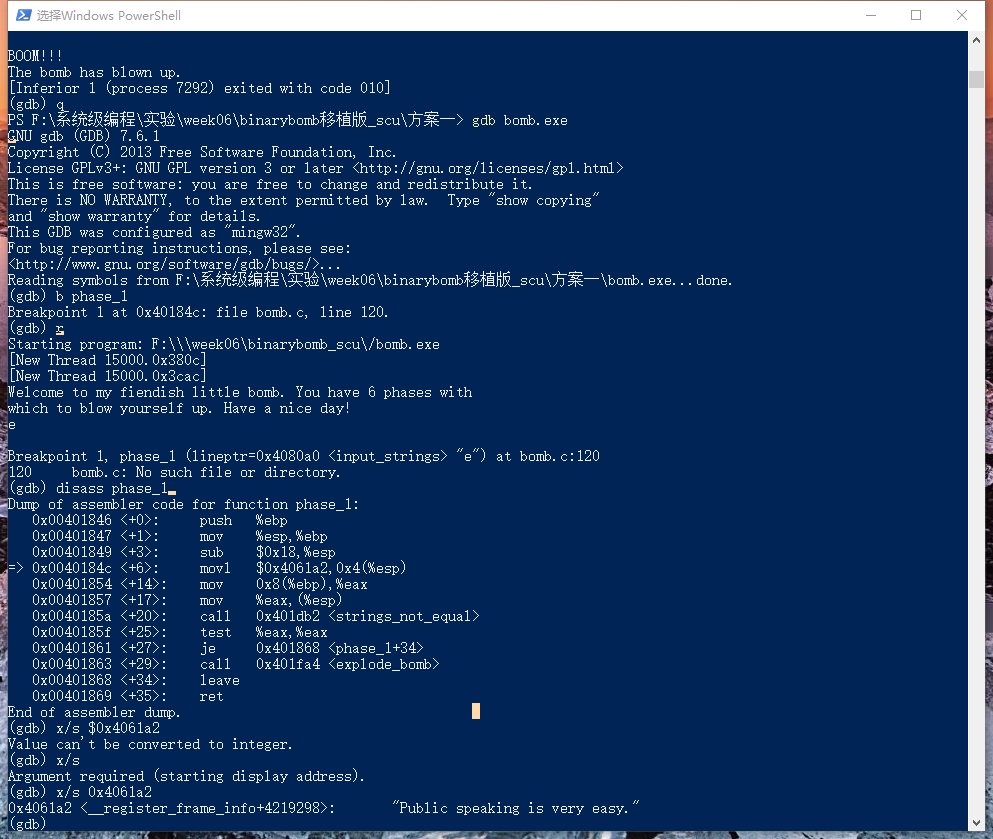
# 二进制炸弹拆解

1. phase\_1:

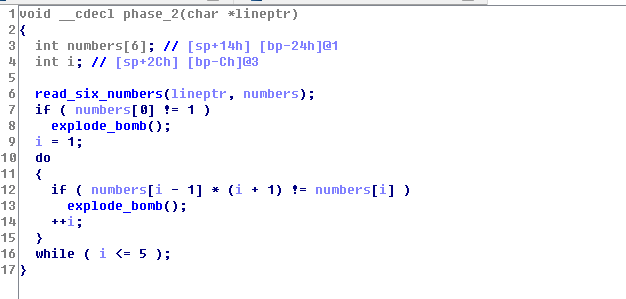
查看phase\_1函数的汇编代码，能够轻易的看出：函数strings\_not\_equal接受两个参数，第一个参数是我们输入的字符串，这个函数做的事就是比较两个参数是否相等，不等则爆炸。所以我们通过调试，查看第二个参数的值，即0x0040184c <+6>: movl $0x4061a2,0x4(%esp)，中位置4061a2的值，输入指令x/s 0x4061a2，以下为操作过程截图：



所以结果为Public speaking is very easy.

1. phase\_2:

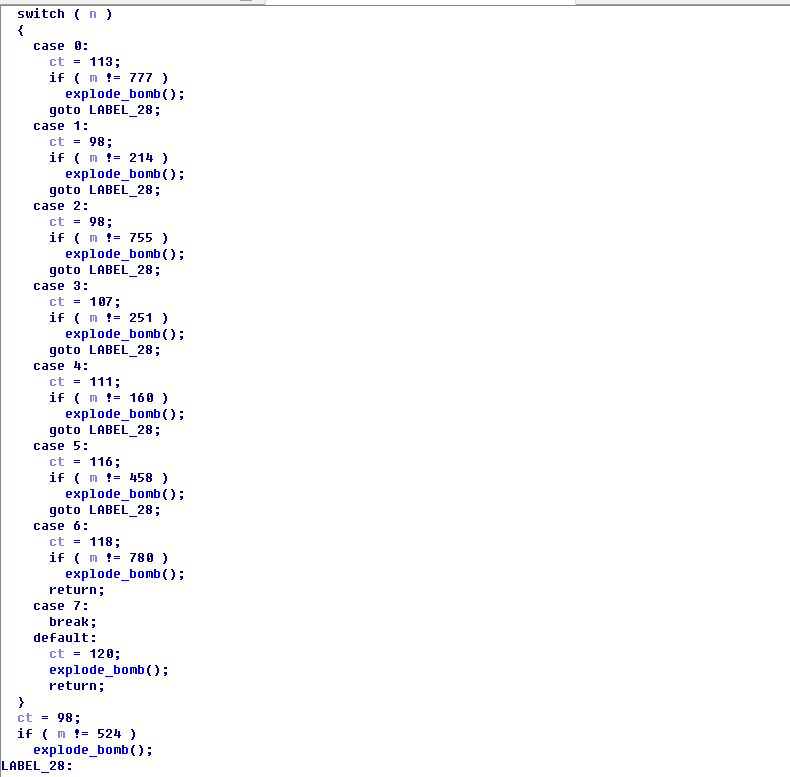
借助ida软件来进行分析，可得出源码如下：



上述可看出，输入结果有六个数字，第一个数字为1，接下来的数字为:numbers[i] = numbers[i - 1] \* (i + 1)，所以结果为1 2 6 24 120 720。

1. phase\_3:

同样借助那个工具来做，很容易得到这个炸弹考查的是switch语句的用法，如下：

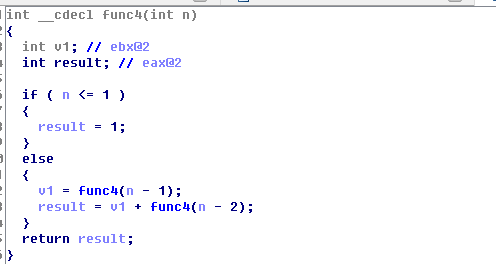




从代码中可看出，要拆解炸弹，只需要输入0 133 777（其他的也行），而133要输入对应的ASCII码，所以结果可以是0 q 777

1. phase\_4:

ida结果如下图：



这是一个递归，实在看不懂代码是在做什么，所以写了一个辅助程序来得出答案，如下：

#include <stdio.h>

int main() {

    for (int i = 0; i < 55; i++) {

        if (func4(i) == 55) {

            printf("%d", i);

            break;

        }

    }

}

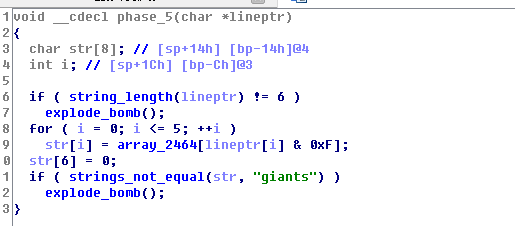
最后结果为9.

1. phase\_5:

首先根据ida的结果找到array\_2464的数据结果：



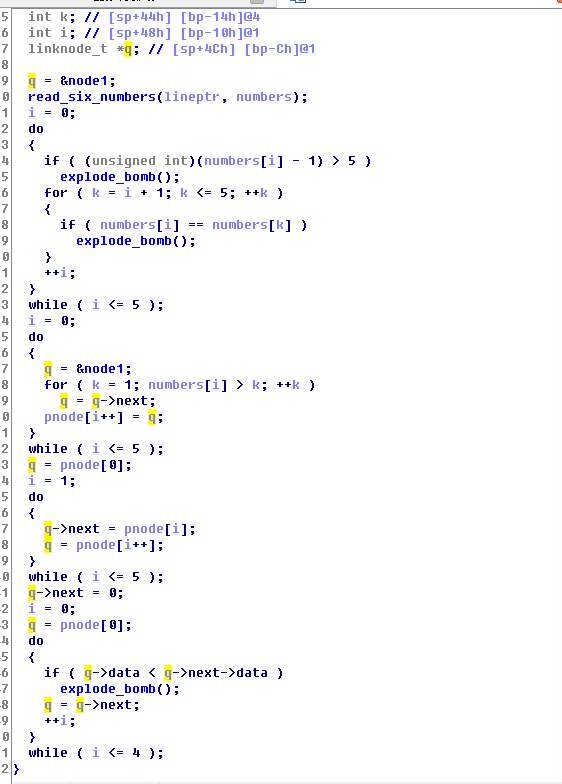
再由ida得到c的代码，如下：



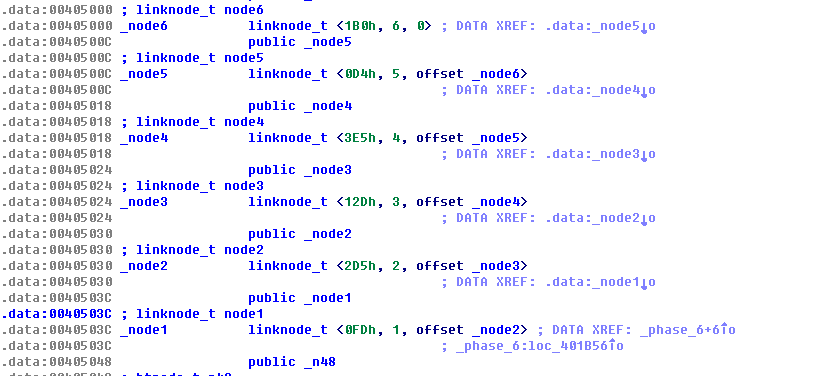
可看出，str的值为giants，对应着array\_2464的数组下标为：15 0 5 11 13 1，取的是字符的后四位，前面四位均为0，而后四位对应的十进制数是那个数组的下标，例如15：就去ASCII码表上找后四位全为1的字符随便取一个即可，故输入的字符串可以是：?PU[]A

1. phase\_6:

ida软件提取出的源代码如下：



从上面代码可看出，是要让一个链表中的值得前一个比后一个大，node的数据如下图：



改变顺序是通过移动numbers[i] – 1个位置赋值给pnode，使得前一个值比后一个值大，最简单的做法就是从大到小排列，所以结果可以是4 2 6 3 1 5

最终六个阶段的运行结果如下图：

