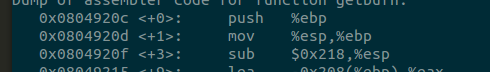
Attack lab 实验记录与学习：  
通过本次Attack lab实验记录一下函数调用过程，底层的堆栈是如何变化的

首先先看一下函数调用的时候会出现的汇编代码





Call 相当于 push %eip jump(所调用的函数)

然后跳到被调用的函数中

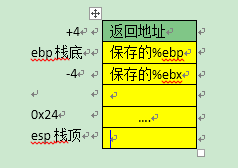
Push %ebp : 把原来函数中的旧的%ebp压入栈中

Mov %esp,%ebp

Sub $0x218,%esp : 将%esp减去0x218个空间，即为被调用函数在栈中开辟了218个空间

下面是图解一下栈在此过程的变化：  
资料源自：<http://blog.csdn.net/shiyuqing1207/article/details/46290521>

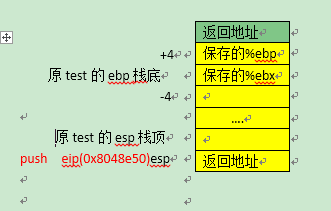
1.以buflab中的函数test中调用getbuf为例



2.test执行到call指令：  
push eip

Jmp getbuf

此时返回地址入栈，此时栈为



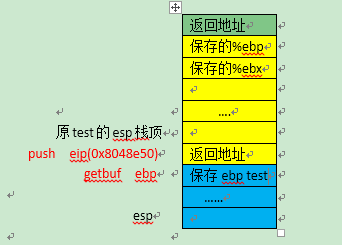
接下来蓝色表示getbuf的栈

Push %ebp 保存旧的ebp的值

Mov %esp,%ebp 使栈指针ebp指向当前的esp处

Sub $0x38,%esp 一共为getbuf开辟了38个空间

%ebp为一个函数的起始栈地址（栈底），%esp为一个函数的栈顶

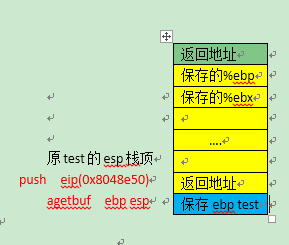


二．接下来说明函数在跳出的时候所执行的代码



函数之间的返回值通常用%eax进行传递

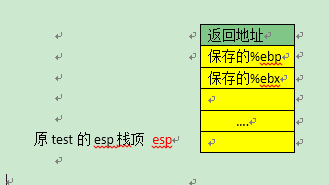
在执行leave的第一步：  
mov %ebp,%esp #将ebp的值给esp,也就是把esp指向当前的栈底，讲开辟的蓝色空间收回



2.leave的第二步：  
pop %ebp #弹出旧的ebp,也就是把调用test函数的函数的栈的ebp弹出恢复。也没有了，

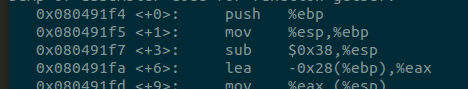
为了getbuf开的空间也完全释放了。

3.ret弹出返回地址以后：



Ret相当于：push %eip

接下来说明一下attackbomb实验：  
本实验通过一个getbuf中没有限定输入字符的长度，所以可以通过输入足够长的字符串来实现把栈中的函数返回地址覆盖进行重写代码，使得程序不按原来程序进行的方式执行。



由push %ebp 和 lea -0x28(%ebp),%eax可知

函数总共要接受输入字符为0x28+ebp,即0x2c，即44，则44的后四位表示的就是

返回地址，我们要做的就是把后四位的返回地址换成自己的返回地址

Level-0:

实验要求我们跳转到smoke这个函数

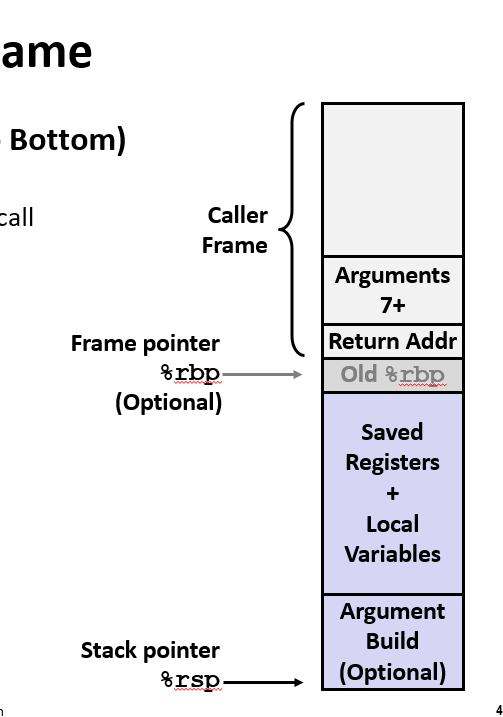
则就把smoke函数的起始地址放到44位之后（注意大端序和小端序）

Level-1:  
实验要求我们跳转到FIZZ函数

并且要求我们输入参数（我们的cookie值）

这个直接先把fizz函数的起始地址放在44位之后

然后在后面的7位中把参数填上（注意大端序和小端序）



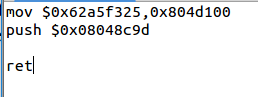
具体的函数参数存储的位置如下所示。

Level-2:  
实验要求我们跳转到bang函数中去，并且要返回我们的cookie值，但是存储cookie的global\_value是全局变量，要求我们要修改这个全局变量

则我们需要自己设计一个exploit 函数，然后返回地址放在我们设计的exploit函数，即把我们自己设计的exploit函数放在输入的buf字符串的开头



即我们要通过gdb跟踪得到%eax即我们的exploit函数的地址，放在44位后面

然后我们需要编写的汇编如下：  


将此汇编代码编译成二进制代码，然后放在答案字节里

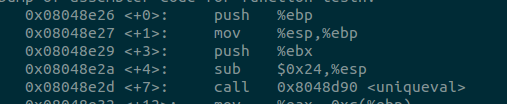
Level-3:  
要求我们改变getbuf的返回值，然后重新跳转到test函数上。

这时候我们需要知道getbuf的返回值储存在%eax上，即要改变%eax的值，然后

还需要改变%ebp的值，把%ebp的值赋值成它原来的旧值，然后再跳回test调用getbuf后的函数。（此题的关键在于要给%ebp赋上旧的值）

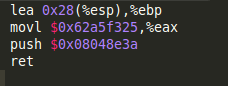
Level-4:  
最后一个level，所有的栈地址都是随机的但需要你输入一段固定的字符串，使得即使地址随机在改变，也要使得每一次攻击成功。这时候就要用上一个关键的指令，也就是nop，nop指令的机器码为0x90,作用是是指令寄存器%eip增加，但不发生任何事，于是我们可以把机器码放在输入的最后，然后前面全为nop

然后我们通过汇编代码：



可以推断出ebp=esp+0x28

接下来我们写的汇编代码就为



将其编译成机器码。

最后添上0x90和返回地址。

以上4个level实验全部完成，

因为本人实在不了解函数调用时堆栈的变化，参考了网上的不少资料才做出来。。。

CMU的实验好难呀。。。。。。。。。。。。。。。。。。。。。。