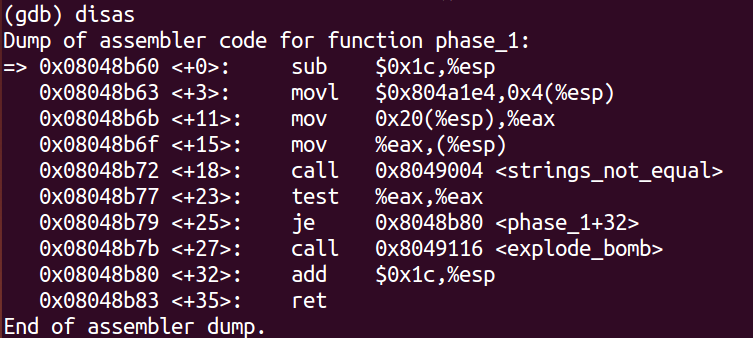
实验三

Phase\_1

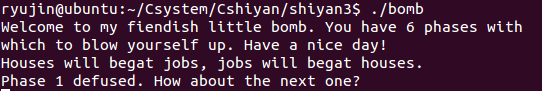
用gdb调试，查看反汇编代码





可以看到，phase1将0x804a1e4处的数据与输入比较，所以密码就是0x804a1e4处的数据。

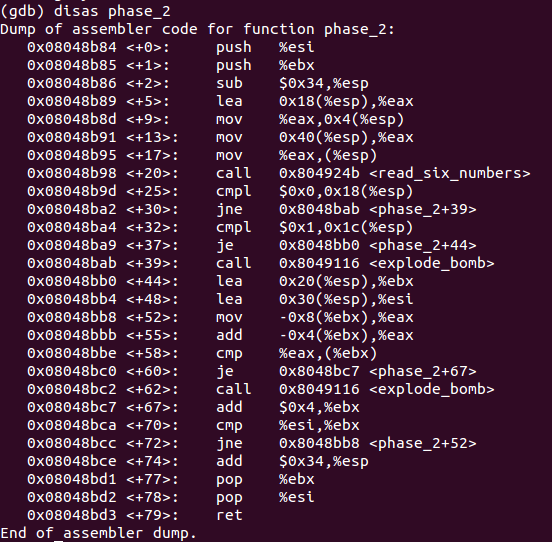
查看得到密码：Houses will begat jobs, jobs will begat houses.



过关。

Phase\_2

反汇编代码



首先将%esi和%ebx寄存器入栈（保存起来），然后在栈上分配0x34字节的空间（即在栈顶减去52个字节）。接着，将栈顶地址加上偏移量0x18存储到%eax寄存器中，并将%eax寄存器中的值存储到%esp寄存器中偏移量为4的位置中。之后，将栈顶地址加上偏移量0x40中的值存储到%eax寄存器中，再将%eax寄存器中的值存储到%esp寄存器中。然后调用函数read\_six\_numbers，读取六个整数并存储到栈顶加上偏移量0x18的位置开始的连续六个位置中。

之后，比较第一个整数（即栈顶加上偏移量0x18处的值）与0的大小关系。如果不等于0，则跳转到地址0x8048bab处，调用函数explode\_bomb，使程序崩溃。否则，比较第二个整数（即栈顶加上偏移量0x1c处的值）是否等于1，如果相等则跳转到地址0x8048bb0处。如果不相等，则同样调用函数explode\_bomb。

接下来，将栈顶地址加上偏移量0x20和0x30分别存储到%ebx和%esi寄存器中。然后，从%ebx指向的位置读取一个整数，并将其加上-0x4(%ebx)指向的整数，将结果存储回%eax寄存器中，即%eax等于前两个数之和。接着，将%eax与(%ebx)指向的整数进行比较，即比较第三个数和前两个数之和，如果相等则跳转到地址0x8048bc7处，否则调用函数explode\_bomb使程序崩溃。

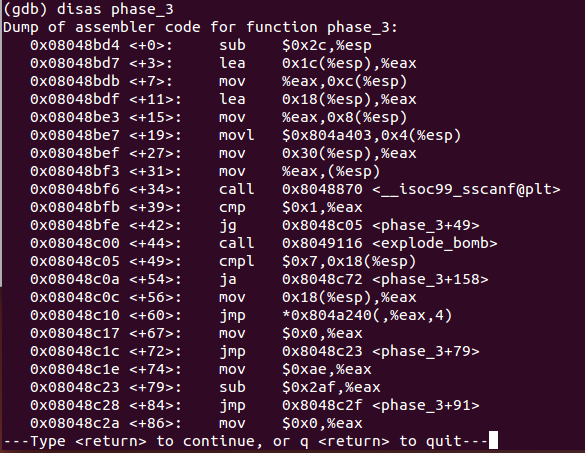
最后，将%ebx寄存器中的值加上4，并再次与%esi寄存器中的值进行比较，如果小于%esi寄存器中的值，则回到地址0x08048bb8继续执行这个循环，否则退出函数。

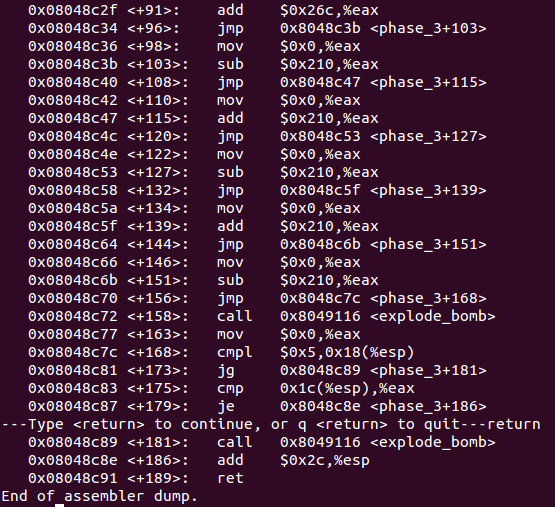
综上，密码为0 1 1 2 3 5



Phase\_3

反汇编代码





1. 将栈指针减去44，以在堆栈上留出空间。
2. 将位于堆栈上0x1c处的地址加载到EAX寄存器中，并将其存储在与当前堆栈位置相距12个字节的位置0xc处。
3. 将位于堆栈上0x18处的地址加载到EAX寄存器中，并将其存储在与当前堆栈位置相距8个字节的位置0x8处。
4. 将常量值0x804a403加载到距当前位置在第四个位置的堆栈上。
5. 将位于堆栈上0x30处的值加载到EAX寄存器中。
6. 将EAX寄存器推入堆栈，并调用\_\_isoc99\_sscanf函数。
7. 将sscanf的返回值与值1进行比较。如果返回值大于1，则跳转到指令+49。否则，调用explode\_bomb函数。
8. 将位于堆栈上0x18处的值与7进行比较。如果该值大于7，则跳转到指令+158。
9. 根据位于堆栈上0x18+处的值，跳转到内存中的七个不同位置之一。
10. 根据跳转到的位置，将常数加、减或移动到EAX寄存器中。
11. 将位于堆栈上0x18处的值与5进行比较。如果该值大于5，则跳转到指令+181。
12. 将EAX寄存器中的值与位于堆栈上0x1c处的值进行比较。如果它们相等，则跳转到指令+186。否则，调用explode\_bomb函数。
13. 将44添加到堆栈指针，并从函数返回。



可以看出输入需两个整数。

利用设断点调试，可以得出第一个数字0-5分别跳转到0x08048c1e, 0x08048c17, 0x08048c2a, 0x08048c36, 0x08048c42, 0x08048c4e。

分别计算eax跳转计算后的值为：-421，-595，92，-528，0，-528。

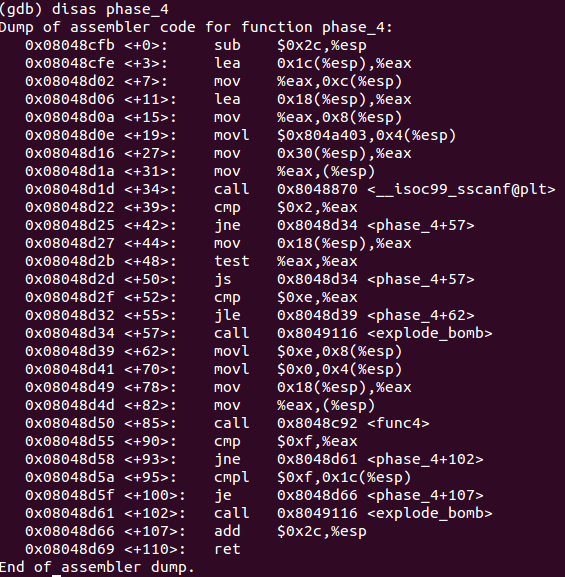
即第二个输入的值。



过关。

Phase\_4

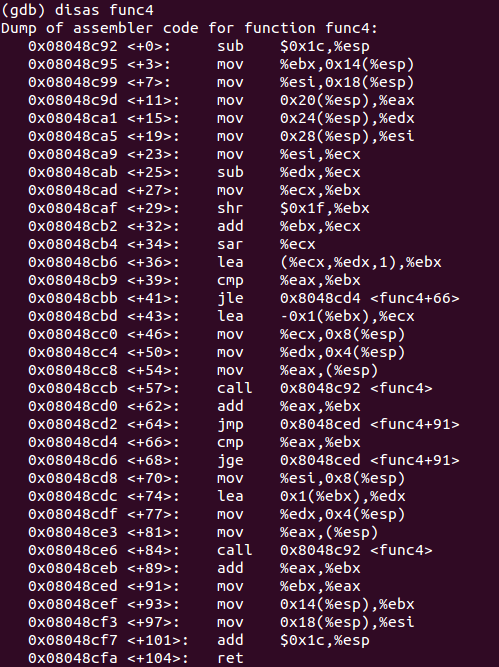
反汇编代码





可见输入为两个整数

Func4函数



func4（）的功能就是递归二分，并将每次二分中间值保留累加，作为返回值，直到二分中间值=传入参数。

题中第一个输入a小于等于14且不等于0；第二个输入b为15；

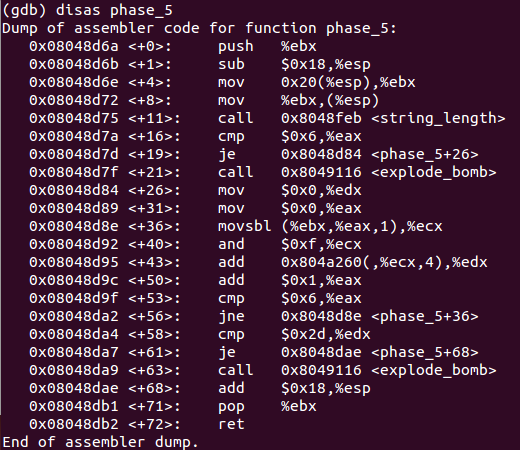
在0-14递归查找a，返回值r为每次查找时的中间值之和。

以5 15为例。第一次r=7，5<7；在0-6继续查找，r=3，5>3；在4-6继续查找，r=5=a。所以最终返回值为7+3+5=15=b。

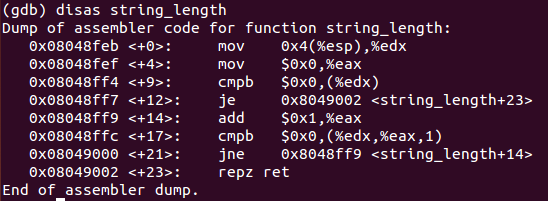


Phase\_5

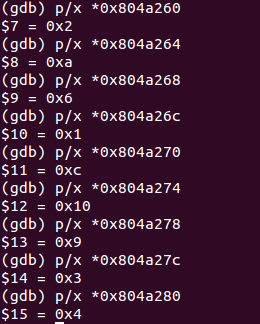
反汇编代码



String\_length函数



String\_length函数计算输入字符串的长度，若长度不为6，炸弹爆炸。



Phase\_5将字符串每一位与0xf相与，得到的结果作为偏置，在0x804a260为首地址的数组中对应加起来，若和与45（0x2d）相等，则过关。

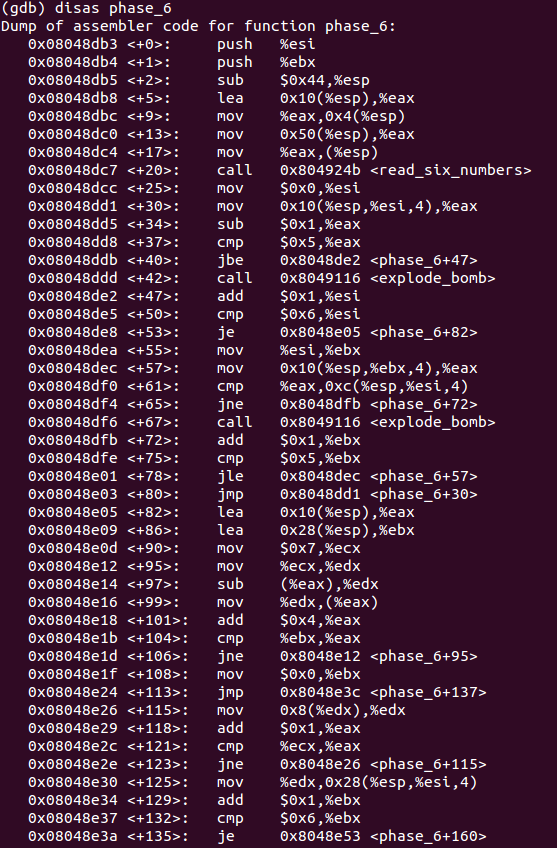
45=0xa+0xa+0xa+0xa+0x3+0x2

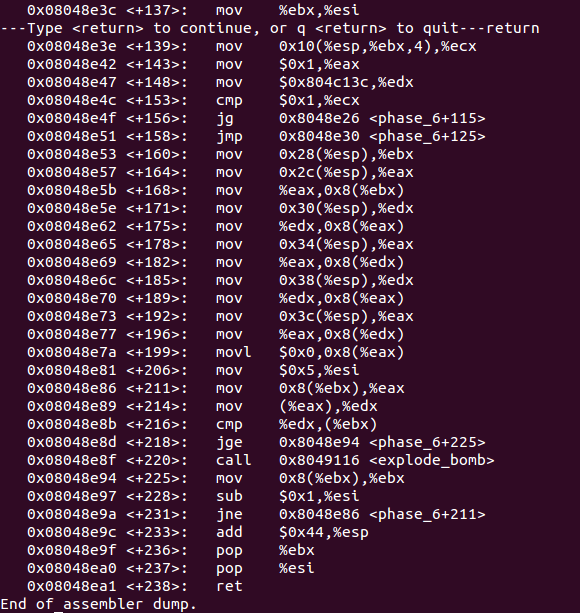
a在ascll码表中为0110 0001（97）与0000 1111（0xf）相与后为1，加上偏置即为0x804a260+1\*4，对应。（答案不唯一）



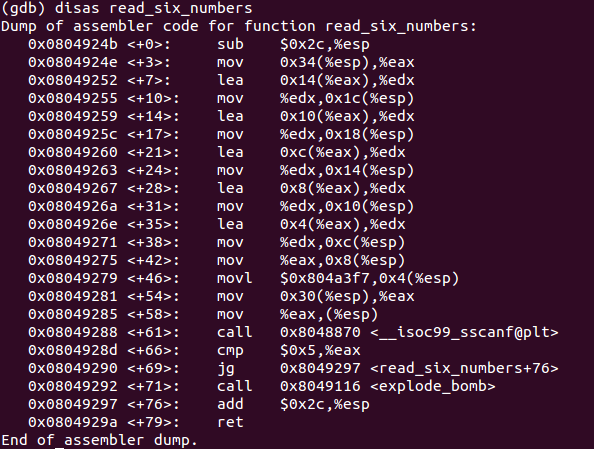
Phase\_6

反汇编代码



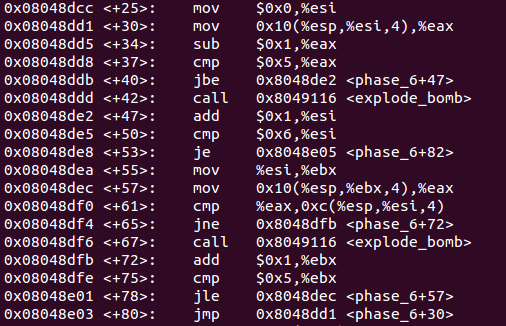


Read\_six\_numbers函数

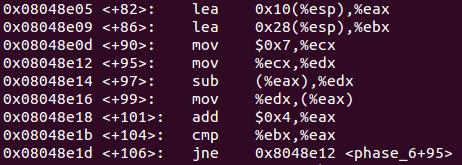




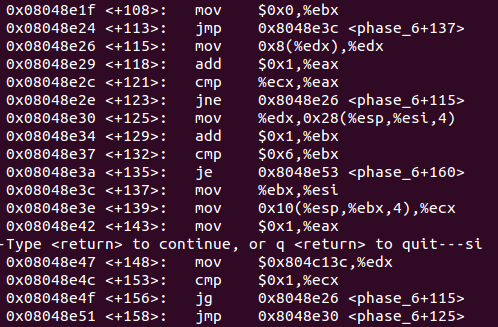
可见要求输入为6个整数。



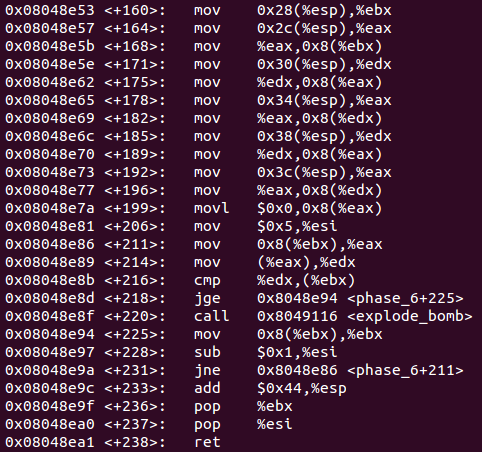
双重循环，esi记录大循环层数，ebx记录小循环层数，每一个数都小于等于6，每一个数与后一个数不等。故这个双重循环判断输入的六个数是否为1,2,3,4,5,6这六个数的排列。



将每个数变换，x=7-x。ebx存第七个数的地址，若eax=ebx，则退出循环。



双重循环，在输入的六个数后加上新的六个数，新的六个数以0x804c13c为首地址。大循环遍历x，小循环对新的六个数加上偏置，例如，x=4且为第一个数，那么第七个数为0x804c13c+24。



判断新的六个数是否为从大到小排序。



查看新的六个数，需要从大到小，则2ce>13d>db>9e>89>43。

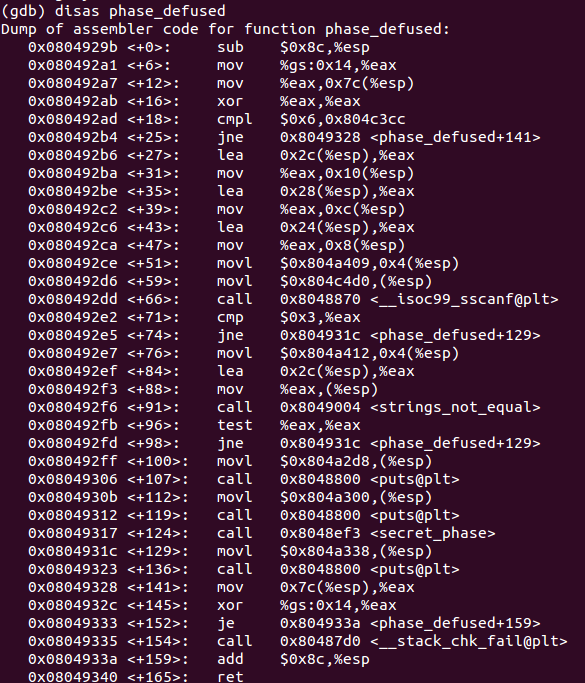
则新x应为6 2 3 1 4 5，则输入为1 5 4 6 3 2。



成功拆除炸弹。

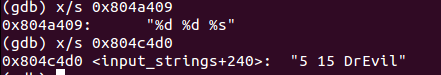
Phase\_defused

反汇编代码



进入隐藏关





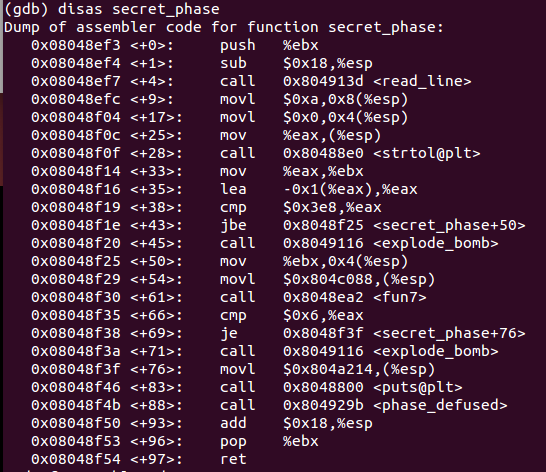
可见隐藏关需在第四关时多输入一个字符串。



字符串应为DrEvil



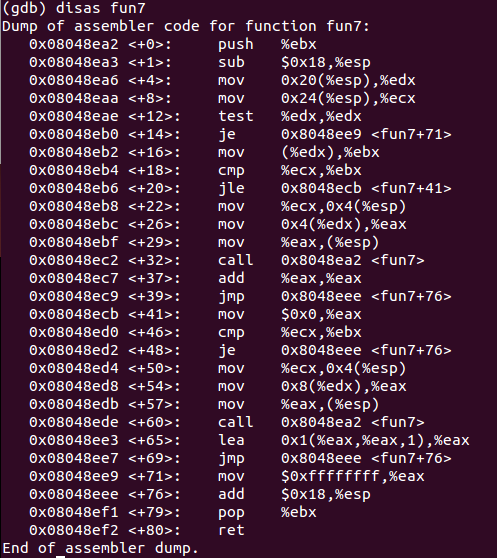
成功进入secret\_phase



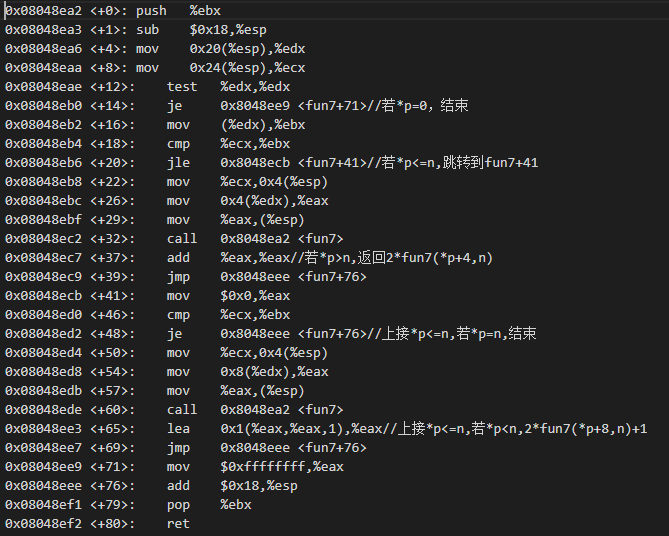
Read\_line函数至strtol@plt函数功能是

将输入的字符串转化为十进制整数，且输入不超过0x3e8，即1000。

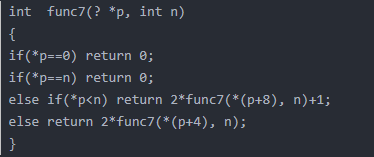
Fun7函数



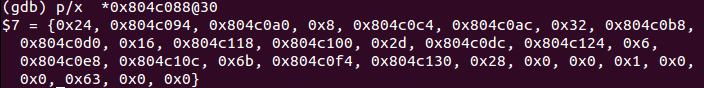
Fun7传入两个参数，\*p和n



可以得出fun7的函数



查看传入\*p的值



推测应该是二叉树，从secret\_phase中看到，需要fun7返回值为6

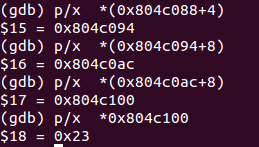


而6=a\*2 => a=3

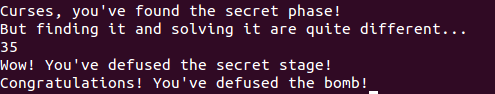
a=3=b\*1+1 => b=1

b=1=c\*1+1 => c=0

故回溯时经过两次+8，一次+4。



答案是0x23，即35。



至此，炸弹全部拆除。