

课程信息

👉逆向过程

第一个通过函数跟入直接判断：

```

RIP 0x400ee9 (phase_1+9) ← call 0x401338
[ DISASM ]
0x400ee0 <phase_1> sub rsp, 8
0x400ee4 <phase_1+4> mov esi, 0x402400
➤ 0x400ee9 <phase_1+9> call strings not equal <0x401338>
rdi: 0x603780 (input_strings) ← 0x666564636261 /* 'abcdef' */
rsi: 0x402400 ← outsd dx, dword ptr [rsi] /* 'Border relations with Canada have never been better.' */
rdx: 0x1
rcx: 0x6

0x400eee <phase_1+14> test eax, eax
0x400ef0 <phase_1+16> je phase_1+23 <0x400ef7>

0x400ef2 <phase_1+18> call explode_bomb <0x40143a>

0x400ef7 <phase_1+23> add rsp, 8
0x400efb <phase_1+27> ret

0x400efc <phase_2> push rbp
0x400efd <phase_2+1> push rbx
0x400efe <phase_2+2> sub rsp, 0x28
[ STACK ]

```

Border relations with Canada have never been better.

第二步是输入6个数字，然后进行比较,整体的判断算法如下：

```

pwndbg> disassemble phase_2
Dump of assembler code for function phase_2:
0x0000000000400efc <+0>: push rbp
0x0000000000400efd <+1>: push rbx
0x0000000000400efe <+2>: sub rsp,0x28
0x0000000000400ef0 <+6>: mov rsi,rsi
0x0000000000400f05 <+9>: call 0x40145c <read_six_numbers>
0x0000000000400f0a <+14>: cmp DWORD PTR [rsp],0x1
0x0000000000400f0e <+18>: je 0x400f30 <phase_2+52>
0x0000000000400f10 <+20>: call 0x40143a <explode_bomb>
0x0000000000400f15 <+25>: jmp 0x400f30 <phase_2+52>
0x0000000000400f17 <+27>: mov eax,DWORD PTR [rbx-0x4]
0x0000000000400f1a <+30>: add eax,eax
0x0000000000400f1c <+32>: cmp DWORD PTR [rbx],eax
=> 0x0000000000400f1e <+34>: je 0x400f25 <phase_2+41>
0x0000000000400f20 <+36>: call 0x40143a <explode_bomb>
0x0000000000400f25 <+41>: add rbx,0x4
0x0000000000400f29 <+45>: cmp rbx,rbp
0x0000000000400f2c <+48>: jne 0x400f17 <phase_2+27>
0x0000000000400f2e <+50>: jmp 0x400f3c <phase_2+64>
0x0000000000400f30 <+52>: lea rbx,[rsp+0x4]
0x0000000000400f35 <+57>: lea rbp,[rsp+0x18]
0x0000000000400f3a <+62>: jmp 0x400f17 <phase_2+27>
0x0000000000400f3c <+64>: add rsp,0x28
0x0000000000400f40 <+68>: pop rbx
0x0000000000400f41 <+69>: pop rbp
0x0000000000400f42 <+70>: ret
End of assembly dump

```

除了第一个是1以外，其他的都是前面一个的2倍。

所以可以得到如下结果：

1 2 4 8 16 32

第三次判断：

第一次输入的数据作为序号找出第二次的值。

组合很多：1，0x137(311)

第四次判断：

就是一个递归，算法如下：

```

1 #include<stdio.h>
2 int test(int x,int y,int z){//y = 0, z = 14
3 {
4     int a = z; // a = eax
5     a = a - y; // y = esi
6     int b = a>>31; //b = ecx
7     a = (a + b)>>1;
8     b = a + b;
9     if(b>x)
10    {
11        z = b-1;
12        test(x,y,z);
13        a = a * 2;
14        return a;
15    }
16    else
17    {
18        a = 0;
19        if(b<x)
20        {
21            y = b + 1;
22            test(x,y,z);
23            a = a + a + 1;
24        }
25        else
26        {
27            return a;
28        }
29    }
30    return 0;
31 }

```

则知道输入的值为 7 和 0 (7 0)

第五次判断：

```

1 #include<stdio.h>
2 target_str = "maduiersnfotvbyl!So you think you can stop the bomb with ctrl-c, do you?"
3 char result[100];
4 char *target = "flyers";
5 int test(char *str)
6 {
7     int a = strlen(str); //eax
8     if(a==6)
9     {
10        for(a = 0;a<6;a++)
11        {
12            int b = ascii(str[a]);
13            b = b & 0xf;
14            result[a] = target_str[b];
15        }
16        if(target == result)
17        {
18            printf("good");
19        }
20    }
21    else
22    {
23        printf("error");
24    }
25 }

```

思路：

通过算法可得序列号为：9，15，14，5，6，7，对照ascii表求得：%&'

第六次判断：

算法如下：

第一轮，先判断第一个数字小于等于6，判断6个数中其他数字都与第一个数字的值不同。

第二轮，先判断第二个数字小于等于6，然后拿出其他数字比较其是否与第二个数字相等

第三路，判断第三个数字小于等于6，然后那出其他数字比较其与第三个数字是否相等

以此类推：

然后进行 7 - input 的求反运算，

下一步是将 node数组的地址根据输入的数组放入到相应的栈空间内

然后设置next将其连起来

然后第一部分的值域要递减，所以得到数字为4 3 2 1 6 5

👉感想

这个题目如果可以用一下工具，比如ida之类的话应该是很快的，但是只能用gdb，所以就头铁逆了几个小时。。。 （我也太菜了吧）

👉课堂录音

【👉点击最左侧的“+”，选择“附件音频与录制”功能，一边听课一边实时录音，遇到重点随时打标记】

👉课堂讲义

【👉点击最左侧的“+”，选择“附件”功能，将老师的课件文件作为附件进行上传，便于随时查看】

👉课堂记录

👉拓展资料


影像资料

网页书签

添加网页书签，了解更多信息，拓展视野，方便查阅。

印象笔记 | 工作必备效率应用

作为你的第二大脑，印象笔记可以帮助你简化工作、学习与生活。你可以在手机、电脑、平板、网页等多种设备和平台间，无缝同步每天的见闻、思考与灵感。快速保存微信文章、微博、网页等内容，一站

 <http://Yinxiang.com>

【👉点击最左侧的“+”，选择“网页书签”功能，将相关课程资料链接粘贴在对话框中，即可插入链接，便于随时查看】