Web 部分

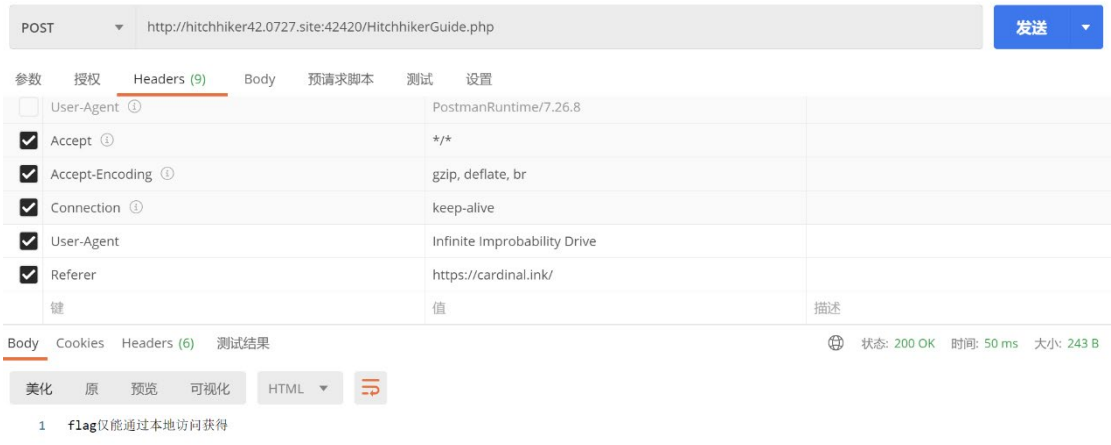
Web1 Hitchhiking_in_the_Galaxy

打开网页，啥也没有。查看源代码，发现 HitchhikerGuide.php，打开，没啥用。

于是打开 Postman，对 http://hitchhiker42.0727.site:42420/HitchhikerGuide.php 发送 post 请求：

提示很明显，改 UA：

好家伙，那么继续改 Referer：


POST
http://hitchhiker42.0727.site:42420/HitchhikerGuide.php
发送
参数 授权 Headers (9) Body 预请求脚本 测试 设置
User-Agent PostmanRuntime/7.26.8
Accept */*
Accept-Encoding gzip, deflate, br
Connection keep-alive
User-Agent Infinite Improbability Drive
Referer https://cardinal.ink/
键 值 描述
Body Cookies Headers (6) 测试结果
状态: 200 OK 时间: 50 ms 大小: 243 B
美化 原 预览 可视化 HTML
1 flag仅能通过本地访问获得

老套娃了：


POST
http://hitchhiker42.0727.site:42420/HitchhikerGuide.php
发送
参数 授权 Headers (10) Body 预请求脚本 测试 设置
Accept */*
Accept-Encoding gzip, deflate, br
Connection keep-alive
User-Agent Infinite Improbability Drive
Referer https://cardinal.ink/
X-Forwarded-For 127.0.0.1
键 值 描述
Body Cookies Headers (6) 测试结果
状态: 200 OK 时间: 17 ms 大小: 265 B
美化 原 预览 可视化 HTML
1 hgame{s3Cret_0f_HitCHhiking_in_the_GAl@xy_i5_dOnT_p@nic!}

完工！

Web2 watermelon

~~*谁能想到我第一次玩合成大西瓜竟然是在 HGAME 呢*~~

首先当然是直接看 js 代码，没什么眉目。

于是想到了万能的 Github，经过一番搜索，呐：

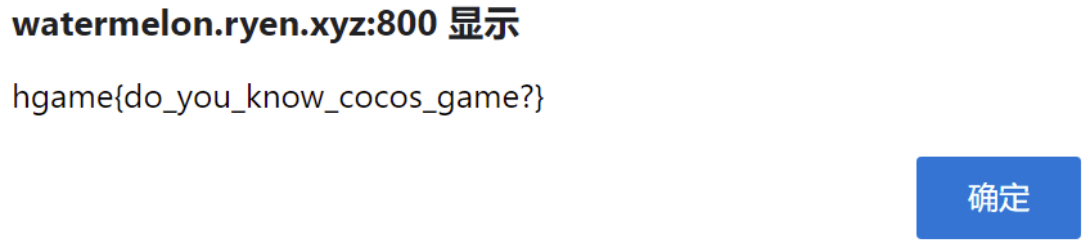
https://github.com/liyupi/daxigua/blob/9df37ad866d6cc875130cbfe7543f189c2b3ab52/src/project.js#L3495

完美！

再次 F12，检索到：

```
3433 HighestFruit(), null != t.node.getComponent(cc.RigidBody) && (t.node.getCc
3434
3435
3436 r() && (a.default.score += this.fruitNumber + 2000, u.default.Instance.Set
3437
3438
3439 L(o.fruitNumber, n.node.position, n.node.width), i.default.Instance.create
3440 uitData").getNumber() && (a.default.score += this.fruitNumber + 2000, u.de
3441
3442
3443 L(o.fruitNumber, n.node.position, n.node.width), i.default.Instance.create
3444
3445
3446
3447
3448
```

果断改成 2000。然后开玩：



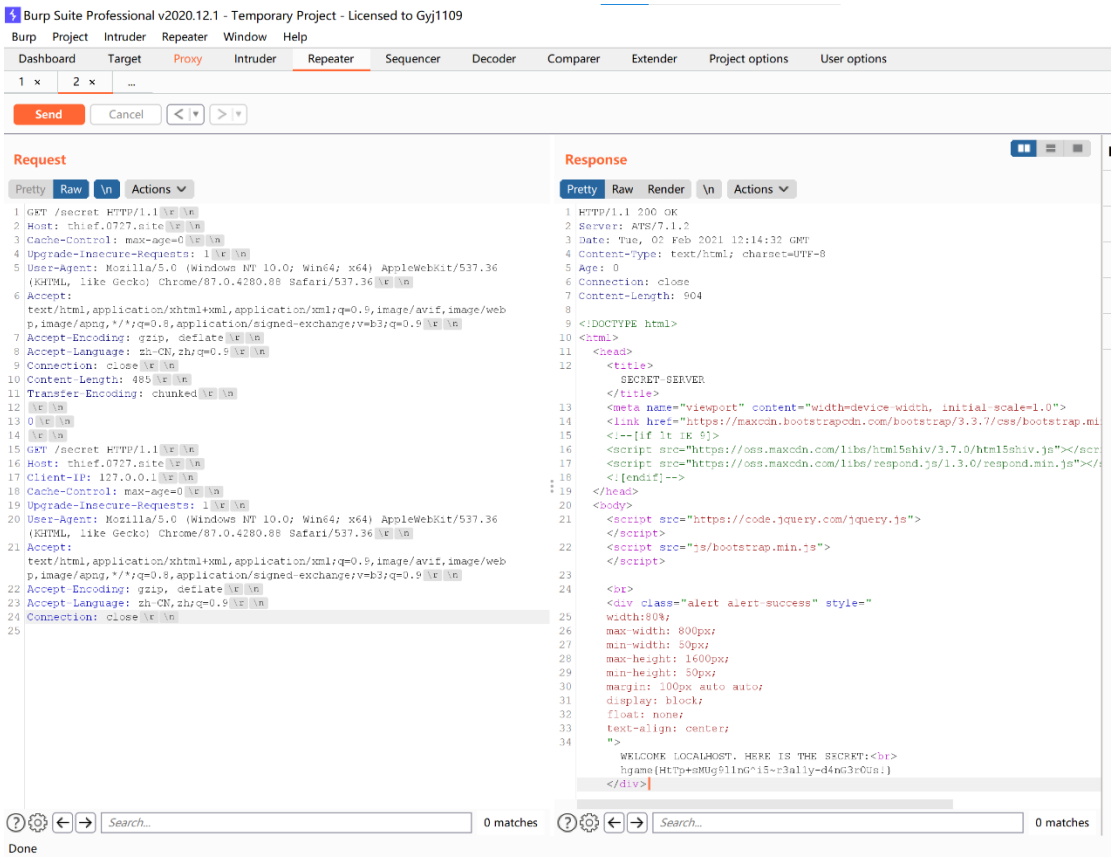
谁能想到最难的竟然是如何快速结束游戏……

Web3 宝藏走私者

提示太明显了，改 Client-IP。

打开 Burp Suite，直接改 Client-IP 发现没用，看到 Response 里有 Server: ATS/7.1.2，显然是用了反代。上 HTTP 协议走私（终于理解了题目里的走私是什么意思）。

伪造一个数据包，没啥难度。直接上图：



（不过第一次知道，原来 Burp Suite 会自动修改 Content-Length 的值，亏得我还自己去算）

Flag 直接出现。

Web4 智商检测鸡

说实话，肯定是我自己的智商出了问题。一看到 100 道题，全是 ax+b 形式的，我直接写了个 Python 脚本，然后“做”了 100 道题.....

```python
import re

while 1:
    line = input()

    if matchObj := re.match(
            r'∫(-?[0-9]{2})([0-9]{2})\(([0-9]*)x\+([0-9]+)\)dx', line,
            re.M | re.I):
        a = int(matchObj.group(3)) / 2
        b = int(matchObj.group(4))
        x = int(matchObj.group(1))
        y = int(matchObj.group(2))
        print(a * y * y + b * y - a * x * x - b * x)
    else:
        print("No match!!")
```

其实，中途发现，cookie 是有规律的。比如最后的是

eyJzb2x2aW5nIjoxMDB9.YBdwjw.6ubtsdV-kWd6BtwWxKrB3XvIRK0

base64 解码以后：

Input length: 55 lines: 1

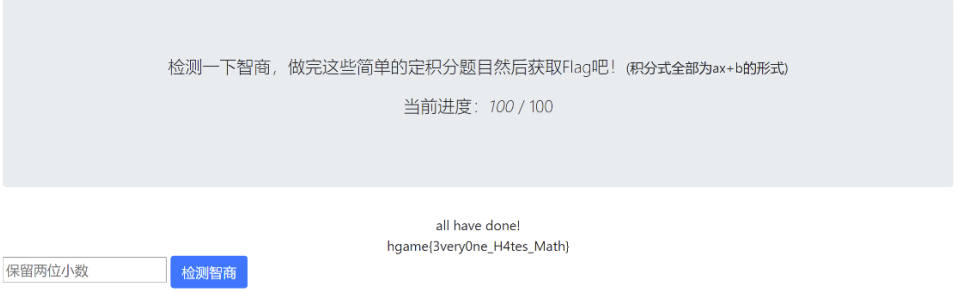
```
eyJzb2x2aW5nIjoxMDB9.YBdwjw.6ubtsdV-kWd6BtwWxKrB3XvIRK0
```

Output time: 1ms length: 39 lines: 1

```
{"solving":100}`.p..®nÛ.VE.è.p[.«.uï!.´
```

开头很明显了。只是我没去研究，反正就 100 道，配合脚本很快就好了.....

最后结果：



Web5 走私者的愤怒

和 Web3 一毛一样，就是测试的时候麻烦点，多试几次就好了。包也长得一样。具体过程略，

结果如下（试了试 Burp 的在浏览器中显示的功能）：

```
WELCOME LOCALHOST. HERE IS THE SECRET:
hgame{Fe3l^tHe-4N9eR+oF_5mu9gl3r!!}
```

Reverse 部分

Re1 apacha

IDA 打开，稍加整理，发现加密过程在 sub_1447()，而 sub_1550()为比对。

打开 sub_1447()，发现一堆位移和异或运算。然后发现，所有步骤逆回去重新做一遍就行了。

这回不用 Python 了，改用 C，所有加密部分代码直接复制过来就能用，舒服！

```
int main() {
    int a[] = {1, 2, 3, 4};
    __int64 v8, result, a3 = (__int64)a;
    unsigned int v5, v6,
        t[] = {2654435769, 1013904242, 3668340011, 2027808484, 387276957, 3041712726, 1401181199};
    DWORD input[35] = {0x0E74EB323, 0x0B7A72836, 0x59CA6FE2,  0x967CC5C1,  0x0E7802674, 0x3D2D54E6,
                       0x8A9D0356,  0x99DCC39C,  0x7026D8ED,  0x6A33FDAD,  0x0F496550A, 0x5C9C6F9E,
                       0x1BE5D04C,  0x6723AE17,  0x5270A5C2,  0x0AC42130A, 0x84BE67B2,  0x705CC779,
                       0x5C513D98,  0x0FB36DA2D, 0x22179645,  0x5CE3529D,  0x0D189E1FB, 0x0E85BD489,
                       0x73C8D11F,  0x54B5C196,  0x0B67CB490, 0x2117E4CA,  0x9DE3F994,  0x2F5AA1AA,
                       0x0A7E801FD, 0x0C30D6EAB, 0x1BADDC9C,  0x3453B04A,  0x92A406F9};
    for (int k = 6; k >= 0; k--) {
        v6 = t[k];
        DWORD *p = &input[33];
        v5 = *p;
        result = (16 * v5) ^ (*input >> 3);
        input[34] -= (((*(DWORD *)(a3 + 4LL * ((34 ^ (unsigned __int8)(v6 >> 2)) & 3)) ^ v5) +
                       (*input ^ v6)) ^
                      (((4 * *input) ^ (v5 >> 5)) + result));
        for (v8 = 33LL; v8 > -1; v8--) {
            v5 = v8 == 0 ? (v6 - 1640531527 == 0 ? '}' : input[34]) : *(--p);
            input[v8] -=
                (((((v5 >> 5) ^ (4 * input[v8 + 1])) + ((16 * v5) ^ (input[v8 + 1] >> 3))) ^
                 ((*(DWORD *)(a3 + 4LL * (((unsigned __int8)v8 ^ (unsigned __int8)(v6 >> 2)) & 3)) ^
                   v5) +
                  (input[v8 + 1] ^ v6)));
        }
    }
    for (int i = 0; i < 35; i++) {
        printf("%c", (char)input[i]);
    }
    return 0;
}
```

```
D: > Program > C++ > Hello World > .\main
hgame{l00ks_1ike_y0u_f0Und_th3_t34}
```

搞定！

Re2 helloRe

查壳，没有，直接 IDA64，吓一跳。一堆 C++函数。但是稍微整理一下以后发现其实就是标准的输入输出，IDA 没识别罢了，直接略过。然后看到 sub_140001290(200i64)。打开，发现应该是标准库，结合交叉引用发现还有一处 sub_140001290(50)的，推测应该是 sleep。

核心代码就这里：

```
34  do
35  {
36    v10 = Block;
37    if ( v8 >= 0x10 )
38      v10 = v9;
39    if ( (*((_BYTE *)v10 + v3) ^ (unsigned __int8)sub_140001430()) != byte_140003480[v3] )
40      goto LABEL_13;
41    ++v3;
42  }
```

进一步打开 sub_140001430()，发现就是 byte_140005044--，查看 byte_140005044，初值为 0xFF，推测函数就是返回一个从 0xFF 递减的序列。

写解密脚本：(数据从 byte_140003480 中取得)

```
1  raw = [
2      0x97, 0x99, 0x9C, 0x91, 0x9E, 0x81, 0x91, 0x9D, 0x9B, 0x9A, 0x9A, 0xAB,
3      0x81, 0x97, 0xAE, 0x80, 0x83, 0x8F, 0x94, 0x89, 0x99, 0x97
4  ]
5  print(''.join(chr(raw[i] ^ (0xFF - i)) for i in range(len(raw))))
6
```

```
问题 2   输出   调试控制台   终端

PowerShell 7.1.1
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

D: > Program > Python > python -u "d:\Program\Python\Hello World\main.py"
hgame{hello_re_player}
D: > Program > Python >
```

Re3 pypy

打开一看，dis 跑不掉，直接人肉反编译：

```python
raw_flag = input('give me your flag:\n')
cipher = list(raw_flag[6:-1])
length = len(cipher)
for i in range(length // 2):
    cipher[2 * i], cipher[2 * i + 1] = cipher[2 * i + 1], cipher[2 * i]
res = []
for i in range(length):
    res.append(ord(cipher[i]) ^ i)
res = bytes(res).hex()
print('your flag: ' + res)

# your flag: 30466633346f59213b4139794520572b45514d61583151576638643a
```

然后逆向。逻辑很清晰的，没啥问题，一次搞定：

```python
res = bytes([
    0x30, 0x46, 0x66, 0x33, 0x34, 0x6f, 0x59, 0x21, 0x3b, 0x41, 0x39, 0x79,
    0x45, 0x20, 0x57, 0x2b, 0x45, 0x51, 0x4d, 0x61, 0x58, 0x31, 0x51, 0x57,
    0x66, 0x38, 0x64, 0x3a
])
cipher = []
for i in range(len(res)):
    cipher.append(res[i] ^ i)
length = len(cipher)
for i in range(length // 2):
    cipher[2 * i], cipher[2 * i + 1] = cipher[2 * i + 1], cipher[2 * i]
print('hgame{' + ''.join(chr(i) for i in cipher) + '}')
```

```
问题 2   输出   调试控制台   终端

PowerShell 7.1.1
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

 D: > Program > Python  python -u "d:\Program\Python\Hello World\HGAME\Week1\pypy_decypt.py"
hgame{G00dj0&_H3r3-I$Y@Ur_$L@G!~!~}
 D: > Program > Python
```

PWN 部分

PWN1 whitegive

看 C 代码里直接有 sh 了，于是把二进制文件拖入 IDA64，得到 paSsw0rd 地址：

```
.rodata:0000000000402012      aPassw0rd       db 'paSsw0rd',0
```

将 402012 转成 10 进制得 4202514，然后输入密码：

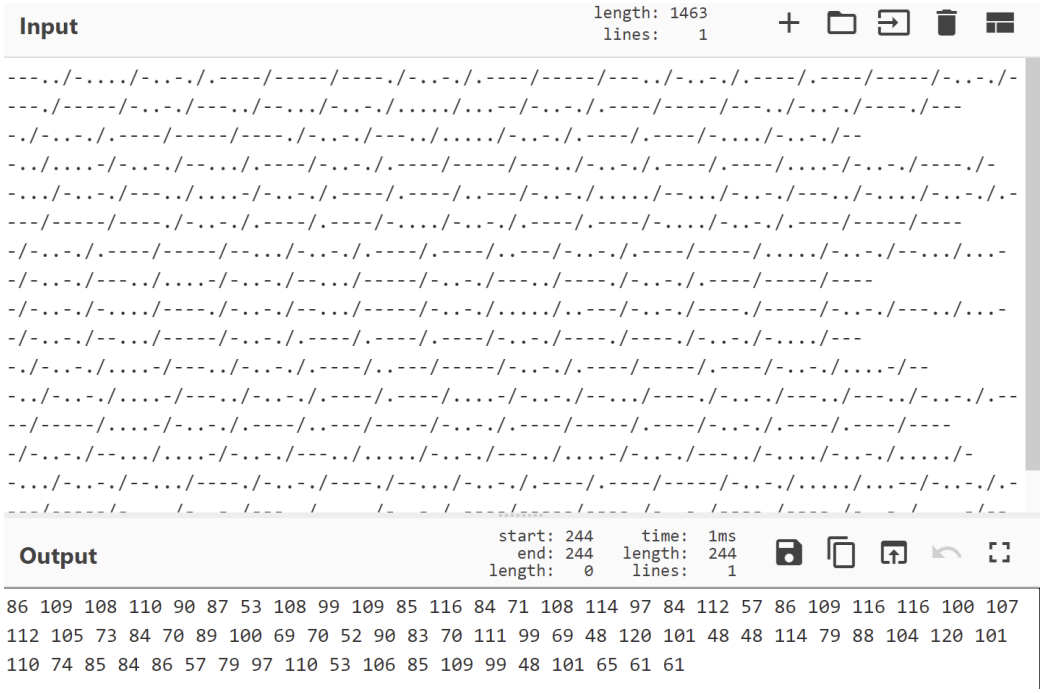
```
~ nc 182.92.108.71 30210
password:4202514
you are right!
ls
bin
dev
flag
lib
lib32
lib64
usr
whitegive
cat flag
hgame{W3lCOme_t0_Hg4m3_2222Z222zO2l}
```

完事。

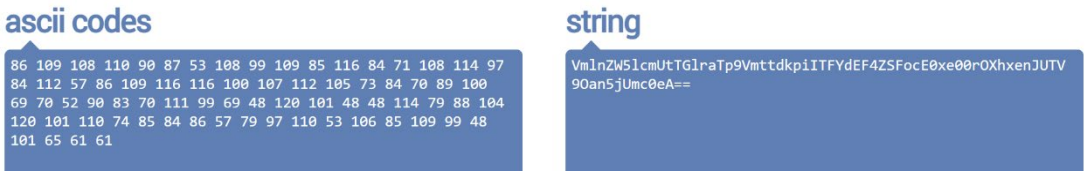
Crypto 部分

Crypto1 まひと

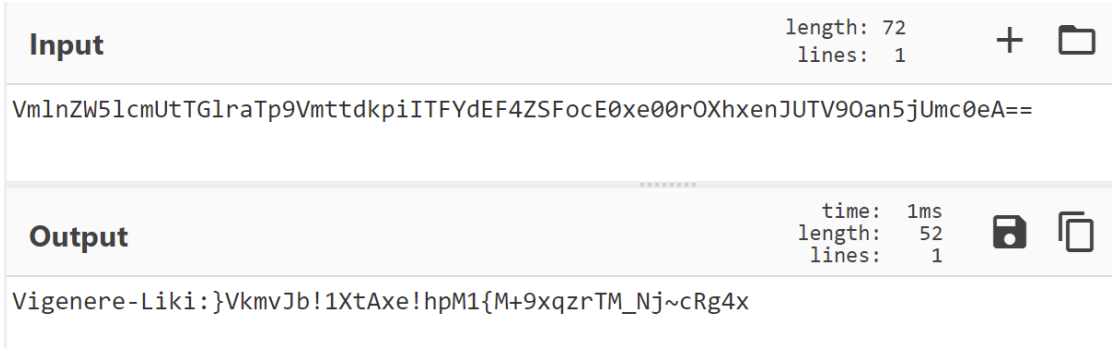
打开，直觉判断摩尔斯电码，解密：



然后去 ascii2string：



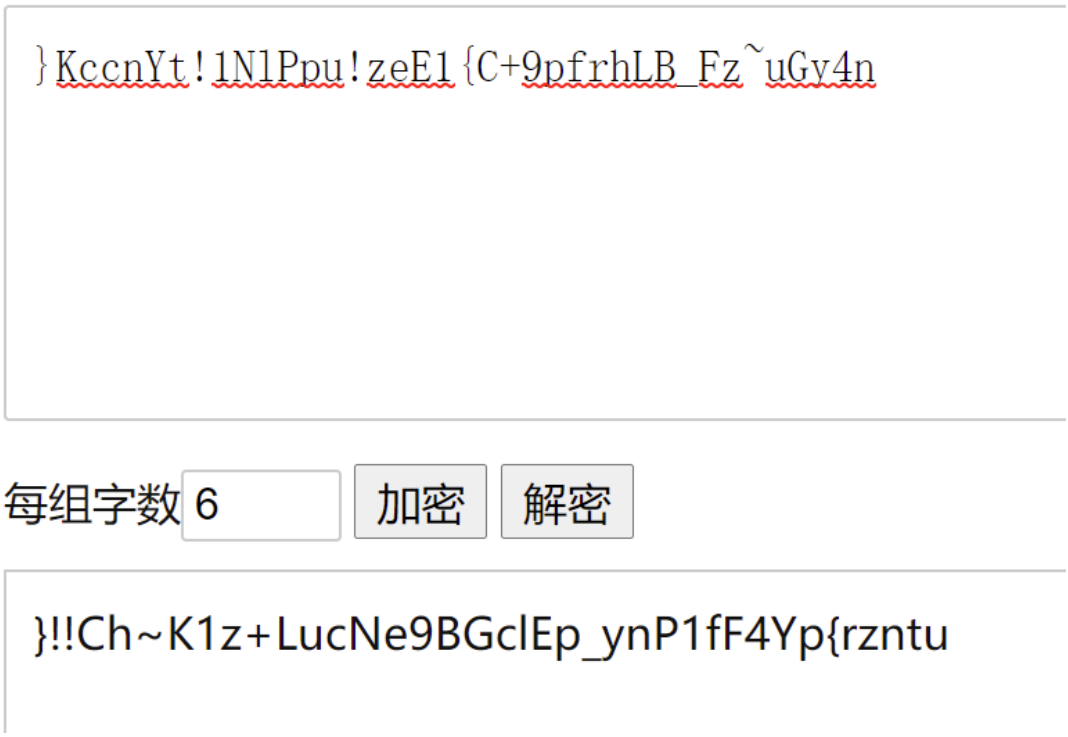
这 base64 再明显不过了：



根据提示找一个解密 Vigenere 的网站：https://www.dcode.fr/vigenere-cipher

先试试看密钥 Liki：

发现大括号的位置显然不对。想到栅栏密码（只有这个可以改变字符顺序了……）：



逐一尝试后发现参数为 6 的时候有点像，最后有 5 个字母，倒叙以后格式就对了

最后想想没啥办法，再去 Vigenere 一次。这次尝试结果中带 hgame 的：

很明显，密钥为 NN 时就是结果。

（纯粹是运气好，第一次的密钥 LIKI 就是碰运气的……）

Crypto2 对称之美

拿到题目，是 16 位 Key 的循环异或加密。考虑到 cipher 有 400 多位，怀疑 FLAG 是 Base 加密的。思考几天没有什么想法，去问了问学长，得知是一段英文，豁然开朗。

先把 cipher 转成二进制流：

```
open('data.bin', 'wb').write(bytearray(cipher))
```

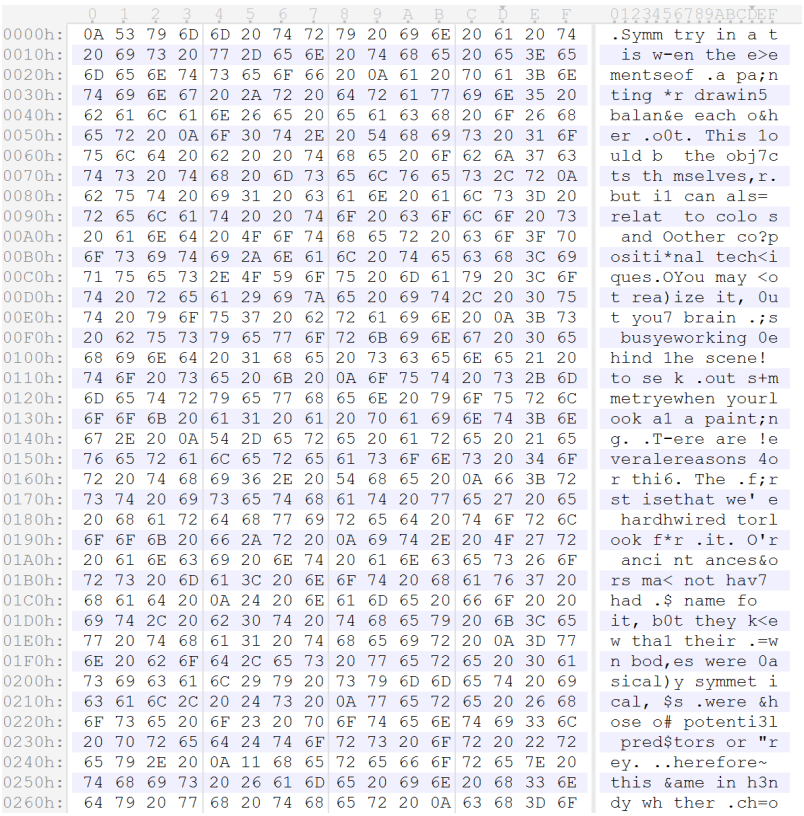
然后直接丢进 xortool：

```
~ /home/gyj1109/.local/bin/xortool -l 16 -c ' ' data.bin
2 possible key(s) of length 16:
vGbNqroSdHlMVT0g
vGbNqroS!HlMVT0g
Found 2 plaintexts with 95%+ valid characters
See files filename-key.csv, filename-char_used-perc_valid.csv
```

打开两个 out 都看了看，显然第一个更好，但还有错：

```
Symm try in a t is w-en the e>ementseof
a pa;nting *r drawin5 balan&e each o&her
o0t. This 1ould b  the obj7cts th mselves,r
but i1 can als= relat  to colo s and Oother co?positi*nal tech<iques.OYou may <ot rea)ize it, 0ut you7
brain
;s busyeworking 0ehind 1he scene! to se k
out s+mmetryewhen yourlook a1 a paint;ng.
T-ere are !everalereasons 4or thi6. The
f;rst isethat we' e hardhwired torlook f*r
it. O'r anci nt ances&ors ma< not hav7 had
$ name fo  it, b0t they k<ew tha1 their
=wn bod,es were 0asical)y symmet ical, $s
were &hose o# potenti3l pred$tors or "rey.
□herefore~ this &ame in h3ndy wh ther
ch=osing $ mate, c3tchingedinner o
avoi!ing bein5 on th  menu ofra snar)ing,
hu<gry pa&k of wol$es or 'ears!
Ta9e a lo*k at you  face ,n the mi ror
a+d imagin7 a lin  straigh& down 1he
midd>e. Youbll see b=th sid s of you
faceeare pret&y symm trical. □his ise
known a! bilat ral symm7try an! it's
w:ere bo1h sides 7ither 6ide of t:is
di3iding li<e appe$r more o  less 1he same.XSo her  is the 4lag:
-game{X0r
i5-a_u□3fU1+4ndvfUNny_□1pH3r}
```

用 010Editor 打开：



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	0A	53	79	6D	6D	20	74	72	79	20	69	6E	20	61	20	74	.Symm try in a t
0010h:	20	69	73	20	77	2D	65	6E	20	74	68	65	20	65	3E	65	is w-en the e>e
0020h:	6D	65	6E	74	73	65	6F	66	20	0A	61	20	70	61	3B	6E	mentseof .a pa;n
0030h:	74	69	6E	67	20	2A	72	20	64	72	61	77	69	6E	35	20	ting *r drawin5
0040h:	62	61	6C	61	6E	26	65	20	65	61	63	68	20	6F	26	68	balan&e each o&h
0050h:	65	72	20	0A	6F	30	74	2E	20	54	68	69	73	20	31	6F	er .o0t. This 1o
0060h:	75	6C	64	20	62	20	20	74	68	65	20	6F	62	6A	37	63	uld b the obj7c
0070h:	74	73	20	74	68	20	6D	73	65	6C	76	65	73	2C	72	0A	ts th mselves,r.
0080h:	62	75	74	20	69	31	20	63	61	6E	20	61	6C	73	3D	20	but i1 can als=
0090h:	72	65	6C	61	74	20	20	74	6F	20	63	6F	6C	6F	20	73	relat to colo s
00A0h:	20	61	6E	64	20	4F	6F	74	68	65	72	20	63	6F	3F	70	and Oother co?p
00B0h:	6F	73	69	74	69	2A	6E	61	6C	20	74	65	63	68	3C	69	positi*nal tech<i
00C0h:	71	75	65	73	2E	4F	59	6F	75	20	6D	61	79	20	3C	6F	ques.OYou may <o
00D0h:	74	20	72	65	61	29	69	7A	65	20	69	74	2C	20	30	75	t rea)ize it, 0u
00E0h:	74	20	79	6F	75	37	20	62	72	61	69	6E	20	0A	3B	73	t you7 brain .;s
00F0h:	20	62	75	73	79	65	77	6F	72	6B	69	6E	67	20	30	65	busyeworking 0e
0100h:	68	69	6E	64	20	31	68	65	20	73	63	65	6E	65	21	20	hind 1he scene!
0110h:	74	6F	20	73	65	20	6B	20	0A	6F	75	74	20	73	2B	6D	to se k .out s+m
0120h:	6D	65	74	72	79	65	77	68	65	6E	20	79	6F	75	72	6C	metryewhen yourl
0130h:	6F	6F	6B	20	61	31	20	61	20	70	61	69	6E	74	3B	6E	ook a1 a paint;n
0140h:	67	2E	20	0A	54	2D	65	72	65	20	61	72	65	20	21	65	g. .T-ere are !e
0150h:	76	65	72	61	6C	65	72	65	61	73	6F	6E	73	20	34	6F	veralereasons 4o
0160h:	72	20	74	68	69	36	2E	20	54	68	65	20	0A	66	3B	72	r thi6. The .f;r
0170h:	73	74	20	69	73	65	74	68	61	74	20	77	65	27	20	65	st isethat we' e
0180h:	20	68	61	72	64	68	77	69	72	65	64	20	74	6F	72	6C	hardhwired torl
0190h:	6F	6F	6B	20	66	2A	72	20	0A	69	74	2E	20	4F	27	72	ook f*r .it. O'r
01A0h:	20	61	6E	63	69	20	6E	74	20	61	6E	63	65	73	26	6F	anci nt ances&o
01B0h:	72	73	20	6D	61	3C	20	6E	6F	74	20	68	61	76	37	20	rs ma< not hav7
01C0h:	68	61	64	20	0A	24	20	6E	61	6D	65	20	66	6F	20	20	had .$ name fo
01D0h:	69	74	2C	20	62	30	74	20	74	68	65	79	20	6B	3C	65	it, b0t they k<e
01E0h:	77	20	74	68	61	31	20	74	68	65	69	72	20	0A	3D	77	w tha1 their .=w
01F0h:	6E	20	62	6F	64	2C	65	73	20	77	65	72	65	20	30	61	n bod,es were 0a
0200h:	73	69	63	61	6C	29	79	20	73	79	6D	6D	65	74	20	69	sical)y symmet i
0210h:	63	61	6C	2C	20	24	73	20	0A	77	65	72	65	20	26	68	cal, $s .were &h
0220h:	6F	73	65	20	6F	23	20	70	6F	74	65	6E	74	69	33	6C	ose o# potenti3l
0230h:	20	70	72	65	64	24	74	6F	72	73	20	6F	72	20	22	72	pred$tors or "r
0240h:	65	79	2E	20	0A	11	68	65	72	65	66	6F	72	65	7E	20	ey. ..herefore~
0250h:	74	68	69	73	20	26	61	6D	65	20	69	6E	20	68	33	6E	this &ame in h3n
0260h:	64	79	20	77	68	20	74	68	65	72	20	0A	63	68	3D	6F	dy wh ther .ch=o

由于刚好 Key 也是 16 位的，于是每一列用的是同一个字符进行异或加密。显然，0x45 位的 balance 和 0x6E 位的 objects 是错的。在 Python 里取到对应字母的密文，和本身应该的字母取异或得到正确的 Key。修改后的 Key 和最后的结果如下：

```
key = 'vGbNq7oSdHlMVTbg'
print(
    bytes([m ^ ord(k)
           for m, k in zip(cipher, itertools.cycle(key))]).decode('ascii'))
```

```
问题 2  输出  调试控制台  终端
D: > Program > Python  python -u "d:\Program\Python\Hello World\main2.py"

Symmetry in art is when the elements of
a painting or drawing balance each other
out. This could be the objects themselves,
but it can also relate to colors and
other compositional techniques.
You may not realize it, but your brain
is busy working behind the scenes to seek
out symmetry when you look at a painting.
There are several reasons for this. The
first is that we're hard-wired to look for
it. Our ancient ancestors may not have had
a name for it, but they knew that their
own bodies were basically symmetrical, as
were those of potential predators or prey.
Therefore, this came in handy whether
choosing a mate, catching dinner or
avoiding being on the menu of a snarling,
hungry pack of wolves or bears!
Take a look at your face in the mirror
and imagine a line straight down the
middle. You'll see both sides of your
face are pretty symmetrical. This is
known as bilateral symmetry and it's
where both sides either side of this
dividing line appear more or less the same.
So here is the flag:
hgame{X0r_i5-a_uS3fU1+4nd$fUNny_C1pH3r}
```

Crypto3 Transformer

Transformer.txt 里貌似是一些加密了的文字。然后猜测 ori 中存放原文，enc 中为密文，格式相同但序号不同。故正则搜索。比如：

这两个应该是一组的，格式完全相同。所以可以知道 r→n，以此类推。

多次检索以后推出 Transformer.txt 的原文：

```
Tqh ufso mnfcyh eaikauh kdkoht qpk aiud zkhc xpkkranc uayfi kfieh 2003, oqh xpkkranc fk "qypth
{hp5d_s0n_szi^3ic&qh11a_}",Dai'o sanyho oa pcc oqh dhpn po oqh hic.

The lift bridge console system has only used password login since 2003, the password is "hgame
{ea5y_f0r_fun^3nd&he11o_}",Don't forget to add the year at the end.
```

话说，我一开始在结尾加了 2003，不对，然后加 2021，对了.....

MISC 部分

MISC1 Base 全家福

签到题。

原文：

R1k0RE1OWldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpVR01ZREtSUlVIQTJET01aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09

先 Base64：

GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQGMZDGMJXIQ======

再 Base32：

6867616D657B57653163306D655F74305F4847344D335F323032317D

最后 Base16：

hgame{We1c0me_t0_HG4M3_2021}

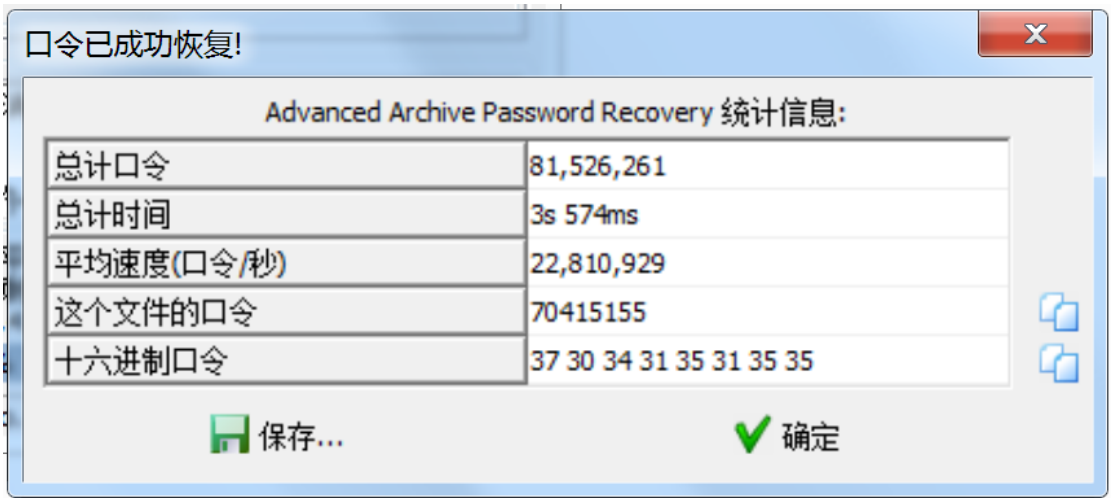
完事。

MISC2 不起眼压缩包的养成的方法

拖入 010Editor，发现结尾提示：

```
...*.ï»¿Password
  is picture ID (
Up to 8 digits)
```

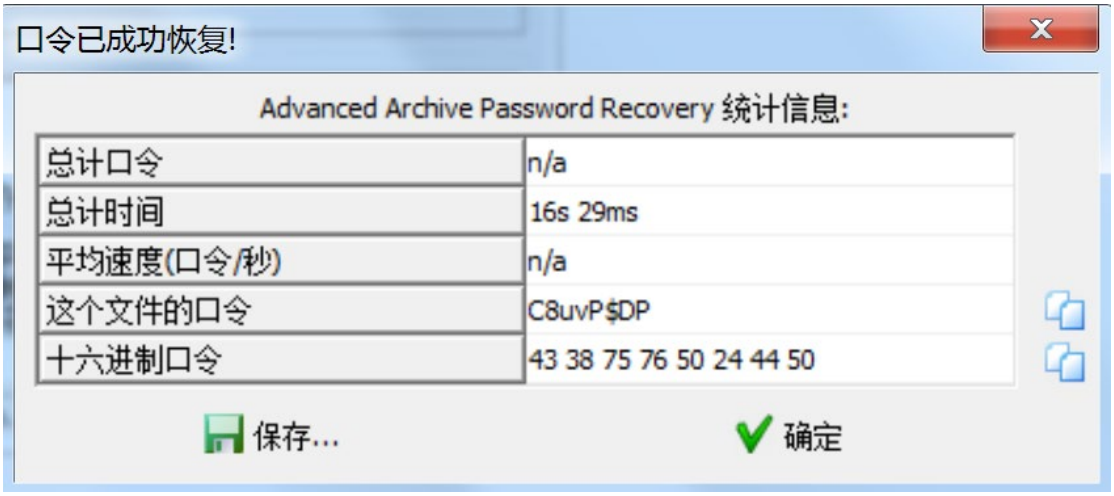
Linux 下用 foremost，分离出 00001230.zip，回到 Windows 打开 ARCHPR，跑字典：



然后看到 plain.zip 和 NO PASSWORD.txt。plain.zip 中也有密码，也有 NO PASSWORD.txt，

查看 CRC32 后发现是相同文件，故明文攻击。

根据 NO PASSWORD.txt 提示，构造一个仅存储的压缩包，然后攻击：



得到 flag.zip。这下实在找不到提示了，考虑伪加密，尝试修改加密位：

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000h: 50 4B 03 04 14 00 00 00 00 00 F3 AB 3D 52 43 97
```

改完以后拿到 flag.txt，在 VSCode 中打开：

```
&#x68;&#x67;&#x61;&#x6D;&#x65;&#x7B;&#x32;&#x49;&#x50;&#x5F;&#x69;&#x73;&#x5F;&#x55;&#x73;&#x65;&#x66;&#x75;&#x31;&#x5F;&#x61;&#x6E;&#x64;&#x5F;&#x4D;&#x65;&#x39;&#x75;&#x6D;&#x69;&#x5F;&#x69;&#x35;&#x5F;&#x57;&#x30;&#x72;&#x31;&#x64;&#x7D;
```

稍微整理一下格式，放到 Python 里转 Ascii：

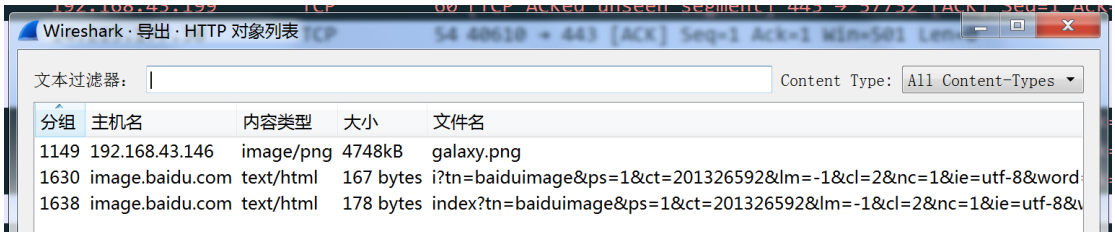
```
print(
    bytes([
        0x68, 0x67, 0x61, 0x6D, 0x65, 0x7B, 0x32, 0x49, 0x50, 0x5F, 0x69, 0x73,
        0x5F, 0x55, 0x73, 0x65, 0x66, 0x75, 0x31, 0x5F, 0x61, 0x6E, 0x64, 0x5F,
        0x4D, 0x65, 0x39, 0x75, 0x6D, 0x69, 0x5F, 0x69, 0x35, 0x5F, 0x57, 0x30,
        0x72, 0x31, 0x64, 0x7D
    ]).decode('ascii'))
```

```
问题 2  输出  调试控制台  终端

PowerShell 7.1.1
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

 D: > Program > Python  python -u "d:\Program\Python\Hello World\main.py"
hgame{2IP_is_Usefu1_and_Me9umi_i5_W0r1d}
 D: > Program > Python
```

完工。

MISC3 Galaxy

流量分析直接导出 HTTP 对象



得到 galaxy.png：

放进 010Editor 改一下高度：

hgame{Wh4t_A_W0nderfu1_Wa11paper}

Flag 到手，注意里面的小写字母 `l` 其实是数字 `1`。

MISC4 Word RE:MASTER

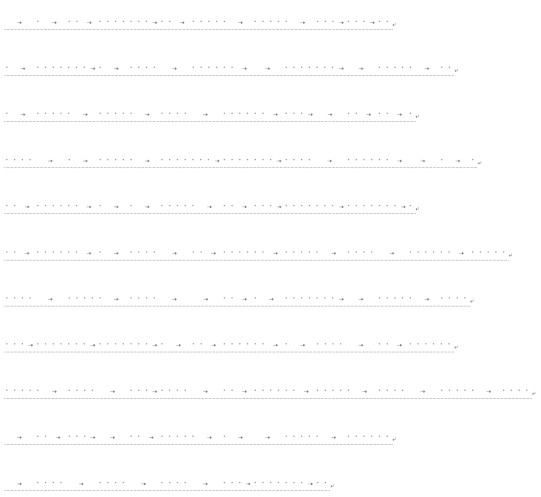
解压得到两个 docx 文件，一个加密一个不加密，显然先从不加密的入手（文件名和文件描述都提示了）。

直接丢去 binwalk -e，发现藏着个 password.xml，打开：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<password>+++++ +++[- >++++ ++++< ]>+++ +.<++ +[->+ ++<]> ++.<+ ++[-> +
++<] >+.<+ ++[-> ---<] >-.++ ++++. <+++[ ->--- <]>-. +++.+ .++++ ++++.
<+++[ ->--- <]>-- ----. +.--- --..+ .++++ +++++ .<+++ [->-- -<]>-
----- .<</password>
```

再明显不过的 Brainfuck 了。网上找一个在线编译器得到结果“DOYOUKNOWHIDDEN?”，成功打开第二个 Word 文档。

显示隐藏字符的情况下一目了然：



复制出来，弄进一个 txt 文件。

根据图片的提示和内容，SNOW 隐写。直接解密：

```
~ > Downloads .\SNOW.EXE -C .\a.txt
hgame{Cha11en9e_Whit3_P4ND0R4_P4R4D0XXX}
```

Done!