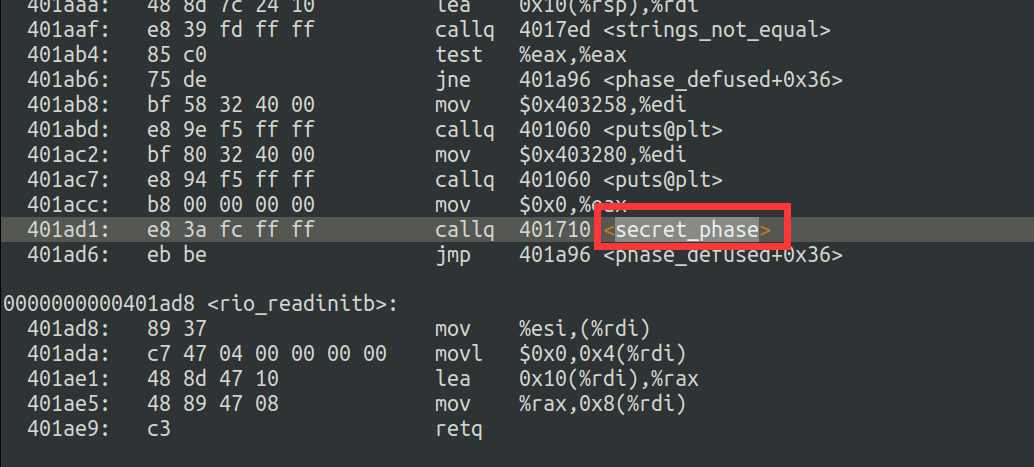
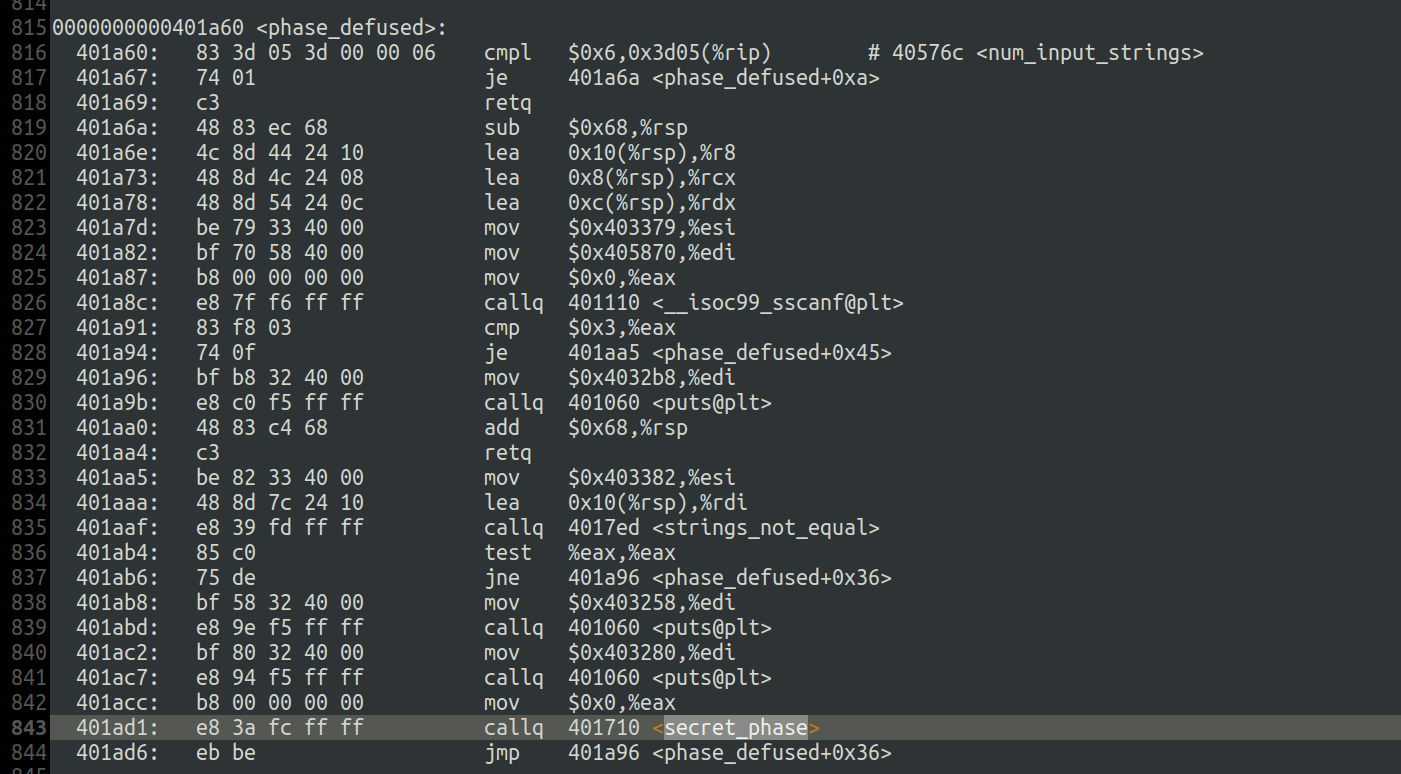
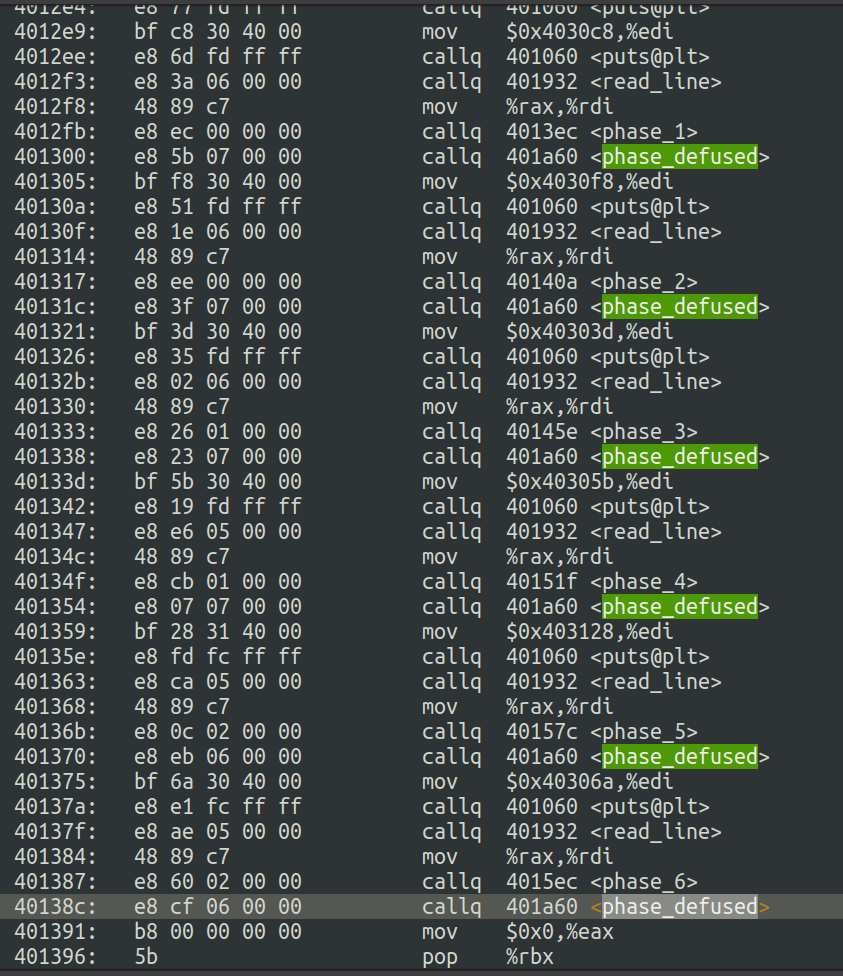
为了发现隐藏关卡的入口，直接利用objdump -d反汇编bomp可执行文件，发现其中存在字样“secret\_phase”,推测这就是隐藏关卡的入口。

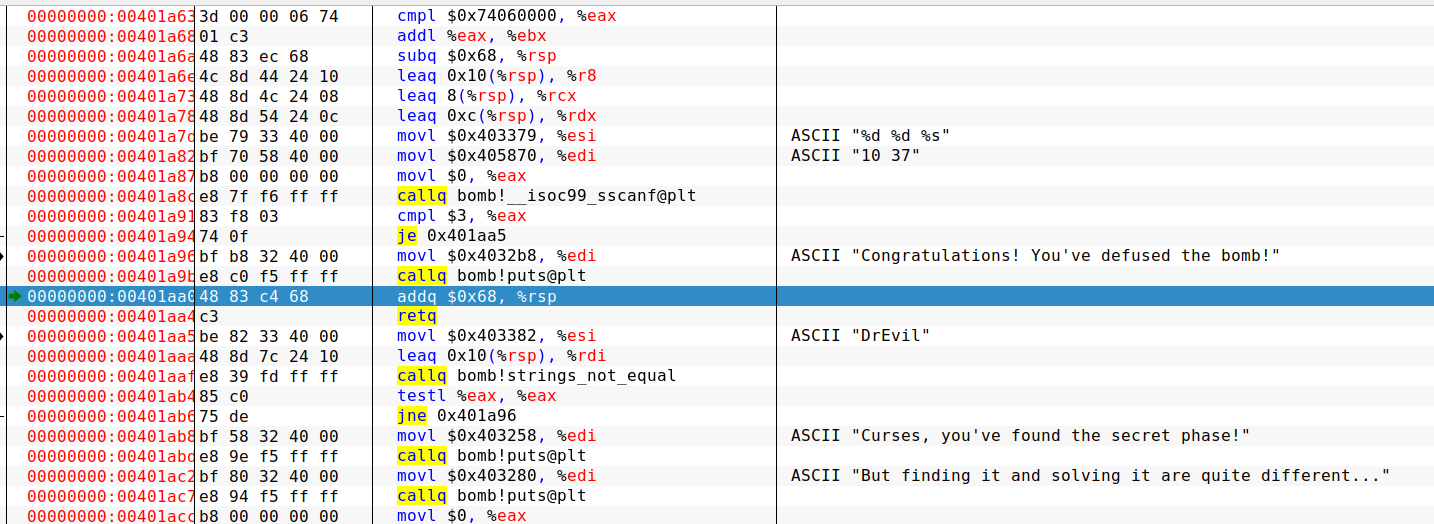


只有在phase\_defused过程才会调用secret\_phase。

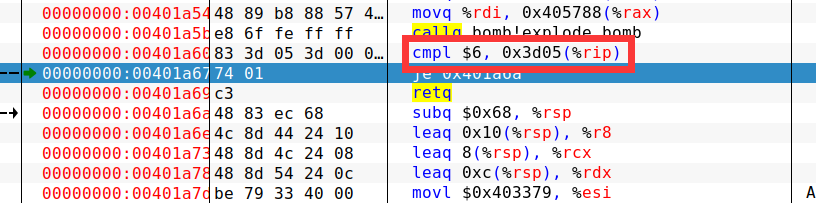




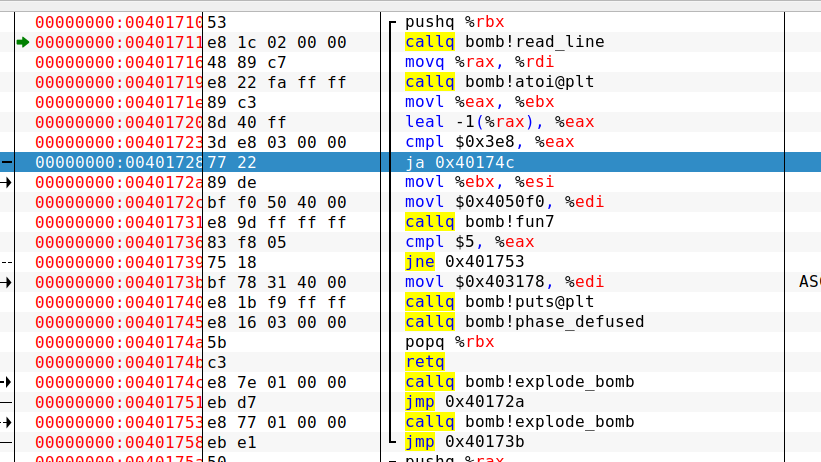
在主函数当中，每通过一关，都会调用依次phase\_defused函数。



在EDB动态调式中，发现phase\_defused函数中以一种奇怪的方式调用了函数scanf(“10 37”, “%d %d %s”);并且只有当返回值为3时(即scanf成功读入了3个内容)才能执行隐藏关卡。并且下面的代码要求%s读取的字符串为”DrEvil”，联想到第4关输入的答案就是10 37，推测这里要在第4关答案后面多输入一个字符串DrEvil。在执行完第4关后没有立即进入隐藏关卡，而是在6关都通过之后才进入了隐藏关卡，推测是程序中对通过的关卡进行了计数，只有在6关都通过了之后才能进入隐藏关卡。在phase\_defused函数中发现了一处比较印证了这一点。

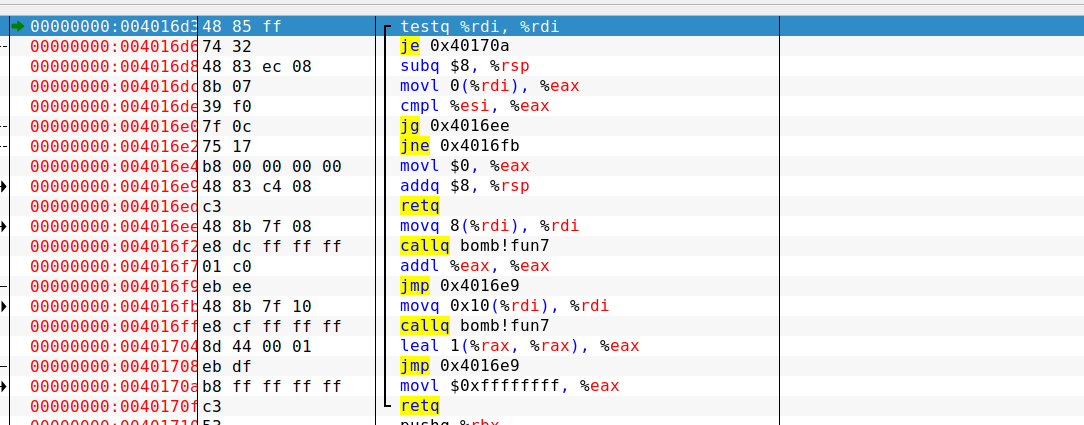


在进入secret关卡之后，分析secret关卡的过程。



可以发现，隐藏关卡要求输入一个字符串，然后调用atoi函数，把这个字符串转换成一个整数放在rax寄存器中，然后把这个值用ebx寄存器保存。接着又检测rax的值减去1后是否大于0x3e8(十进制的1000)，如果大于则炸弹爆炸，因此要求输入的数字小于等于1001。

接着，隐藏关卡调用了函数fun7，参数分别为0x4050f0和ebx寄存器的值，ebx寄存器的值也就是我们通过终端输入的整数。要求fun7函数的返回值等于5，则隐藏关卡通过。下面分析函数fun7。



函数fun7首先检测了rdi寄存器，也就是第一个参数是否为0，如果为0则返回0xffffffff(十进制-1)，而我们希望fun7返回0x5，也就不符合要求，因此不希望第一个参数为1。Fun7涉及到递归，我们把这段代码手工反编译，可以得到如下C语言代码：

int fun7(int \*p, int x)

{

    if (!p)

        return -1;

*//sub $8, %rsp*

    int ret = \*p;

    if (ret > x)

    {

        ret = 2 \* fun7(\*(p + 2), x); //(char\*)p+0x8

    }

    else if (ret < x)

    {

        ret = 2 \* fun7(\*(p + 4), x) + 1; //(char\*)p+0x10

}

else

{

ret = 0;

}

return ret;

}

通过这段递归发现p==0或x==\*p是终止条件，而p==0会引起返回-1，-2再乘2得到的仍然是一个负数，不会返回+5，因此猜测应该让x==\*p作为递归终止条件。

通过分析内存内容：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址p | p+0内容 | p+0x8内容 | p+0x10内容 |
| 0x4050f0 | 0x24 | 0x405110 | 0x405130 |
| 0x405110 | 0x8 | 0x405190 | 0x405150 |
| 0x405130 | 0x32 | 0x405170 | 0x4051b0 |
| 0x405190 | 0x6 | 0x4051f0 | 0x405250 |
| 0x405150 | 0x16 | 0x405270 | 0x405230 |
| 0x405170 | 0x2d | 0x4051d0 | 0x405290 |
| 0x4051b0 | 0x6b | 0x405210 | 0x4052b0 |
| 0x4051f0 | 0x01 | 0x0 | 0x0 |
| 0x405250 | 0x07 | 0x0 | 0x0 |
| 0x405270 | 0x14 | 0x0 | 0x0 |
| 0x405230 | 0x23 | 0x0 | 0x0 |
| 0x4051d0 | 0x28 | 0x0 | 0x0 |
| 0x405290 | 0x2f | 0x0 | 0x0 |
| 0x405210 | 0x63 | 0x0 | 0x0 |
| 0x4052b0 | 0x03e9 | 0x0 | 0x0 |

通过内存可以看到这部分内存类似一棵二叉树。

在递归返回时，为了让最终的返回值是5，可以通过这样的方法构造。

在最后一次调用函数时，返回0；在倒数第二次调用函数时，走2\*fun7+1分支，这样返回值会变成1；在倒数第三次调用函数时，走2\*fun7分支，这样会让返回值变成2；在倒数第四次调用函数时，走2\*fun7+1分支，这样会让返回值变成5。

因此这样分析，可以得到如下的函数调用过程。

第一次走2\*fun7+1分支：p=0x4050f0, x > 0x24

第二次走2\*fun7分支：p=0x405130, x < 0x32

第三次走2\*fun7+1分支：p=0x405170,x > 0x2d

第四次让函数返回0：p=0x405290, x==0x2f

这样分析后，所需输入的数字应该是0x2f，即为十进制的47。

至此为止，所求出每一关的solution分别为：

I am just a renegade hockey mom.

0 1 1 2 3 5

6 786

10 37 DrEvil

5 115

6 5 1 4 3 2

47