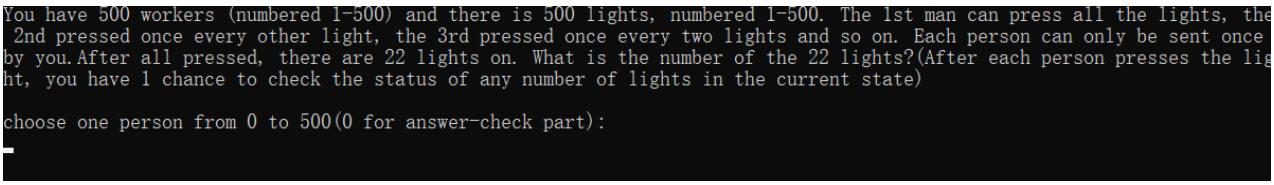
Do you want to be a lightmaster?

Flag1

思路一

打开exe 可以看到以下内容：



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 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1491625364964811001211441691962252562893243614004414 84

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}
```

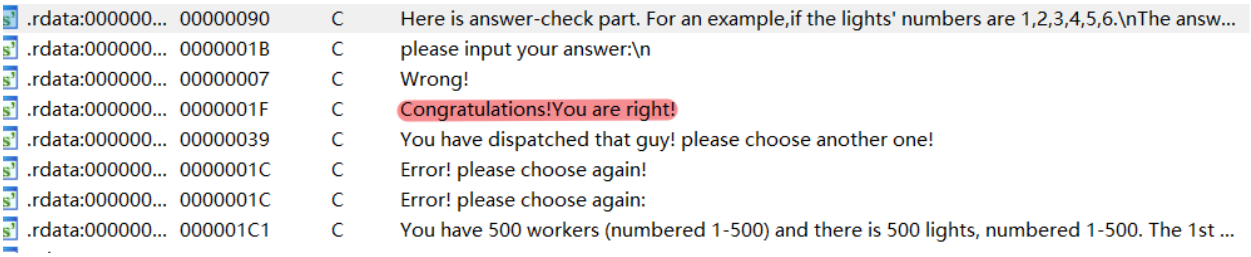
思路二 —— **IDA**静态分析

将lightgame.exe拖入ida

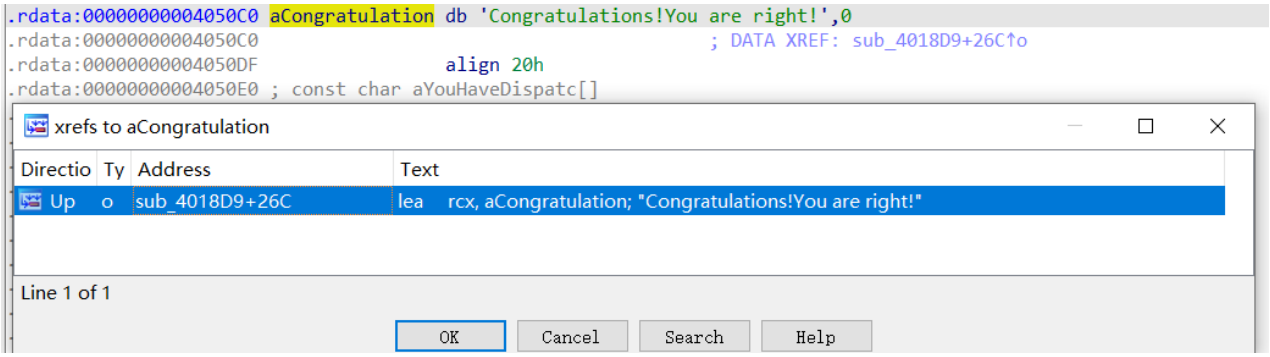
因为exe中有很多字符串提示，所以我们先进行shift+f12字符串搜索

我们能看到一些敏感信息，比如

Congratulations!You are right!



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

```
_int64 sub_4018D9()
{
  size_t v0; // rax
  __int64 result; // rax
  unsigned int v2; // eax
  __int64 v3[32]; // [rsp+20h] [rbp-60h]
  char v4[48]; // [rsp+120h] [rbp+A0h] BYREF
  char Str[44]; // [rsp+150h] [rbp+D0h] BYREF
  int v6; // [rsp+17Ch] [rbp+FCh]
  void *Buf2; // [rsp+180h] [rbp+100h]
  int v8; // [rsp+188h] [rbp+108h]
  int i; // [rsp+18Ch] [rbp+10Ch]

  puts(
    "Here is answer-check part. For an example,if the lights' numbers are
1,2,3,4,5,6.\n"
    "The answer is md5(123456).\n"
    "https://www.md5hashgenerator.com/\n");
  v8 = printf("please input your answer:\n") + 6;
  scanf("%s", Str);
  v0 = strlen(Str);
  if ( v0 == v8 )
  {
    strcpy(v4, "r00t2023{w0w!u_r_here_we1c0me!!}");
    v3[0] = 20i64;
    v3[1] = 1i64;
    v3[2] = 4i64;
    v3[3] = 23i64;
```

```
    v3[4] = 81i64;
    v3[5] = 83i64;
    v3[6] = 2i64;
    v3[7] = 11i64;
    v3[8] = 79i64;
    v3[9] = 69i64;
    v3[10] = 3i64;
    v3[11] = 21i64;
    v3[12] = 64i64;
    v3[13] = 16i64;
    v3[14] = 109i64;
    v3[15] = 23i64;
    v3[16] = 61i64;
    v3[17] = 92i64;
    v3[18] = 0i64;
    v3[19] = 66i64;
    v3[20] = 84i64;
    v3[21] = 59i64;
    v3[22] = 69i64;
    v3[23] = 84i64;
    v3[24] = 3i64;
    v3[25] = 2i64;
    v3[26] = 82i64;
    v3[27] = 11i64;
    v3[28] = 4i64;
    v3[29] = 16i64;
    v3[30] = 68i64;
    v3[31] = 73i64;
    Buf2 = malloc(4ui64);
    for ( i = 0; i < v8; ++i )
    {
      v6 = v3[i] ^ (unsigned __int8)v4[i];
      *((_BYTE *)Buf2 + i) = v6;
    }
    if ( !memcmp(Str, Buf2, v8) )
    {
      puts("Congratulations!You are right!");//通过比较字符串Str和Buf2，若相同则输出了Congratulations
      v2 = strlen(v4);
      sub_401864(v4, v2);
      result = 1i64;
    }
    else
    {
      result = 0i64;
    }
  }
  else
  {
    puts("Wrong!");
    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：

```
for ( i = 0; i < v8; ++i )
{
    v6 = v3[i] ^ (unsigned __int8)v4[i];
    *((_BYTE *)Buf2 + i) = v6;
}
```

v3数组和v4数组异或得到Buf2

v3数组显然直接给出了

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

v4 = r00t2023{w0w!u_r_here_we1c0me!!}

用python简单写一下

```
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]
v4 = "r00t2023{w0w!u_r_here_we1c0me!!}"
for i in range(len(v3)):
    print(chr(v3[i]^ord(v4[i])),end="")
```

输出结果

```
f14ccc08423bae2eb4e01d212abfa1e4
```

Flag2

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

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

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

```
__int64 __fastcall sub_401864(__int64 a1, int a2)
{
  __int64 v3[4]; // [rsp+20h] [rbp-30h] BYREF
  char v4; // [rsp+40h] [rbp-10h]
  int v5; // [rsp+4Ch] [rbp-4h]

  v5 = a2 - 1;
  v3[0] = 0xC24DB39773225A3i64;
  v3[1] = 0x2791505B78B52975i64;
  v3[2] = 0x724E5E5DA98D0A2Bi64;
  v3[3] = 0x8DDC2F4DE3569D0Bui64;
  v4 = -99;
  return sub_4016D6((__int64)v3, 0x21u, a1, a2 - 1);
}
```

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

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

```

```
  sub_4015A3(&v5[32], a3, a4);
  v10 = 0;
  v9 = 0;
  v7 = 0;
  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    v10 = (v10 + 1) % 256;
    v9 = ((unsigned __int8)v5[v10 + 32] + v9) % 256;
    v6 = v5[v10 + 32];
    v5[v10 + 32] = v5[v9 + 32];
    v5[v9 + 32] = v6;
    v7 = (unsigned __int8)(v5[v10 + 32] + v5[v9 + 32]);
    *(_BYTE *)(a1 + i) ^= v5[v7 + 32];
  }
  return result;
}
```

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*Imn*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:

```
j = 0;
for(i = 0;i<256;i++){
    j = (j+S[i]+T[i])mod 256;
    swap(S[i],S[j]);
}
```

- 伪随机数生成算法

- 生成密钥流，伪代码如下

```
i,j = 0;
while(true){
    i = (i+1) mod 256;
    j = (j+S[i]) mod 256;
    swap(S[i],S[j]);
    t = (S[i]+S[j]) mod 256;
    k = S[t];
}
```

回到**sub_4016D6**函数

```
sub_4015A3(&v5[32], a3, a4);
v10 = 0;
v9 = 0;
v7 = 0;
for ( i = 0; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    v10 = (v10 + 1) % 256;
    v9 = ((unsigned __int8)v5[v10 + 32] + v9) % 256;
    v6 = v5[v10 + 32];
    v5[v10 + 32] = v5[v9 + 32];
    v5[v9 + 32] = v6;
    v7 = (unsigned __int8)(v5[v10 + 32] + v5[v9 + 32]);
    *(_BYTE *)(a1 + i) ^= v5[v7 + 32];
  }
```

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

```
v4 = "r00t2023{w0w!u_r_here_we1c0me!!}";
v3[0] = 0xC24DB39773225A3i64;
v3[1] = 0x2791505B78B52975i64;
v3[2] = 0x724E5E5DA98D0A2Bi64;
v3[3] = 0x8DDC2F4DE3569D0Bui64;
```

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

```
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

```
#include<stdio.h>

/*
RC4初始化函数
*/
void rc4_init(unsigned char* s, unsigned char* key, unsigned long Len_k)
{
    int i = 0, j = 0;
    char k[256] = { 0 };
    unsigned char tmp = 0;
    for (i = 0; i < 256; i++) {
        s[i] = i;
        k[i] = key[i % Len_k];
    }
    for (i = 0; i < 256; i++) {
        j = (j + s[i] + k[i]) % 256;
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }
}

/*
RC4加解密函数
unsigned char* Data     加解密的数据
unsigned long Len_D     加解密数据的长度
unsigned char* key      密钥
```

```
unsigned long Len_k     密钥长度
*/
void rc4_crypt(unsigned char* Data, unsigned long Len_D, unsigned char* key, unsigned long Len_k) //加解密
{
    unsigned char s[256];
    rc4_init(s, key, Len_k);
    int i = 0, j = 0, t = 0;
    unsigned long k = 0;
    unsigned char tmp;
    for (k = 0; k < Len_D; k++) {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
        t = (s[i] + s[j]) % 256;
        Data[k] = Data[k] ^ s[t];
    }
}
void RC4()
{
    //字符串密钥
    unsigned char key[] = "r00t2023{w0w!u_r_here_we1c0me!!}";
    unsigned long key_len = sizeof(key) - 1;
    //数组密钥
    //unsigned char key[] = {};
    //unsigned long key_len = sizeof(key);

    //加解密数据
    unsigned char data[] = {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};
    //加解密
    rc4_crypt(data, sizeof(data), key, key_len);

    for (int i = 0; i < sizeof(data); i++)
    {
        printf("%c", data[i]);
    }
    printf("\n");
    return;
}
int main(){
    RC4();
    return 0;
}

```

输出结果：**r00t2023{h0pe_y0u_wi11_Enj0Y_1t!}**