Level0

test函数调用getbuf函数，调用完以后返回test函数，leve0要求我们调用getbuf函数后，通过输入我们的level0.txt，使得调用完以后不返回test函数了而是执行smoke函数。

实现：输入超出缓冲区大小的字符串来覆盖到getbuf的返回地址那块地方，将返回地址用smoke函数的入口地址来覆盖

Level1

和level0类似，让bufbomb在test中执行后返回到fizz处执行fizz函数

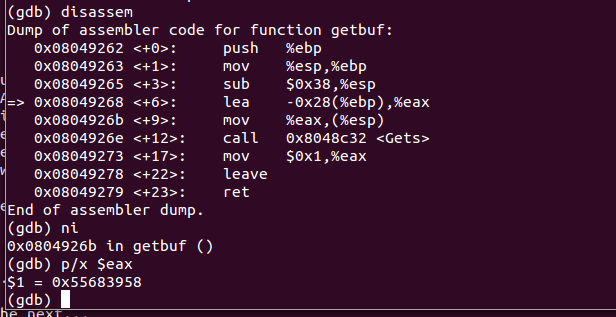
Level2

要执行栈中你指定的特殊指令,设置一个全局变量变你的cookie值,再去执行bang函数.

与之前的区别是bang要传入参数，但是参数是是一个全局变量，这里就是不同于level1那样在栈上覆盖了，因为全局变量要去内存地址中修改，没有可以直接去调用修改的函数，故需要我们自己首先确定好全局变量的地址，自己编写汇编去修改全部变量





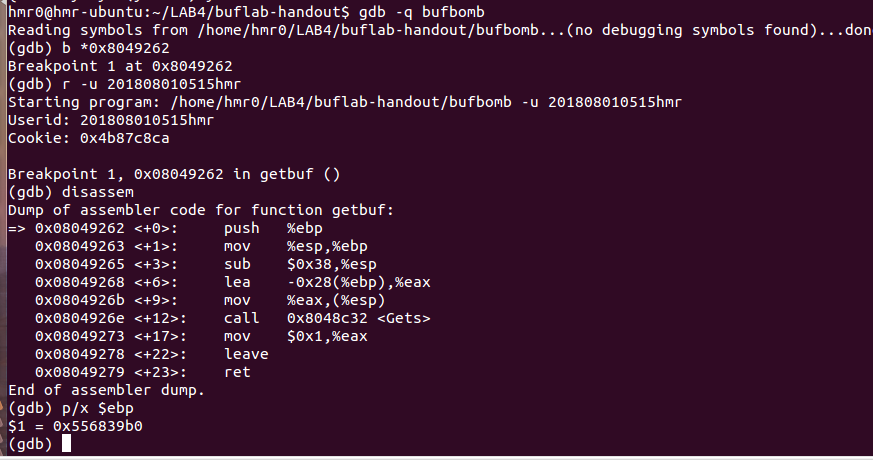


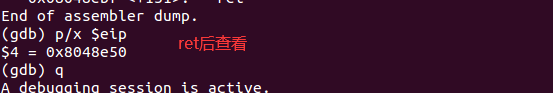
Level3

要修改getbuf()函数的返回值(正常状态为0x1)为你的cookie值,然后让函数正常返回到test

.







Level4

PS ：c 表示继续执行被调试程序，直至下一个断点或程序结束

level中每次运行testn的ebp都不同，所以每次getbufn里面保存的test的ebp也是随机的，但是栈顶的esp是不变的，我们就要找到每次随机的ebp与esp之间的关系来恢复ebp。我们先通过调试来看一下getbufn里面保存的ebp的值的随机范围为多少

ebp buf存储位置%ebp-0x208 Test中%ebp值0x55683980 0x55683778 0x556839b0

0x556839f0 0x556837e8

0x556839c0 0x556837b8

0x556839f0 0x556837e8

0x556839e0 0x556837D8

Test中%ebp值

0x556839b0

0x55683a20

0x556839f0

0x55683a20

0x55683a10

发现getbufn的ebp的值比testn的ebp的值小0x30

在getsbufn执行ret后%esp-0x8的值就是getbufn的%ebp值.

则此时test的值应该为%esp+0x28.