

# Do you want to be a lightmaster?

## Flag1

思路一

打开exe 可以看到以下内容：

```
You have 500 workers (numbered 1-500) and there is 500 lights, numbered 1-500. The 1st man can press all the lights, the 2nd pressed once every other light, the 3rd pressed once every two lights and so on. Each person can only be sent once by you. After all pressed, there are 22 lights on. What is the number of the 22 lights?(After each person presses the light, you have 1 chance to check the status of any number of lights in the current state)
choose one person from 0 to 500(0 for answer-check part):
_
```

翻译过来就是：

你有 500 个工人（编号 1-500），有 500 个灯，编号 1-500。第一个人可以按下所有的灯，2号每隔一盏灯按一次，3号每两盏灯按一次，依此类推。每个人只能派一次。全部按下后，有22个灯亮。这22个灯的编号是什么？（每个人按下灯后，你都有1次机会检查当前状态下任意编号的灯的状态）

从0到500中选一个（0表示检查答案）：

这道题是来源于一个经典的数学问题：

来源：[\(30条消息\) 数学回味系列之4 - 开灯关灯问题奇偶数开关灯问题linolzhang的博客-CSDN博客](#)

有编号1~100个灯泡，起初所有的灯都是灭的。有100个同学来按灯泡开关，如果灯是亮的，那么按过开关之后，灯会灭掉。如果灯是灭的，按过开关之后灯会亮。

现在开始按开关。

Light 1 只会被第1个同学按；

Light 2 只会被第1个同学和第2个同学按；

Light 3 只会被第1个同学和第3个同学按；

Light 4 只会被第1个同学、第2个同学以及第4个同学按；

.....

Light N 只会被第1个同学、.....、第N个同学按；

到这你也许能够发现规律：light N 只会被第N的因数个同学按 所以当N的因数个数为奇数个的时候，灯才会是亮的。那么问题转换为什么数的因子是奇数个的呢？

除去1 这个因子 其他的因子应该是偶数个

因为任何正数都能被唯一表示成多个质因数幂次乘积的方式，所以当因子为偶数个的时候，指数和应该为偶数次，这也就意味着该数必须满足完全平方数

至此，从数学层面上，这题已经解决好了，也就是说，最后亮着的灯的编号应该就是500以内的完全平方数即

```
1 | 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484
```

连起来就是149162536496481100121144169196225256289324361400441484

我们进入answer-check

```
choose one person from 0 to 500(0 for answer-check part):
0
Here is answer-check part. For an example, if the lights' numbers are 1,2,3,4,5,6.
The answer is md5(123456).
https://www.md5hashgenerator.com/
please input your answer:
_
```

md5网站 <https://www.md5hashgenerator.com/>

最后可以得到

149162536496481100121144169196225256289324361400441484

Generate →

Your String	149162536496481100121144169196225256289324361400441484	
MD5 Hash	f14ccc08423bae2eb4e01d212abfa1e4	Copy
SHA1 Hash	7a63871bf3912794a43c4e81954fc5092c17774d	Copy

```
choose one person from 0 to 500(0 for answer-check part):
0
Here is answer-check part. For an example, if the lights' numbers are 1,2,3,4,5,6.
The answer is md5(123456).
https://www.md5hashgenerator.com/

please input your answer:
f14ccc08423bae2eb4e01d212abfa1e4
Congratulations!You are right!
```

```
1 | flag1 : r00t2023{f14ccc08423bae2eb4e01d212abfa1e4}
```

## 思路二 —— IDA静态分析

将lightgame.exe拖入ida

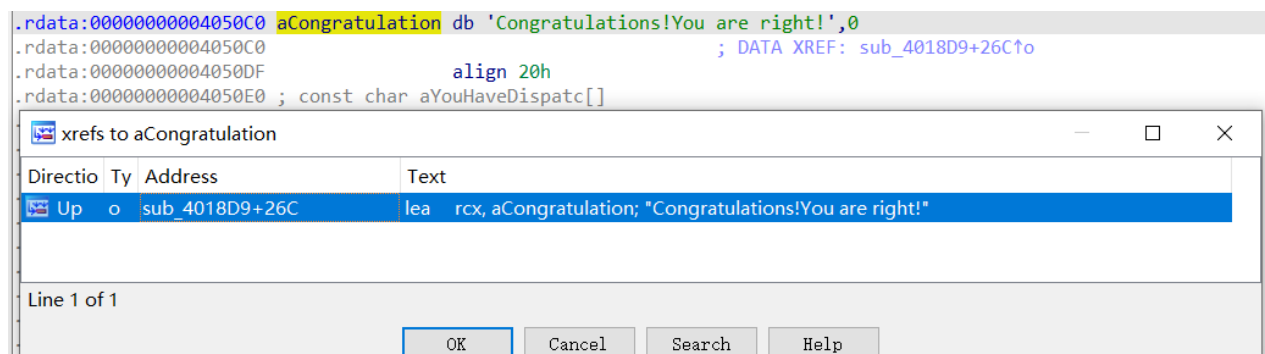
因为exe中有很多字符串提示，所以我们先进行shift+f12字符串搜索

我们能看到一些敏感信息，比如

Congratulations!You are right!

.rdata:000000... 00000090	C	Here is answer-check part. For an example,if the lights' numbers are 1,2,3,4,5,6.\n
.rdata:000000... 0000001B	C	please input your answer:\n
.rdata:000000... 00000007	C	Wrong!
.rdata:000000... 0000001F	C	Congratulations!You are right!
.rdata:000000... 00000039	C	You have dispatched that guy! please choose another one!
.rdata:000000... 0000001C	C	Error! please choose again!
.rdata:000000... 0000001C	C	Error! please choose again!
.rdata:000000... 000001C1	C	You have 500 workers (numbered 1-500) and there is 500 lights, numbered 1-500. The 1st ...

点进去 按x 查找引用



发现这个sub\_4018D9函数对他进行了引用

进到这个函数里面

```

1  _int64 sub_4018D9()
2  {
3      size_t v0; // rax
4      __int64 result; // rax
5      unsigned int v2; // eax
6      __int64 v3[32]; // [rsp+20h] [rbp-60h]
7      char v4[48]; // [rsp+120h] [rbp+A0h] BYREF
8      char Str[44]; // [rsp+150h] [rbp+D0h] BYREF
9      int v6; // [rsp+17Ch] [rbp+FCh]
10     void *Buf2; // [rsp+180h] [rbp+100h]
11     int v8; // [rsp+188h] [rbp+108h]
12     int i; // [rsp+18Ch] [rbp+10Ch]
13
14     puts(
15         "Here is answer-check part. For an example,if the lights' numbers are
16         1,2,3,4,5,6.\n"
17         "The answer is md5(123456).\n"
18         "https://www.md5hashgenerator.com/\n");
19     v8 = printf("please input your answer:\n") + 6;
20     scanf("%s", Str);
21     v0 = strlen(Str);
22     if ( v0 == v8 )
23     {
24         strcpy(v4, "r00t2023{w0w!u_r_here_we1c0me!!}");
25         v3[0] = 20i64;
26         v3[1] = 1i64;
27         v3[2] = 4i64;
28         v3[3] = 23i64;

```

```
28     v3[4] = 81i64;
29     v3[5] = 83i64;
30     v3[6] = 2i64;
31     v3[7] = 11i64;
32     v3[8] = 79i64;
33     v3[9] = 69i64;
34     v3[10] = 3i64;
35     v3[11] = 21i64;
36     v3[12] = 64i64;
37     v3[13] = 16i64;
38     v3[14] = 109i64;
39     v3[15] = 23i64;
40     v3[16] = 61i64;
41     v3[17] = 92i64;
42     v3[18] = 0i64;
43     v3[19] = 66i64;
44     v3[20] = 84i64;
45     v3[21] = 59i64;
46     v3[22] = 69i64;
47     v3[23] = 84i64;
48     v3[24] = 3i64;
49     v3[25] = 2i64;
50     v3[26] = 82i64;
51     v3[27] = 11i64;
52     v3[28] = 4i64;
53     v3[29] = 16i64;
54     v3[30] = 68i64;
55     v3[31] = 73i64;
56     Buf2 = malloc(4ui64);
57     for ( i = 0; i < v8; ++i )
58     {
59         v6 = v3[i] ^ (unsigned __int8)v4[i];
60         *((_BYTE *)Buf2 + i) = v6;
61     }
62     if ( !memcmp(Str, Buf2, v8) )
63     {
64         puts("Congratulations!You are right!");//通过比较字符串Str和Buf2, 若相同则输出了
        Congratulations
65         v2 = strlen(v4);
66         sub_401864(v4, v2);
67         result = 1i64;
68     }
69     else
70     {
71         result = 0i64;
72     }
73 }
74 else
75 {
76     puts("Wrong!");
77     result = 0i64;
```

```
78     }
79     return result;
80 }
```

关键逻辑：比较**Str**和**Buf2**

```
1  scanf("%s", Str); // Str即为我们输入的字符串，也就是说当我们输入的字符串和Buf2相同
2  v0 = strlen(Str);
3  if ( v0 == v8 )
4  {
5      strcpy(v4, "r00t2023{w0w!u_r_here_we1c0me!!}");
6      v3[0] = 20i64;
7      v3[1] = 1i64;
8      v3[2] = 4i64;
9      v3[3] = 23i64;
10     v3[4] = 81i64;
11     v3[5] = 83i64;
12     v3[6] = 2i64;
13     v3[7] = 11i64;
14     v3[8] = 79i64;
15     v3[9] = 69i64;
16     v3[10] = 3i64;
17     v3[11] = 21i64;
18     v3[12] = 64i64;
19     v3[13] = 16i64;
20     v3[14] = 109i64;
21     v3[15] = 23i64;
22     v3[16] = 61i64;
23     v3[17] = 92i64;
24     v3[18] = 0i64;
25     v3[19] = 66i64;
26     v3[20] = 84i64;
27     v3[21] = 59i64;
28     v3[22] = 69i64;
29     v3[23] = 84i64;
30     v3[24] = 3i64;
31     v3[25] = 2i64;
32     v3[26] = 82i64;
33     v3[27] = 11i64;
34     v3[28] = 4i64;
35     v3[29] = 16i64;
36     v3[30] = 68i64;
37     v3[31] = 73i64;
38     Buf2 = malloc(4ui64);
39     for ( i = 0; i < v8; ++i )
40     {
41         v6 = v3[i] ^ (unsigned __int8)v4[i];
42         *((_BYTE *)Buf2 + i) = v6;
```

```

43     }
44     if ( !memcmp(Str, Buf2, v8) )
45     {
46         puts("Congratulations!You are right!");//通过比较字符串Str和Buf2, 若相同则输出了
        Congratulations
47         v2 = strlen(v4);
48         sub_401864(v4, v2);
49         result = 1i64;
50     }

```

Str 是输入的字符串，所以正确答案应该存在Buf2数组中

Buf2数组的生成是通过：

```

1  for ( i = 0; i < v8; ++i )
2  {
3      v6 = v3[i] ^ (unsigned __int8)v4[i];
4      *((_BYTE *)Buf2 + i) = v6;
5  }

```

v3数组和v4数组异或得到Buf2

v3数组显然直接给出了

```

1  strcpy(v4, "r00t2023{w0w!u_r_here_we1c0me!!}");

```

v4 = r00t2023{w0w!u\_r\_here\_we1c0me!!}

用python简单写一下

```

1  v3 =
    [20,1,4,23,81,83,2,11,79,69,3,21,64,16,109,23,61,92,0,66,84,59,69,84,3,2,82,11,4,16,6
    8,73]
2  v4 = "r00t2023{w0w!u_r_here_we1c0me!!}"
3  for i in range(len(v3)):
4      print(chr(v3[i]^ord(v4[i])),end="")

```

输出结果

```

1  f14ccc08423bae2eb4e01d212abfa1e4

```

不知道大家是否记得 第二阶段提示：Congratulations后面藏了东西

```
1  if ( !memcmp(Str, Buf2, v8) )
2  {
3      puts("Congratulations!You are right!");//通过比较字符串Str和Buf2, 若相同则输出了
    Congratulations
4      v2 = strlen(v4);
5      sub_401864(v4, v2);
6      result = 1i64;
7  }
```

在Congratulations后面有一个函数 sub\_401864，这里先记住传进去的参数v4，v2

v4是r00t2023{w0w!u\_r\_here\_we1c0me!!}；v2是v4的长度

我们进到这个函数里看看

```
1  __int64 __fastcall sub_401864(__int64 a1, int a2)
2  {
3      __int64 v3[4]; // [rsp+20h] [rbp-30h] BYREF
4      char v4; // [rsp+40h] [rbp-10h]
5      int v5; // [rsp+4Ch] [rbp-4h]
6
7      v5 = a2 - 1;
8      v3[0] = 0xC24DB39773225A3i64;
9      v3[1] = 0x2791505B78B52975i64;
10     v3[2] = 0x724E5E5DA98D0A2Bi64;
11     v3[3] = 0x8DDC2F4DE3569D0Bui64;
12     v4 = -99;
13     return sub_4016D6((__int64)v3, 0x21u, a1, a2 - 1);
14 }
```

这个函数里v3显然是一个有用的信息，先记下，除此之外没什么，但我们看到return 中有一个函数sub\_4016D6，参数里面传进了v3 和 a1，a1也就是v4，也就是传入了v3和v4

再进去看看

```
1  __int64 __fastcall sub_4016D6(__int64 a1, unsigned int a2, __int64 a3, unsigned int
    a4)
2  {
3      __int64 result; // rax
4      _BYTE v5[303]; // [rsp+0h] [rbp-80h] BYREF
5      char v6; // [rsp+12Fh] [rbp+AFh]
6      int v7; // [rsp+130h] [rbp+B0h]
7      unsigned int i; // [rsp+134h] [rbp+B4h]
8      int v9; // [rsp+138h] [rbp+B8h]
9      int v10; // [rsp+13Ch] [rbp+BCh]
10 }
```



```

11  sub_4015A3(&v5[32], a3, a4);
12  v10 = 0;
13  v9 = 0;
14  v7 = 0;
15  for ( i = 0; ; ++i )
16  {
17      result = i;
18      if ( i >= a2 )
19          break;
20      v10 = (v10 + 1) % 256;
21      v9 = ((unsigned __int8)v5[v10 + 32] + v9) % 256;
22      v6 = v5[v10 + 32];
23      v5[v10 + 32] = v5[v9 + 32];
24      v5[v9 + 32] = v6;
25      v7 = (unsigned __int8)(v5[v10 + 32] + v5[v9 + 32]);
26      *(_BYTE *)(a1 + i) ^= v5[v7 + 32];
27  }
28  return result;
29  }

```

熟悉RC4的朋友 应该能一眼看出 这就是一个RC4加密

不熟悉的朋友可以参考这篇文章了解一下

[RC4加密算法详解lmn的博客-CSDN博客](#)

**RC4中主要有两个算法：**

- 密钥调度算法
  - 建立状态表S，S中的元素被初始化为0-255
  - 建立临时状态表T，设密钥K长度为L个字节，若L>256，则K赋 给T，多余字节丢弃，若字节L<256循环使用
  - S初始置换 伪代码如下：

```

1  j = 0;
2  for(i = 0; i < 256; i++){
3      j = (j + S[i] + T[i]) mod 256;
4      swap(S[i], S[j]);
5  }

```

- 伪随机数生成算法

- 生成密钥流，伪代码如下

```
1  i,j = 0;
2  while(true){
3      i = (i+1) mod 256;
4      j = (j+S[i]) mod 256;
5      swap(S[i],S[j]);
6      t = (S[i]+S[j]) mod 256;
7      k = S[t];
8  }
```

回到sub\_4016D6函数

```
1  sub_4015A3(&v5[32], a3, a4);
2  v10 = 0;
3  v9 = 0;
4  v7 = 0;
5  for ( i = 0; ; ++i )
6  {
7      result = i;
8      if ( i >= a2 )
9          break;
10     v10 = (v10 + 1) % 256;
11     v9 = ((unsigned __int8)v5[v10 + 32] + v9) % 256;
12     v6 = v5[v10 + 32];
13     v5[v10 + 32] = v5[v9 + 32];
14     v5[v9 + 32] = v6;
15     v7 = (unsigned __int8)(v5[v10 + 32] + v5[v9 + 32]);
16     *(_BYTE *)(a1 + i) ^= v5[v7 + 32];
17 }
```

这段后面的for循环显然与伪随机数生成密钥有关

那么前面的sub\_4015A3 应该就是密钥生成 传入的参数回溯到最开始，发现就是v4

也就是说v4就是key，那么v3应该就是data 待加密的数据了

至此我们明确了key和data，那么剩下的就是解密了

```
1  v4 = "r00t2023{w0w!u_r_here_we1c0me!!}";
2  v3[0] = 0xC24DB39773225A3i64;
3  v3[1] = 0x2791505B78B52975i64;
4  v3[2] = 0x724E5E5DA98D0A2Bi64;
5  v3[3] = 0x8DDC2F4DE3569D08ui64;
```

这里又涉及到一个大小端序的问题

详细请参考

[\(30条消息\) 记 IDA 中对内存小端逆序的理解逆向大小端序沐一·林的博客-CSDN 博客](#)

所以v3数组正确顺序应该是

```
1 unsigned char v3[] = {0xa3, 0x25, 0x32, 0x77, 0x39, 0xdb, 0x24, 0x0c, 0x75, 0x29,
    0xb5, 0x78, 0x5b, 0x50, 0x91, 0x27, 0x2b,
    0x0a, 0x8d, 0xa9, 0x5d, 0x5e, 0x4e, 0x72, 0x0b, 0x9d, 0x56, 0xe3, 0x4d, 0x2f, 0xdc, 0x8d, 0x9d};
```

接下来可以选择网站直接解密，也可以写脚本，这里介绍写脚本的方法

因为RC4是对称加密，所以RC4的加解密密钥是相同的，又因为RC4是采用异或的方式加密，所以加解密的流程完全相同

## EXP

```
1 #include<stdio.h>
2
3 /*
4  RC4初始化函数
5  */
6 void rc4_init(unsigned char* s, unsigned char* key, unsigned long Len_k)
7 {
8     int i = 0, j = 0;
9     char k[256] = { 0 };
10    unsigned char tmp = 0;
11    for (i = 0; i < 256; i++) {
12        s[i] = i;
13        k[i] = key[i % Len_k];
14    }
15    for (i = 0; i < 256; i++) {
16        j = (j + s[i] + k[i]) % 256;
17        tmp = s[i];
18        s[i] = s[j];
19        s[j] = tmp;
20    }
21 }
22
23 /*
24  RC4加解密函数
25  unsigned char* Data    加解密的数据
26  unsigned long Len_D    加解密数据的长度
27  unsigned char* key      密钥
```

```

28 unsigned long Len_k      密钥长度
29 */
30 void rc4_crypt(unsigned char* Data, unsigned long Len_D, unsigned char* key,
    unsigned long Len_k) //加解密
31 {
32     unsigned char s[256];
33     rc4_init(s, key, Len_k);
34     int i = 0, j = 0, t = 0;
35     unsigned long k = 0;
36     unsigned char tmp;
37     for (k = 0; k < Len_D; k++) {
38         i = (i + 1) % 256;
39         j = (j + s[i]) % 256;
40         tmp = s[i];
41         s[i] = s[j];
42         s[j] = tmp;
43         t = (s[i] + s[j]) % 256;
44         Data[k] = Data[k] ^ s[t];
45     }
46 }
47 void RC4()
48 {
49     //字符串密钥
50     unsigned char key[] = "r00t2023{w0w!u_r_here_we1c0me!!}";
51     unsigned long key_len = sizeof(key) - 1;
52     //数组密钥
53     //unsigned char key[] = {};
54     //unsigned long key_len = sizeof(key);
55
56     //加解密数据
57     unsigned char data[] = {0xa3, 0x25, 0x32, 0x77, 0x39, 0xdb, 0x24, 0x0c,
58         0x75, 0x29, 0xb5, 0x78, 0x5b, 0x50, 0x91, 0x27, 0x2b, 0x0A,
59         0x8d, 0xa9, 0x5d, 0x5e, 0x4e, 0x72, 0x0b, 0x9d, 0x56, 0xe3, 0x4d, 0x2f, 0xdc, 0x8d, 0x9d};
60     //加解密
61     rc4_crypt(data, sizeof(data), key, key_len);
62
63     for (int i = 0; i < sizeof(data); i++)
64     {
65         printf("%c", data[i]);
66     }
67     printf("\n");
68     return;
69 }
70 int main(){
71     RC4();
72     return 0;
73 }

```

输出结果: r00t2023{h0pe\_y0u\_wi11\_Enj0Y\_1t!}