bomblab 报告

姓名：徐一诺

学号：2024201607

总分	phase_1	phase_2	phase_3	phase_4	phase_5	phase_6	secret_phase
0	0	0	0	0	0	0	0

解题报告

phase_1

题目答案:NVIDIA has been the single worst company we've ever dealt with. So NVIDIA, ___!

讲解题目思路
题目中将0x1d40（%rip）中存储的字符串地址加载到%rsi中，并调用了<string_not_equal>进行两个字符串的比较。如果两个字符串相等则将返回值%eax设为0。在test %eax，%eax中检测%eax的值，如果不是0则爆炸，是0则通过。因此我们只需要输入一个与0x1d40（%rip）中存储地址中的字符串相等的字符串即可解决该炸弹。

phase_2

题目答案:246012 901923 141684 659016

讲解题目思路
题目首先调用__isoc99_sscanf@plt读取参数，同时用cmp $0x4,%eax检查是否读入了4个参数（不是4个则爆炸），并分别存储在%rsp，%rsp+0x4,%rsp+0x8,%rsp+0xc中。随后进行了matA(*23)和matB(32*)两个矩阵的提取。分别存储在%rdi，%rsi中。用%r8d和%rax两个索引进行AB矩阵的乘法运算，得出一个2*2的矩阵。随后将这四个元素与输入的四个数进行比较，全部一致即可拆除炸弹。
这里采用了x/6wd $rsi和$rdi的方法得到A，B矩阵的具体值。A为456，897，561，234，168，948.B为252，286，103，528，69，531.
计算矩阵乘积即可得到答案246012 901923 141684 659016

phase_3

题目答案:0 I 187 （此答案不唯一）

讲解题目思路
题目首先调用__isoc99_sscanf@plt读取参数，同时用cmp $0x2,%eax检查是否读入了3个参数，并分别存储在%rsp，%rsp+0xf,%rsp+0x10,%rsp+0x14中。随后xor %al，0xf(%rsp)对第一个输入进行异或掩码操作，cmpl $0x7,0x10(%rsp)比较第二个参数是否小于等于7（且大于等于0）.随后以%rax（第二个输入）作为索引，%rdx作为跳转基地址。jmp *%rax就是根据第二个输入决定跳转地址（本质就是switch语句，第二个输入就是case）。case部分验证(这里以case 0为例)cmpl $0xbb,0x14(%rsp)即第三个输入是否为187。并在最后cmp %al,0xf(%rsp)验证第一个字符是否是case设置的字符I。最终即得出答案为0 I 187

phase_4

题目答案:31 AC

讲解题目思路
题目首先调用__isoc99_sscanf@plt读取参数，同时用cmp $0x2,%eax检查是否读入了2个参数，并分别存储在%rsp+0x10,%rsp+0xc中。
随后分别以调用以%edi=5为参数的<func4_1>和以%r8d=B,%ecx=C,%edx=A,%esi=0x4,%edi=0x5为参数的<func4_2>。将两个输入参数分别与两个函数返回值相比较，都相等即可拆除炸弹。
<func4_1>函数首先规定了f(0)=0,f(1)=1,递归式为f(n)=2f(n-1)+1。由此易得f(n)=2^(n)-1。所以第一个输入应该为31.
<func4_2>函数的是通过%esi=0x4,%edi=0x5的比较生成一个%r8d=B,%ecx=C,%edx=A决定的字符串。该例子下首先将%esi和f(%edi-1)进行比较，若后者大于等于前者则%edi-1后递归调用，则根据<func4_1>函数功能可知最后一步时%edi=0x3,%esi=0x4。随后经过判断易得会进入1721-172d部分生成字符串。mov %r12b,0x0(%rbp)，mov %r13b,0x1(%rbp)，movb $0x0,0x2(%rbp)根据参数可知为AC。
综上，最终答案为31 AC。

phase_5

题目答案:ttttvw（此答案不唯一）

讲解题目思路
题目首先调用<string_length>函数读入一个字符串，首先cmp $0x6,%eax检查字符串长度是否为6。然后加载一个数组地址（0x19ff(%rip)）到%rsi。通过遍历输入字符串中的字符低四位作为索引%rdx，在数组中寻找对应值并累加到%ecx中。最终在6个字符全部遍历后，cmp $0x3c,%ecx要求累加值等于60。
我们通过x/16d %rsi即可得到对应数组(选择16的原因时索引值使用的是低四位，所以推测数组有16个数)
2 10 6 1 12 16 9 3 4 7 14 5 11 8 15 13。
因此只要该字符串能够凑出6个索引对应的值和为60即可。这里采用ttttvw(12*4+9+3=60)。

phase_6

题目答案:5 1 4 6 3 2 puzzle（puzzle为进入secret_phase的额外字符串，前六个数字为本题答案）

讲解题目思路
题目首先调用<read_six_numbers>读取六个整数，存储在以地址%rsp+0x10开始的数组中。输入验证sub $0x1，%eax cmp $0x5,%eax保证每个输入的数字不大于6(不小于1)，mov 0x0(%r13,%rbx,4),%eax cmp %eax,0x0(%rbp)通过遍历(add $0x1,%rbx)确保输入的数字没有重复，即输入的六个数字是1-6的一个排列。
接下来通过%r12作索引，mov $0x7,%ecx mov %ecx,%eax sub (%r12),%eax用7分别减去六个数字得到新的一组排列，记为a[0-5]。
然后从0x48f9(%rip)中读取一个链表地址到%rdx，按照a[0]到a[5]的顺序链接一个新的链表b[0-5]，并将其存储在以地址%rsp+0x30开始的数组中。
最后mov 0x8(%rbx),%rax mov (%rax),%eax cmp %eax,(%rbx)遍历新链表，检查每个节点的值是否大于等于下一个节点的值，保证是降序排列即可。
我们通过x/24wx $rdx即可得到对应原始链表的值
node1 826 node2 941 node3 876 node4 223 node5 220 node6 912.
最后得出结果即为5 1 4 6 3 2

secret_phase

题目答案:33113（此答案不唯一）

讲解题目思路
首先要想办法进入secret_phase，经过观察发现在phase_defused中有call 1c18 <secret_phase>。分析可得需要满足如下条件才能够进入
cmpl $0x6,0x4577(%rip) 检查六个phase是否都完成了输入。movzbl 0x47c9(%rip),%ecx test %cl,%cl 检查input_strings+0x258是否不为空。lea 0x140b(%rip),%rsi call 1d0c <strings_not_equal> 检查存储在<input_strings+0x258>的字符串与%rsi中的字符串是否相等。
我们通过x/s $rsi可以得知字符串为puzzle。通过x/600s input_strings可以得知位于input_strings+0x258的是phase_6的输入。因此我们将puzzle放在phase_6答案的后面即可进入secret_phase

secret_phase中的内容主要是在调用func7后返回值%eax不等于0即可拆除炸弹。

首先构造了四个含有八个元素的数组。分别存储在%rsp,%rsp+0x20,%rsp+0x40,%rsp+0x60。数组分别为[-2,-1,1,2,2,1,-1,-2],[1,2,2,1,-1,-2,-2,-1],[-1,0,0,1,1,0,0,-1],[0,1,1,0,0,-1,-1,0]。
其中函数参数分别为%esi（a），%edx（b），%ecx（c）。函数内部的参数分别为add (%rsp,%rsi,4),%r8d（new_a），add 0x20(%rsp,%rsi,4),%r11d（new_b），add 0x40(%rsp,%r10,4),%eax（a1），add 0x60(%rsp,%r10,4),%edx（b1），%r9d为输入字符串索引，
%r10d为字符串数值（也是数组索引）。0x45ff（%rip）为一个由0和1组成的链表。
是一个递归调用自己的函数，新的参数分别由上一次递归的%r8d（new_a），%r11d（new_b），%ecx提供
这里唯一能够返回%eax=1的部分需要以下限制。
输入字符串长度不大于20，%esi=4,%edx=7,每一轮递归后第a1个链表节点偏移量为b1的值不能是1，第new_a个链表节点偏移量为new_b的值不能是1。
经过读取可知链表如下
row0 0 0 1 0 0 1 0 0
row1 0 0 0 1 0 0 0 1
row2 1 0 0 1 0 0 1 0
row3 1 0 0 0 0 0 0 0
row4 0 1 0 0 1 0 1 0
row5 1 0 0 1 1 0 0 0
row6 0 0 0 0 0 1 0 1
经过尝试可知33113是一个可行的字符串。