# **getbuf**

* 分析代码得到栈帧布局

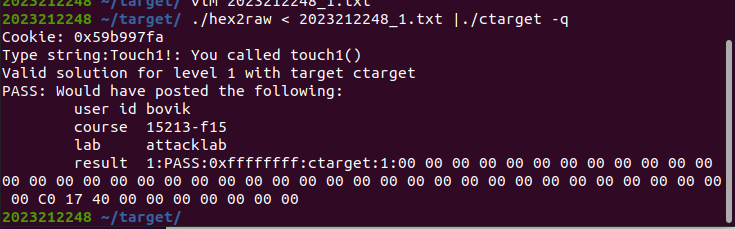
高地址
  
|---------------------|
  
| 上一栈帧的 ebp | ← 被覆盖风险
  
|---------------------|
  
| 返回地址 (retq) | ← 被覆盖风险
  
|---------------------|
  
| 40 字节缓冲区 | ← Gets 函数写入位置
  
| |
  
| |
  
|---------------------| ← rsp 栈顶
  
低地址

* **缓冲区溢出**：Gets 函数不检查输入长度，若输入超过 **40 字节**，会覆盖返回地址，导致程序跳转到攻击者指定的位置执行任意代码。
* 返回地址位于缓冲区后8 字节，因此需要构造：
  + 40 字节填充数据（覆盖缓冲区）。
  + 8 字节目标地址（覆盖返回地址）

# touch1

00000000004017c0 <touch1>:
  
4017d3: e8 e8 f4 ff ff callq 400cc0 <puts@plt> ; 打印成功信息

* 就是只要执行这个函数即可通关，就是覆盖返回地址为touch1的地址



# touch2

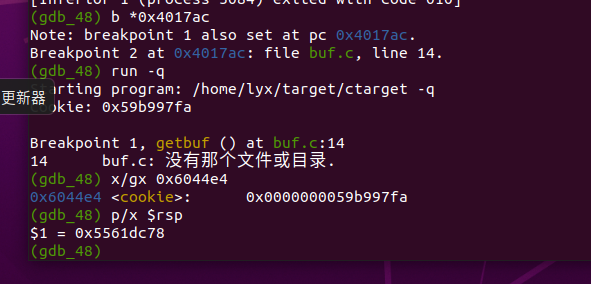
4017fc: 3b 3d e2 2c 20 00 cmp 0x202ce2(%rip),%edi ; 比较 edi 和0x202ce2(%rip)

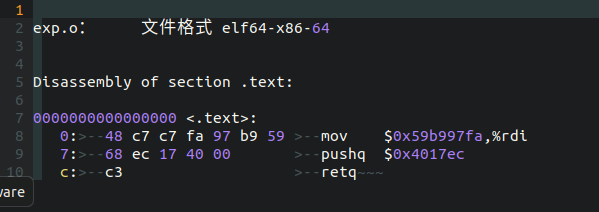
* 计算 %rip = 0x4017fc + 6 = 0x401802 0x401802 + 0x202ce2 = 0x6044e4
* 就是要让edi和他相等

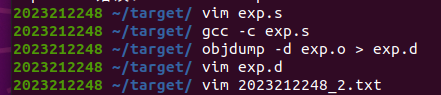
### 思路

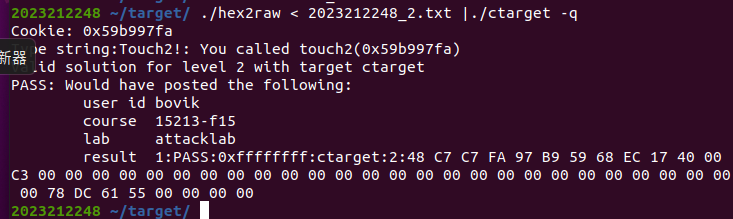
* 断点打到getbuff，查rsp得到getbuf的开始位置，
* 然后修改代码给rdi赋值和跳转到touch2 指令，
* 覆盖返回地址是getbuf的地址让他执行我修改的代码

### 过程和结果

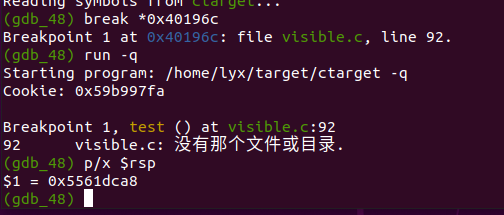




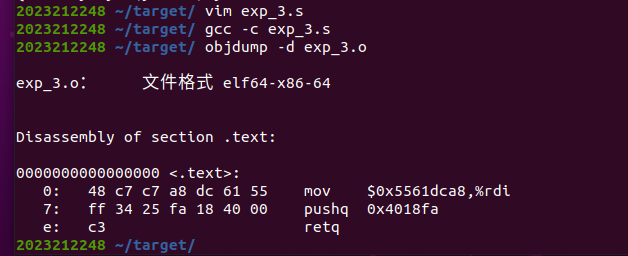




# touch3

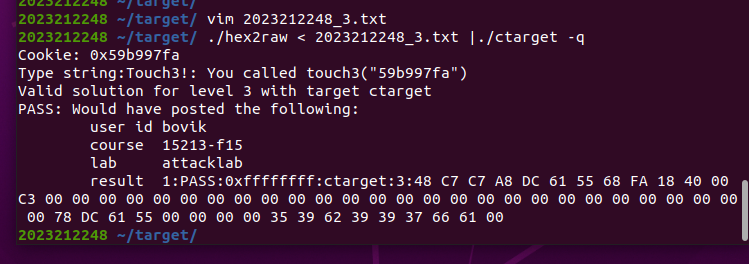
* hexmatch(cookie,sval)==1后才能进入validate(3)  
  而hexmatch函数的作用为将cookie转成字符串并和sval比较，如果相等则返回1，说明除了需进入touch3函数外，\*sval必须等于cookie的字符串形式
* 查看汇编代码
* %rsi（函数第二个参数）为char\* sval，%edi为cookie，而877行将%rdi转入%rsi，说明初始状态下%rdi中存放着char\* sval，即%rdi需要修改  
  所以需要在buffer中注入代码，而为了运行注入的代码，同Phase 2 一样需要跳转回栈顶地址0x5561dc78
* 注意到hexmatch函数中将%r12，%rbp，%rbx入栈，而这样会造成栈中原来输入的内容的覆盖，将数据放在getbuf的栈空间里面，很有可能就被这两个函数cover了。所以要把数据放到一个相对安全的栈空间里，选择放在父帧即test的栈空间里。gdb看一下test栈空间地址
* 

### 构造payload



根据ASCII cookie是 35 39 62 39 39 37 66 61 00

* 48 c7 c7 a8 dc 61 55 68
    
  fa 18 40 00 c3 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
    
  78 dc 61 55 00 00 00 00
    
  35 39 62 39 39 37 66 61
    
  00



# Phase 4

在ROP攻击中设置了栈随机化，所以我们不能像前面三个一样定位到精确地址插入代码

* 将cookie放到%rdi，把touch2的地址放到栈中，以ret执行
* 猜测是需要一个mov命令来放参数，另外一个结合提示就是pop命令了，pop会把栈顶的cookie弹出到另外一个寄存器，再用mov命令写到%rdi里。
* 查询farm可知pop %rax+ret可以用两种gadget表示
* 而pop的内容（0x59b997fa）应该放在pop+retq指令之后，此时pop指令会将pop后对应位置的元素pop进对应的寄存器中
* 而touch2函数地址（0x4017ec）应该放在movq+retq指令之后，当ret指令运行完毕后之后的地址会充当返回地址进入touch2函数

### 构造payload

