姓名：汪俊轩

学号：21307130169

关卡推演过程

phase_1

在phase_1处设置断点，再运行程序，发现在寄存器%rdi和寄存器%rbp内储存了在第一关所输入字符串的字符串的地址，在栈中也有保存，位置为0x20+%rsp，之后，程序调用函数_Z16string_not_equalPcS_，在该处添加断点，进入函数后发现该函数从%rsi与%rdi指向的位置将两者开始逐一比较（每次取出的大小为1byte，与char大小相同），%rdi指向的是我们输入的字符串，%rsi是字符串each line is important，直到遇见0（即字符串结束符）为止。如果每次都相同就将返回值置为1，否则置为0，回到phase_1后如果返回值为1就通过第一关，为0就爆炸。所以第一关的口令是each line is important。

phase_2

在函数phase_2与函数read_six_numbers处设置断点，由函数名称可以初步猜测本关卡需要读入6个数字，观察到函数read_six_numbers的调用了一个int*与一个char*，其中一个的值为0x555555558020，名称为phase_2_numbers，于是猜测后观察代码并通过单步运行验证得到read_six_numbers是将6个数字读入了phase_2_numbers数组中，之后把该数组地址赋给%rax，每次将下一个数字与当前数字乘以-3相比较，逐个比较的分析思路类似phase_1中逐个比较字符串字符的思路，此处不再赘述，相等则继续比较直至最后一个，否则引爆炸弹，由此可以得到第二题是需要输入6个构成等比数列的数，公比是-3，因此可以得到一个答案是1 -3 9 -27 81 -243。

phase_3

设置断点，查看执行__isoc99_sscanf@plt前的%rsi储存的值，输出得到%d %c %d，这是函数scanf的输入格式控制字符串，由此可得第三关要求输入的是数字，字母，数字的组合。在执行了输入之后，读入的值被保存在栈中，函数先判断了读入的数据个数是否为3，不为3则引爆炸弹。将读入的第一个数字与7比较，如果大于7则引爆炸弹，小于7则继续运行，所以我们输入的第一个数需要是一个小于等于7的整数，随后根据输入的第一个数字的不同会跳转到不同的语句去进行接下来的比较，此处我的选择是输入了3，语句跳转到了_Z7phase_3Pc+123处，将输入的第三个数与0x10做了比较，相等则继续运行，不相等则引爆炸弹，因此在输入的第一个数字为3的情况下第三个数字需要是16。之后，给%al赋值0x74（十进制数116，字符t的ASCII码值）并将其与输入的第二个字符相比较，不相等则引爆炸弹，因此，第二个字符应为t。由此可知第三题的其中一个答案是3 t 16。

phase_4

由main函数可知此处的输入是一个64位的long型整数。该整数储存在%rdi中，函数将该整数复制一份到%rax。随后将%rdi算术右移的32位，即将该数字的前32位取出，再减一放置到%edx中，将其与0xd（十进制数13）比较，若大于则引爆炸弹，再将%eax取出（即输入数的后32位），令其自减1后与0xd比较，若小于等于即可跳过炸弹，否则引爆炸弹。之后调用_ZL4hopei函数，此时先观察phase_4后面的部分，发现它将_ZL4hopei函数的返回值与0x1000000作比较，相等就过关否则炸弹爆炸，因此我们进入函数_ZL4hopei之后应让其返回值为0x1000000。进入_ZL4hopei，这是一个递归函数，每次调用时传入的数据都为上次调用的一半，直至0为止会返回1，分析得到除最后一次外该函数每次被调用都会使得最终的返回值左移两位，因此需要使得第一次传入的数据是在右移四次之后才为0，而首次传入的数据就是输入数右移32位的结果，因此输入数的前32位数需要大于等于8。综上所述，满足所有上述条件之一的数为0xc00000001。

phase_5

由main（）函数可知该关卡要求输入3个整数，进入函数体之后，函数phase_5先是将三个子函数的地址保存到了寄存器，分别在栈底0x8，0x18,0x28的位置中，之后判断第一个数字的范围，分小于等于1（保存0x28（%rsp）处的地址到%rdi），大于1小于等于3（保存0x18（%rsp）处的地址到%rdi），大于3（保存0x08（%rsp）处的地址到%rdi），其中如果等于8时将会额外多执行一个语句，将0x79保存到phase_2_nums+24的地方，也就是phase_2函数储存的6个数后面的第七个数的位置，这点在后面进入隐藏关卡时有用。判断了输入的第一个数的范围之后调用_Z13run_lock_testP8baselockii。该函数将%rdi指向的值赋值给了%rax，也就是前面根据输入的第一个数的

范围保存的函数的地址。之后虚函数根据不同的%rax的值执行不同的函数，小于等于1时虚函数为_ZN5lock27acquireEi，大于1小于等于3时虚函数为_ZN5lock17acquireEi，大于3位_ZN5lock27acquireEi。之后在该虚函数内还会调用另一虚函数，按照上方分类顺序依次为function _ZN5lock210is_holdingEi，_ZN5lock110is_holdingEi，_ZN8baselock10is_holdingEi。之后还存在虚函数的调用，此处不再赘述。在这些函数的调用过程中通过比较分别对输入的第二个数字符号和第三个数字的数值进行了限制，三条路线都进行一次之后可以得到三类答案，我第一次选择的是第一个数字小于等于1的路线得到了一组答案为0 0 15，可以通过本关卡，但是要进入隐藏关卡需要选择第一个数字大于3的路径并且要等于8，于是本问最后的口令为8 0 4080。

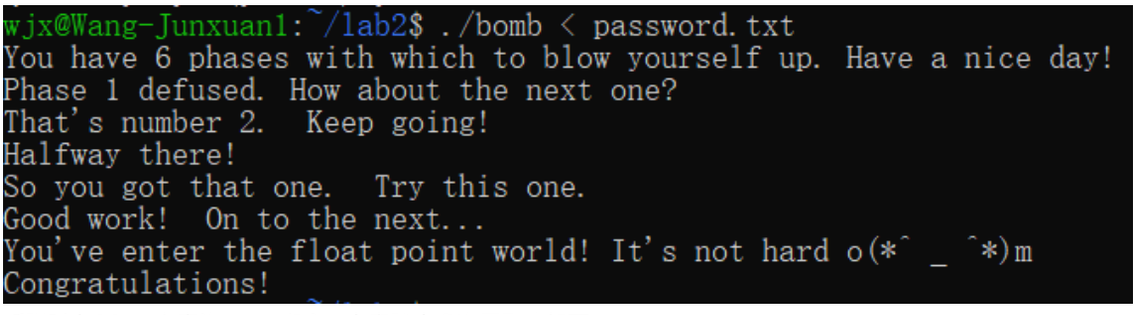
phase_6

进入phase_6之后函数先调用了函数string_len判断输入的字符串的长度。要求长度必须为6。之后把输入的字符串逐个字符连接到(1+2)*(9-0)之后，储存的连接好的字符串名为w2。之后调用函数build_candidate_expression_tree建立待比较的表达式树，在该函数中，待处理的表达式储存在数据结构cand_stack中，数字和符号被分开处理，数字被存储到数据结构digit_stack中，运算符保存在数据结构op_stack中，之后调用函数attach_node建立表达式数，并将结果储存在数据结构cand中，由于不知道cand的数据类型是自定义的tree_node我不知道如何输出于是我只能使用类似x /200s *(cand)的命令查看里面储存的内容，发现里面是每间隔了0xc储存了一个有效内容，函数build_candidate_expression_tree执行完毕之后执行了函数compare_answer_and_candidate，此时看见了数据结构ans我就使用类似输出cand的方法输出了里面的内容，发现其与cand具有相同的数据分布，此时已经可以猜测phase_6的口令是要输入字符使得构造的cand与ans相同，再继续查看函数compare_answer_and_candidate验证猜想，该函数中将cand与ans的内容也是在内存空间中每隔0xc比较一次，不同则引爆炸弹，猜想得以验证，通过查看ans的内容可以得到phase_6的答案为: /(3-4)

secret_phase

进入函数congratulations并查看里面的内容，发现触发隐藏关的条件是在phase_2_nums+24的位置储存了数字121，我现在phase_2里面寻找始终一无所获，因为函数read_six_numbers只会读入6个数字，无法在第七个数字的位置添加值，之后就猜想应该是在其他函数的某处如果满足了某隐藏条件就会直接在目标位置存入值，于是考 把整个bomb反汇编得到bomb.s之后在vscode中打开，使用查找功能查找phase_2_nums+24在哪里出现过,发现在phase_5内出现过，依据之前在phase_5内写的思路分析成功进入了隐藏关卡。进入之后发现涉及了与浮点数有关的汇编代码与寄存器，代码将输入的整数转化为float浮点数之后，将该浮点数乘以2，在加上5.3982412455708344e-315。再转化为double类型浮点数，要求该double类型浮点数小于3820.0000009999999，且大于3819.9999990000001。反推回去最终可以得到一个输入的口令为1905。

拆弹成功的截图



拆弹时在CSDN上查询gdb的指令，在实验中多次用到的如下

打印函数：disas (函数名)

打印内存地址：x \（输出格式） （内存地址）

设置单步运行：set step-mode on

下一步：n

显示目前执行的语句：layout asm

显示寄存器的值：layout reg

意见+建议

无