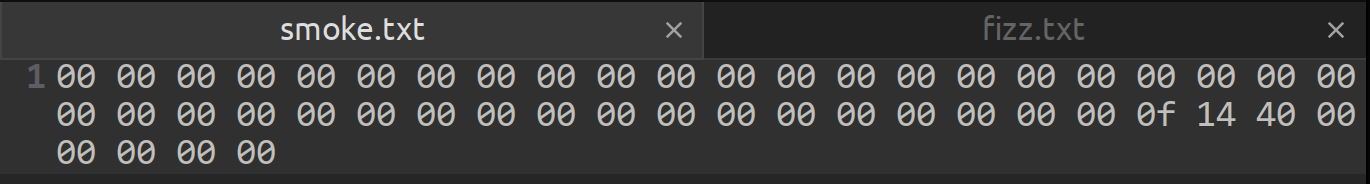
**一个函数的栈帧的地址通常并不是固定的，随程序运行实例的不同而不同，即每次运行有一个随机的、不固定的值。在此种条件下，要求getbuf（getbufn）函数返回cookie的值，而不是返回值1，并且能正确回到调用函数处继续执行。**

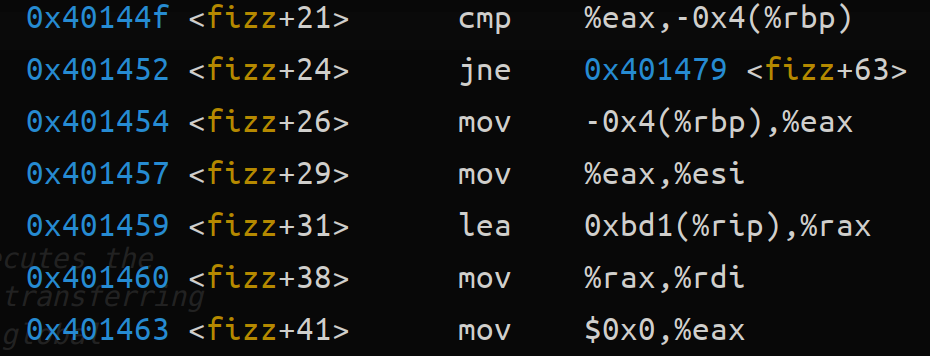
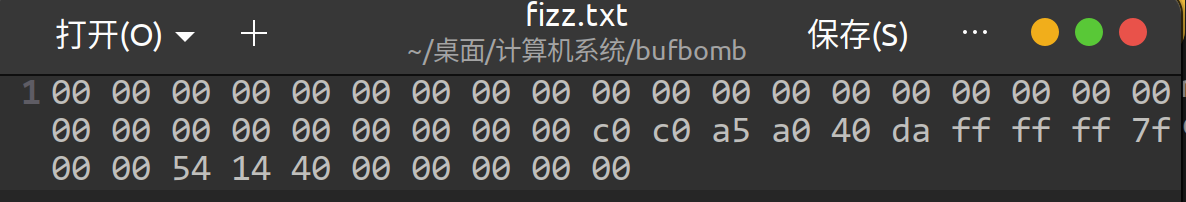
**三、实验记录及问题回答**

**（1）实验任务 6.1 的实验记录**

**1.在getbuf时**

**开辟了一个大小为32的数组，然后程序调用了Get，里面是一个memcpy将用户输入的文件中的内容拷贝到数组下，程序栈底的ebp push进去的时候占用了8个字节，所以在rbp的地址上面存放的是断点地址，在ret之后会跳转到对应的断点处因此我们在前40个字节中随意填入数字，在之后的八个字节中填入smoke函数的指令地址，在ret之后就可以跳转到smoke函数，但是要注意memcpy时候的是将内容直接拷贝，所以到从高位到低位倒过来写。**填入0f 14 40 。对应的密码如图所示。



**2.通过观察fizz函数，我们发现**fizz函数在cmp语句进行之后将-0x4(%rbp)的值重新给了eax用于调用之后的printf函数，所以在这个题目中我们其实大可以绕过传递rdi参数，让程序直接跳转到cmp和fne之后，将cookie直接放到栈的rbp-0x4对应的内存，在之后可以直接被程序调用。在编写代码的同时，需要让保持原来的rbp供程序恢复，所以在原来压入rbp的位置填入原来的值。最后的对应密码为

**四、体会**

**这次实验让我对与栈的结构有了更深的理解，使用memcpy函数修改栈的内容，从而改变函数的走向和返回地址，这让我意识到假如程序不对缓冲区有所保护，会很容易收到外来的攻击。我也学到了如何在机器码的维度进行内存相关内容的写入与识别，让我对机器指令有了更深度的理解。**