CTF学习笔记——[HCTF 2018]admin

原创

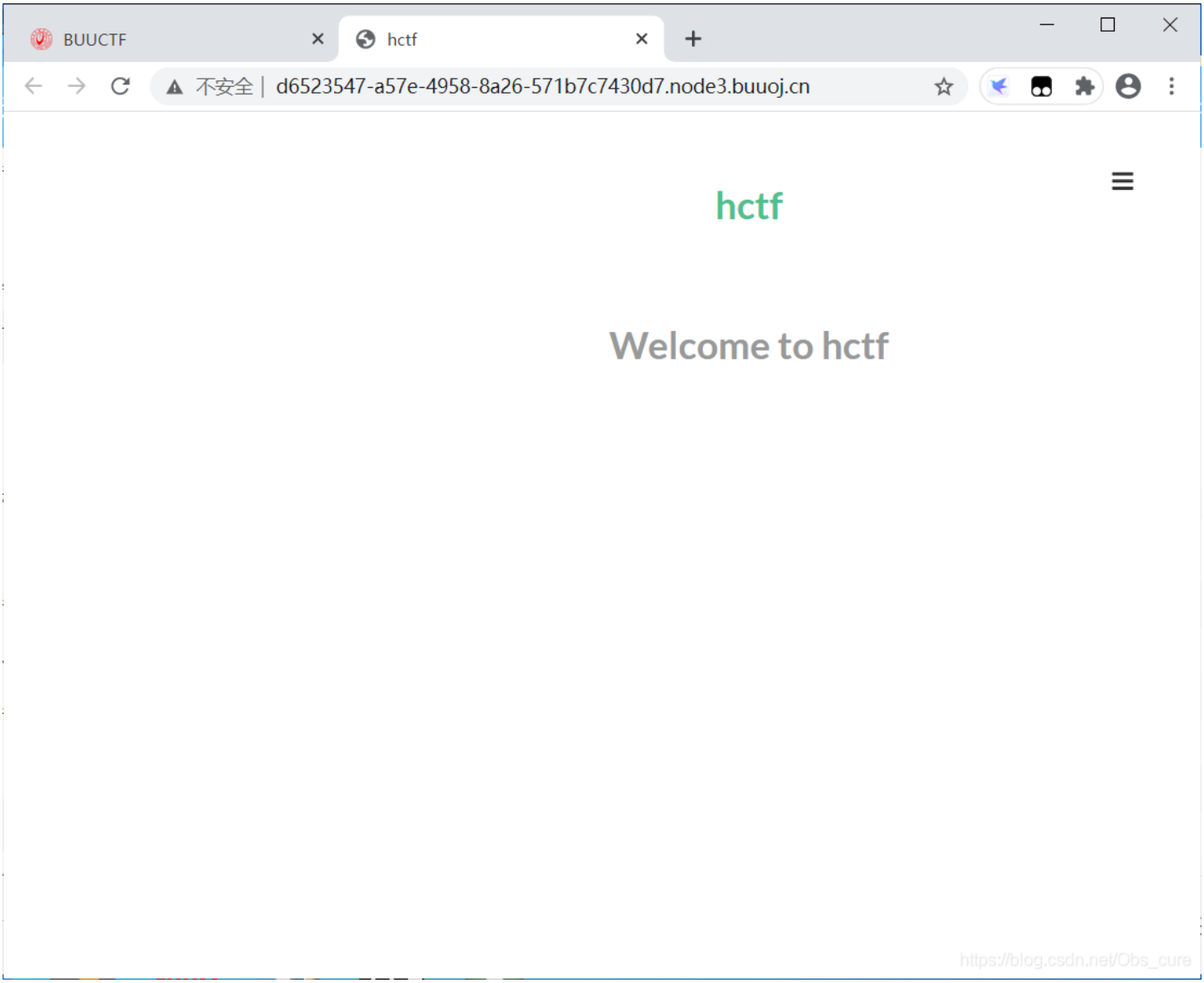
Obs_cure 于 2021-02-15 01:21:26 发布 426 收藏 3

文章标签： 网络安全

本文链接：https://blog.csdn.net/Obs_cure/article/details/113805070

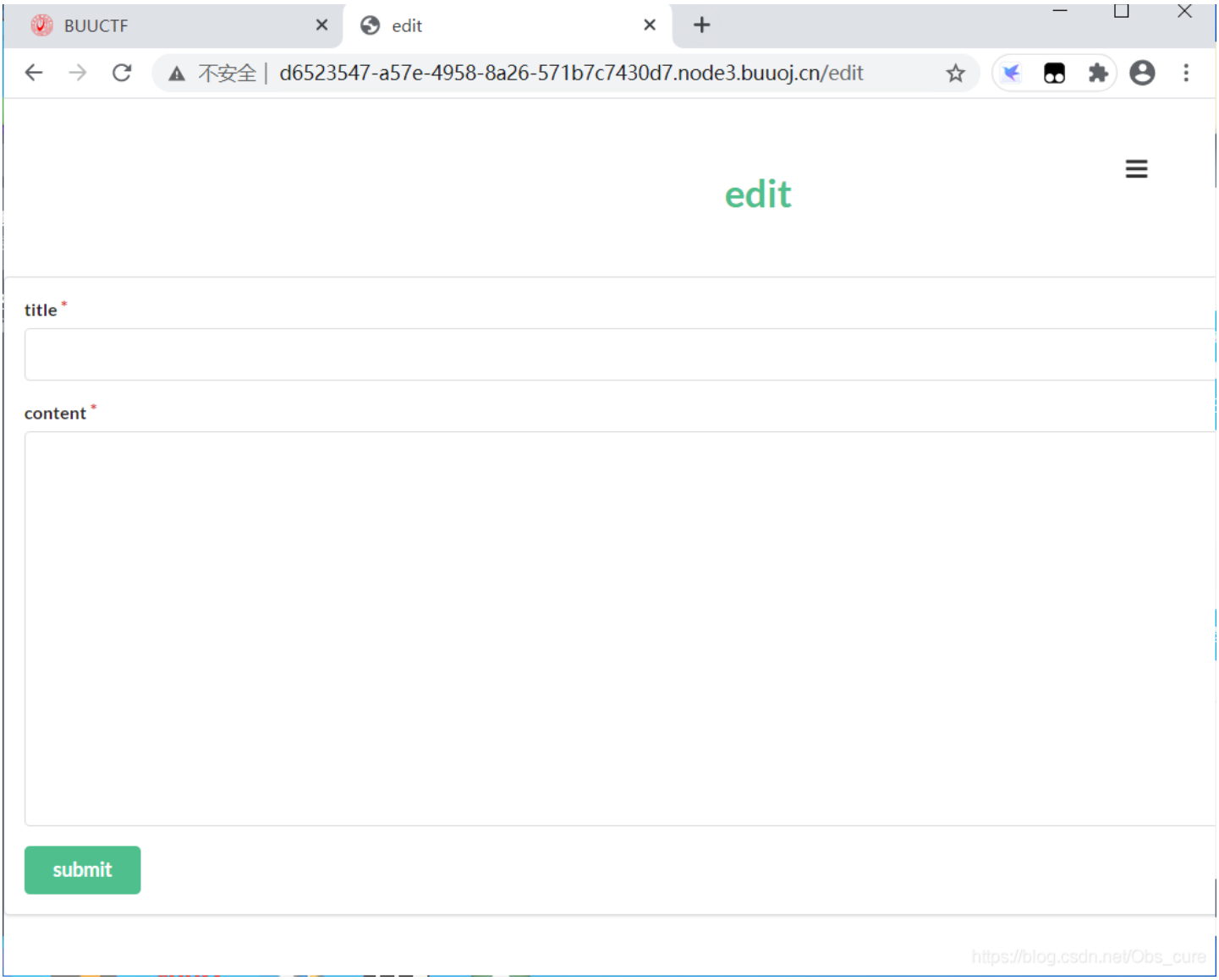
一、[HCTF 2018]admin

1.题目

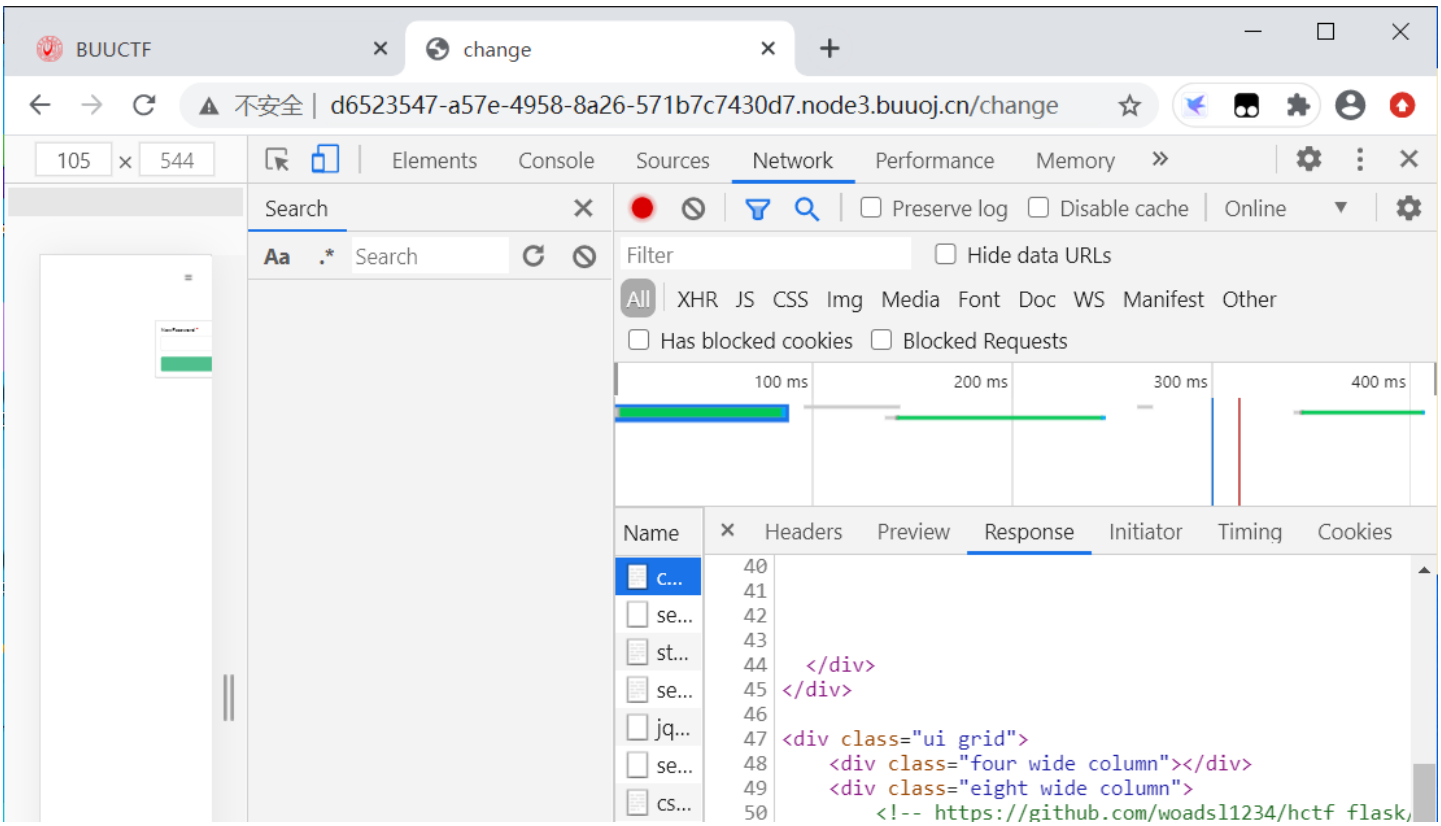


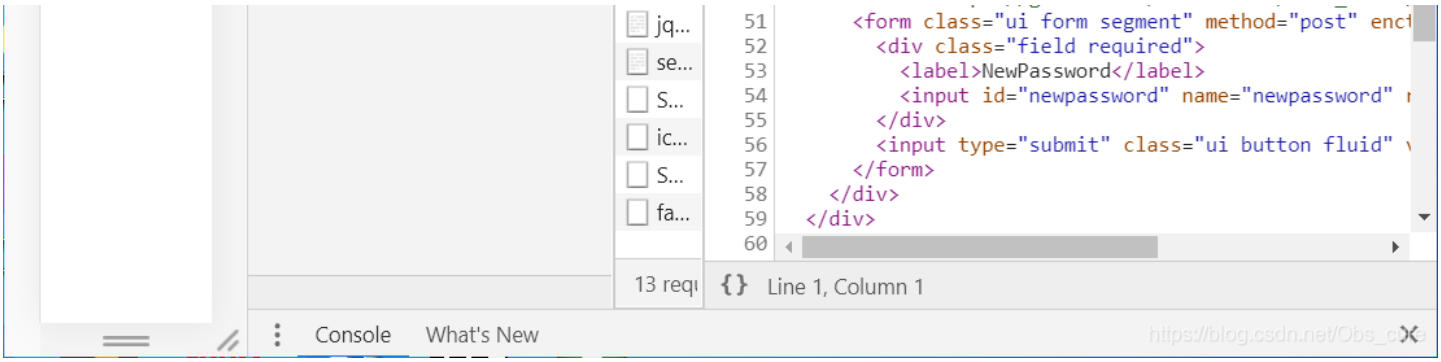
2.解题步骤

这个靶场还是很完整的，注册登陆一应俱全，还有留言板。注册和登陆部分尝试了一下，大部分我认知中的注入都被过滤了。于是老老实实注册了个账号。发现里面有个留言板。这不就是师傅们经常攻击的地方吗~

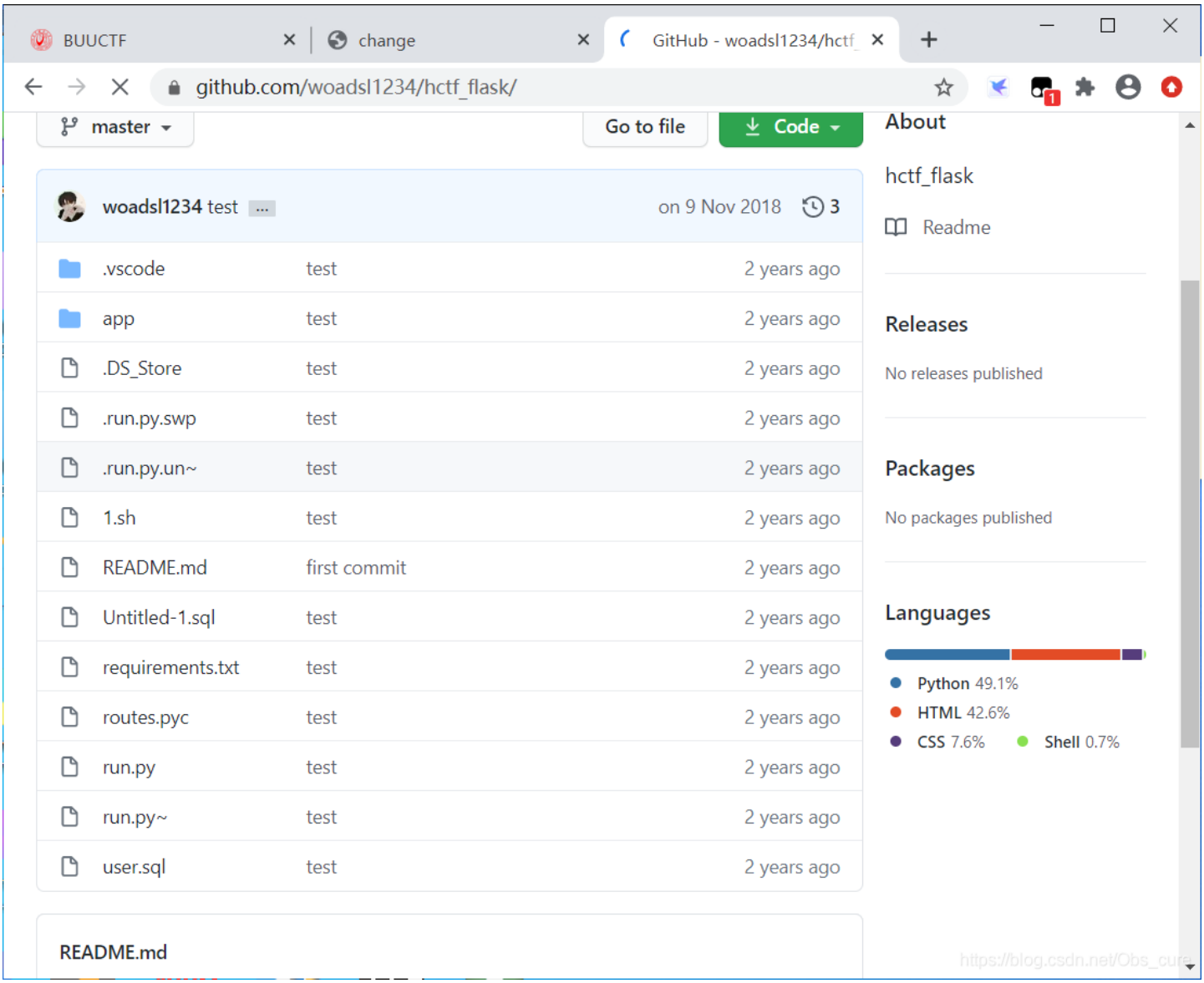


随便注入的两下没什么反应，看看源码呢~





嗯？这个注释？

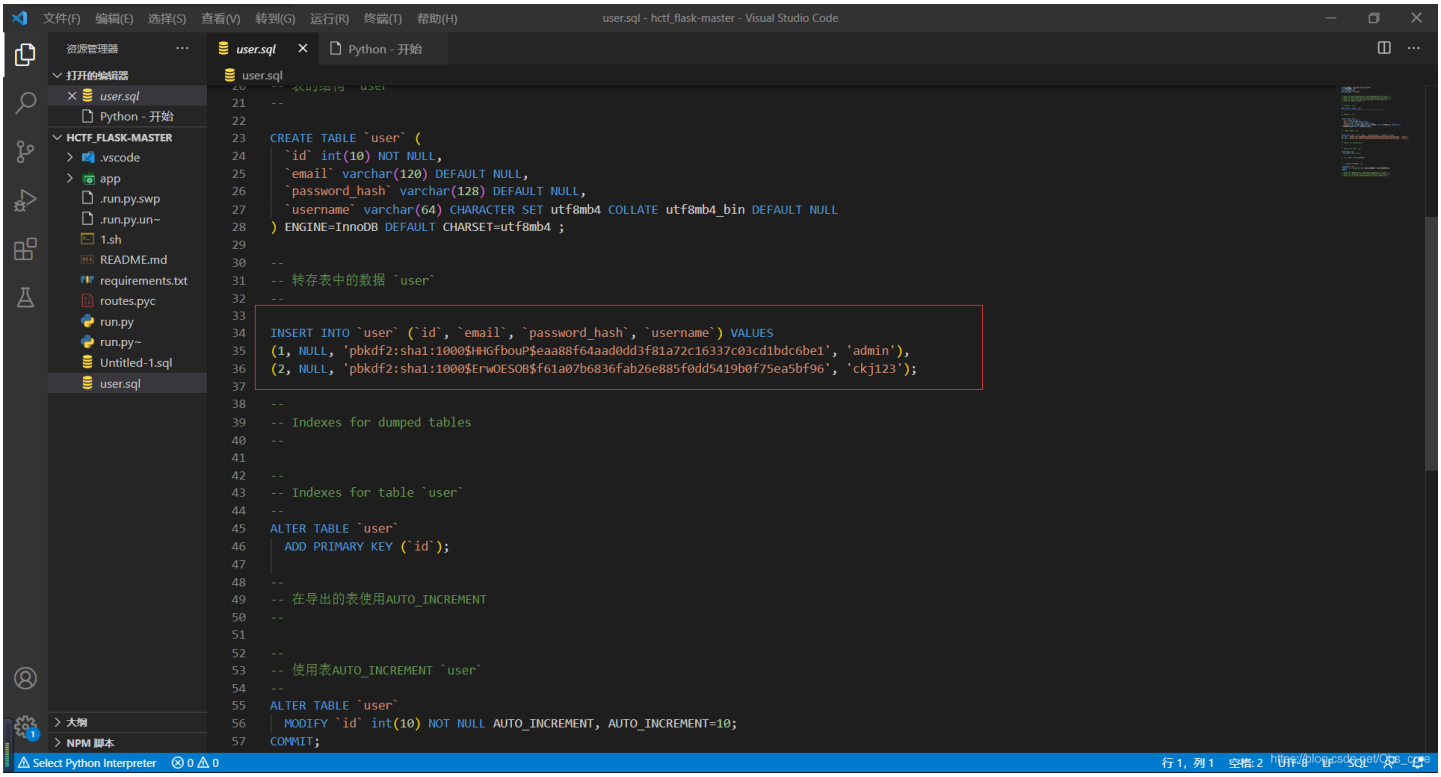


虽然能猜到这个是网站的源码，但由于没有网页的开发经验，所以很遗憾还是没看出什么门道~看WriteUp！
说实话看完之后有点被震撼到。跟着师傅们把三种做法都过一遍~
这道题的突破口首先是github上的提示flask。

Flask是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ，模板引擎则使用 Jinja2 。Flask使用 BSD 授权。[百度百科]

这样来看，这是一道模板注入类型的题目。

然后进行源码审计。

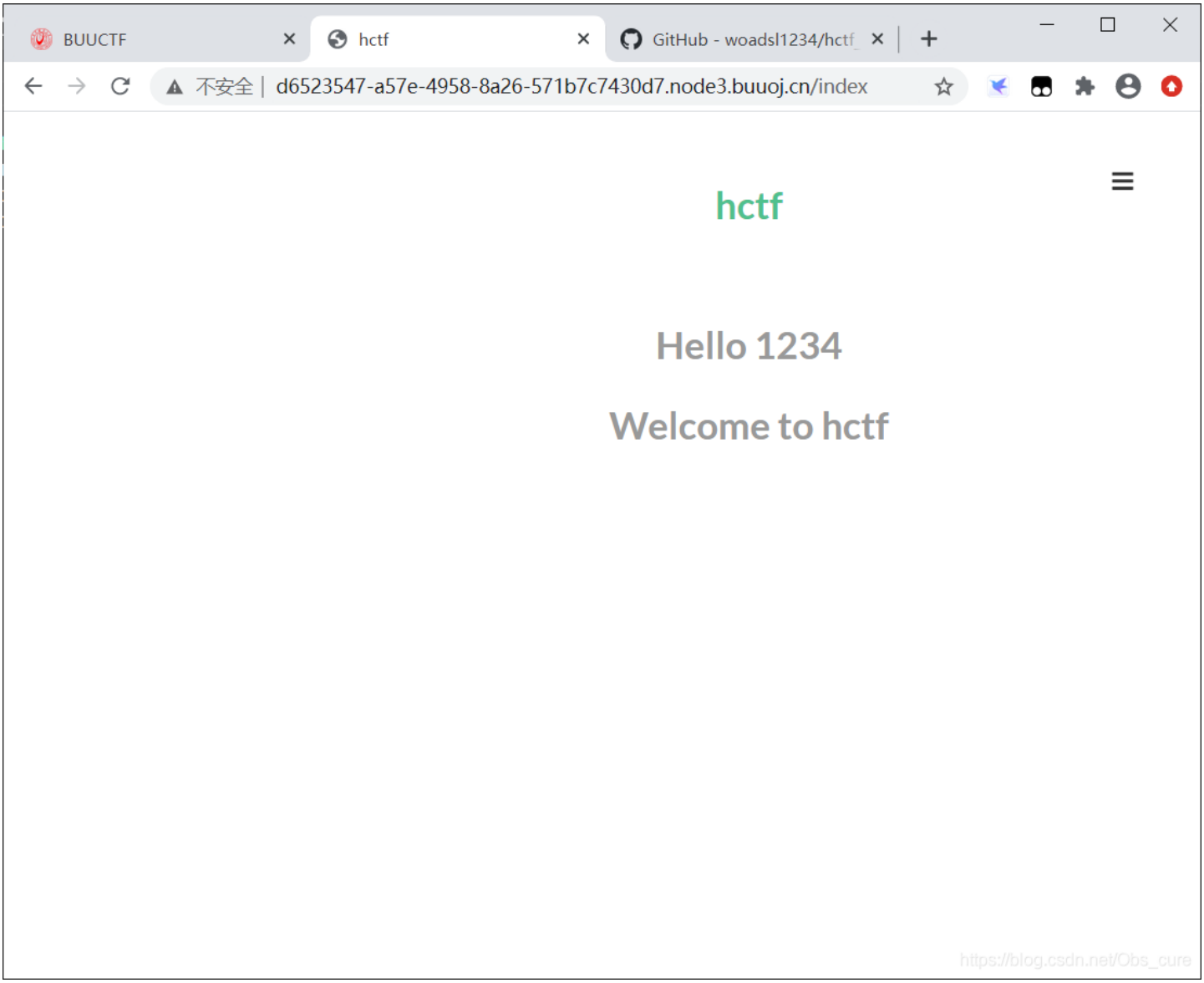


首先是发现库中有两个账号，可惜密码被加密了~
接下来就是几种登陆admin的方式

（1）flask session伪造

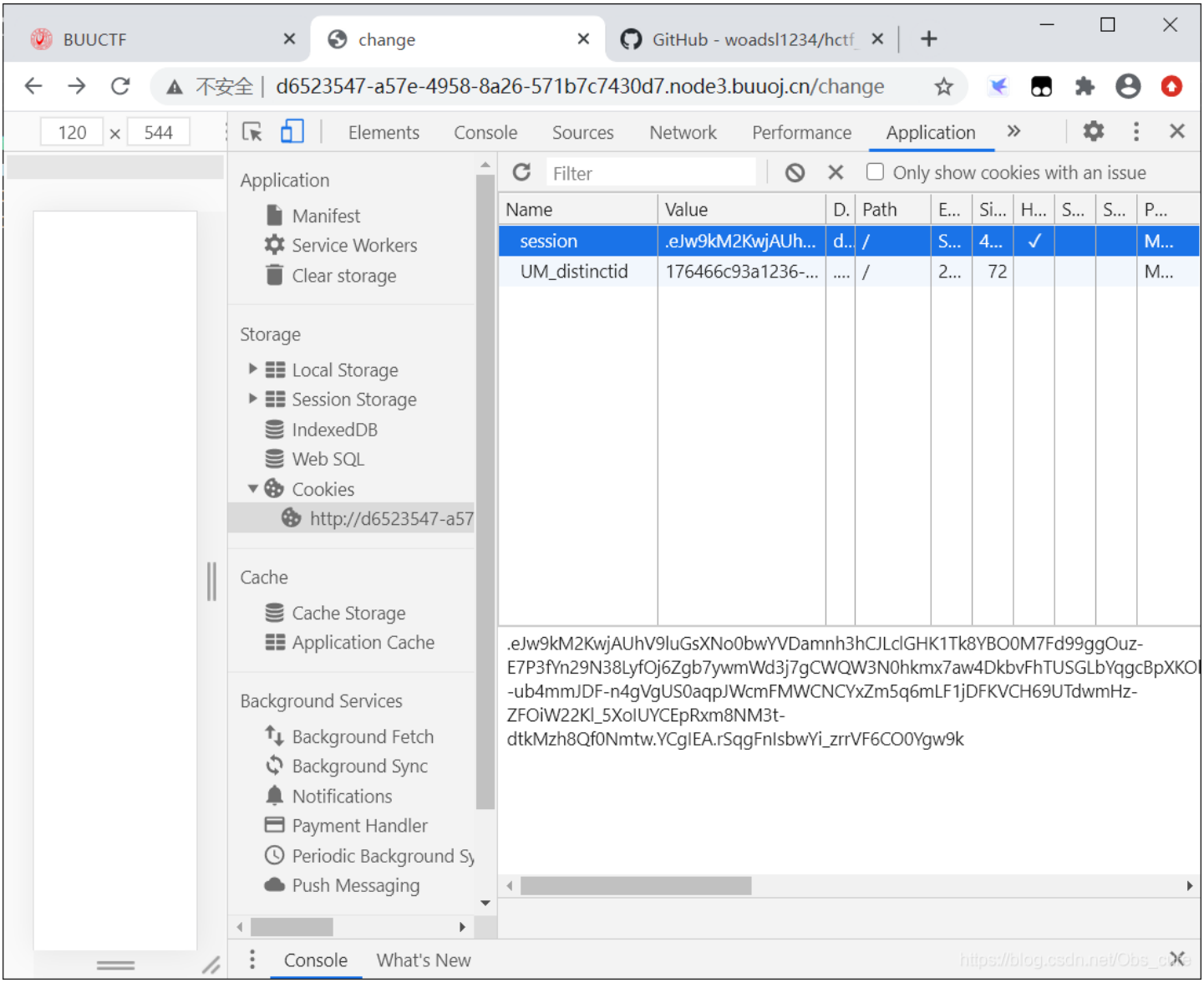
什么是Session?
在计算机中，尤其是在网络应用中，称为“会话控制”。Session对象存储特定用户会话所需的属性及配置信息。这样，当用户在应用程序的Web页之间跳转时，存储在Session对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。[百度百科]

在测试靶场网页的时候就发现了，比如登陆之后再刷新网页，他依旧会显示登陆者的名字



这个功能是靠session实现的。把账号的信息临时储存在我的电脑上（客户端）。只有在关闭浏览器才会销毁session信息。这是一种方便的机制。这里有个大神的博客讲的很好，指路一下~

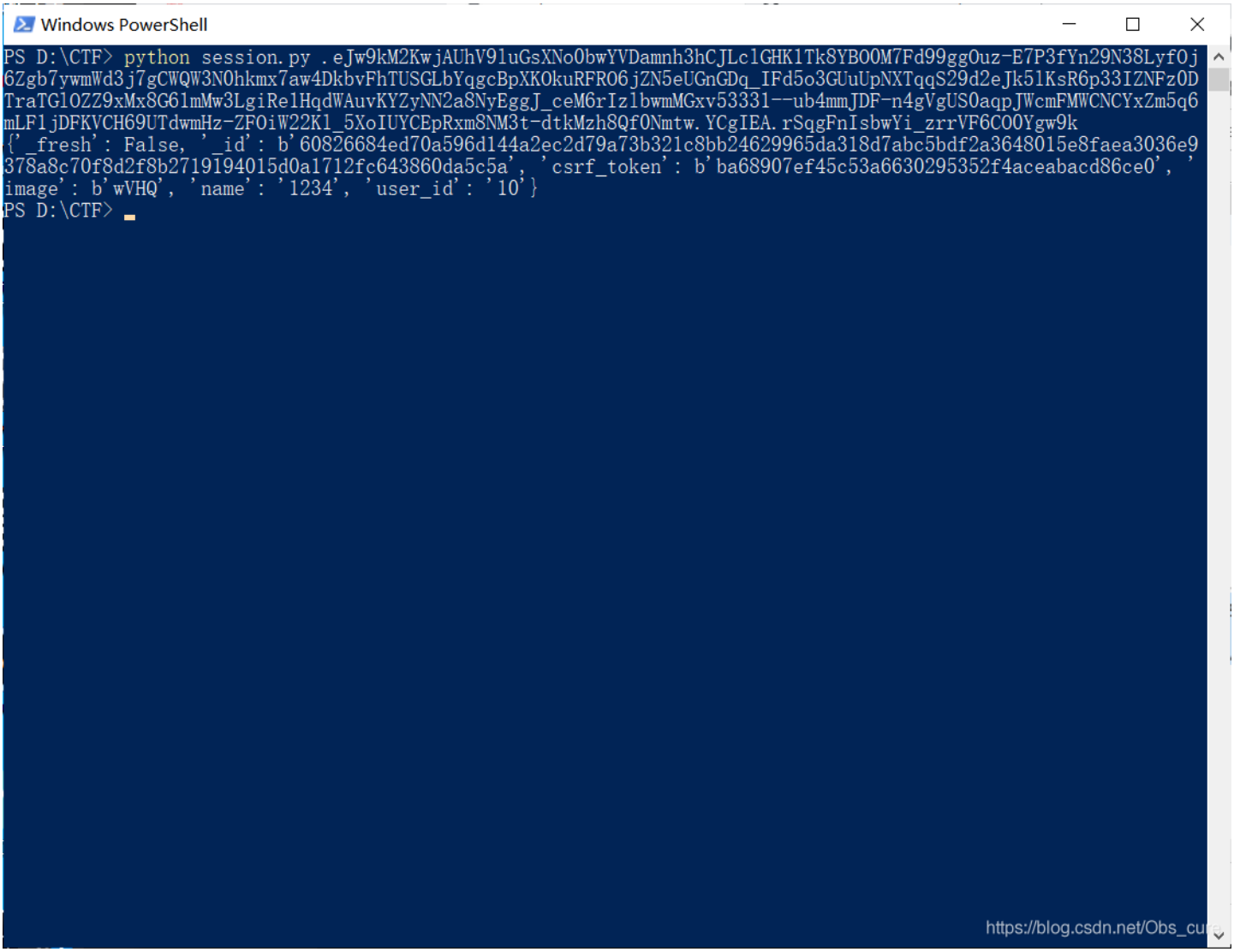
session要求浏览器必须支持cookie功能，那么对于session的伪造，是否和cookie伪造类似呢？
这道题的session在这个位置：



接下来就是session伪造的部分啦！从上面的图看，session明显是加密过的。那么我们要去找flask的加密算法是什么~在看过师傅们的WriteUp之后发现，flask的session加密是一个算法。这里直接贴师傅的源码了

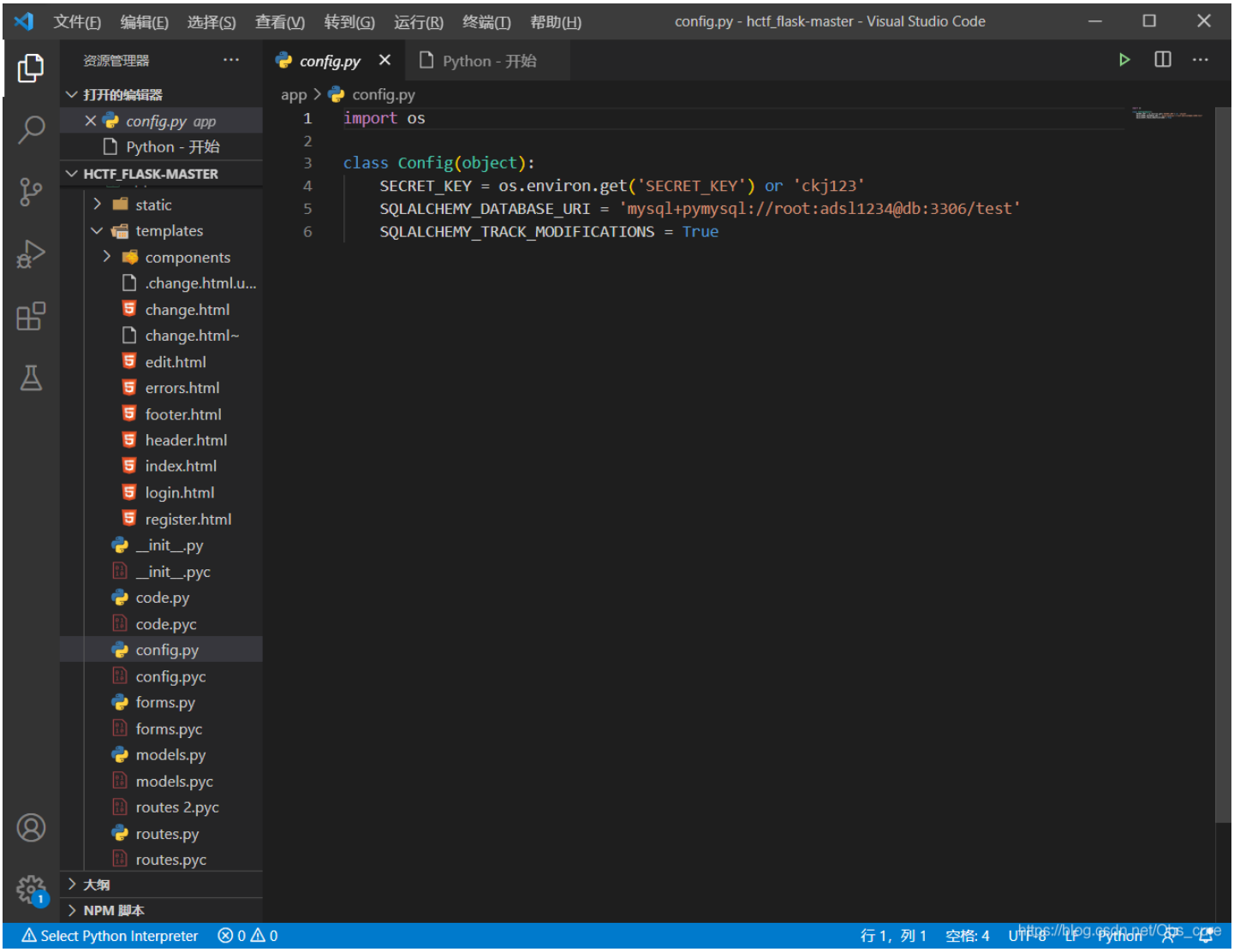
```python
#!/usr/bin/env python3
import sys
import zlib
from base64 import b64decode
from flask.sessions import session_json_serializer
from itsdangerous import base64_decode

def decryption(payload):
    payload, sig = payload.rsplit(b'.', 1)
    payload, timestamp = payload.rsplit(b'.', 1)

    decompress = False
    if payload.startswith(b'.'):
        payload = payload[1:]
        decompress = True

    try:
        payload = base64_decode(payload)
    except Exception as e:
        raise Exception('Could not base64 decode the payload because of '
                        'an exception')

    if decompress:
        try:
            payload = zlib.decompress(payload)
        except Exception as e:
            raise Exception('Could not zlib decompress the payload before '
                            'decoding the payload')

    return session_json_serializer.loads(payload)

if __name__ == '__main__':
    print(decryption(sys.argv[1].encode()))
```

我们可以通过解密来查看session的格式



基于这个格式，我们可以尝试伪造一个admin的了~

需要注意的是，session加密过程中有一个SECRET_KEY，需要我们知晓。这个东西一般放在cofig之类的文件里。

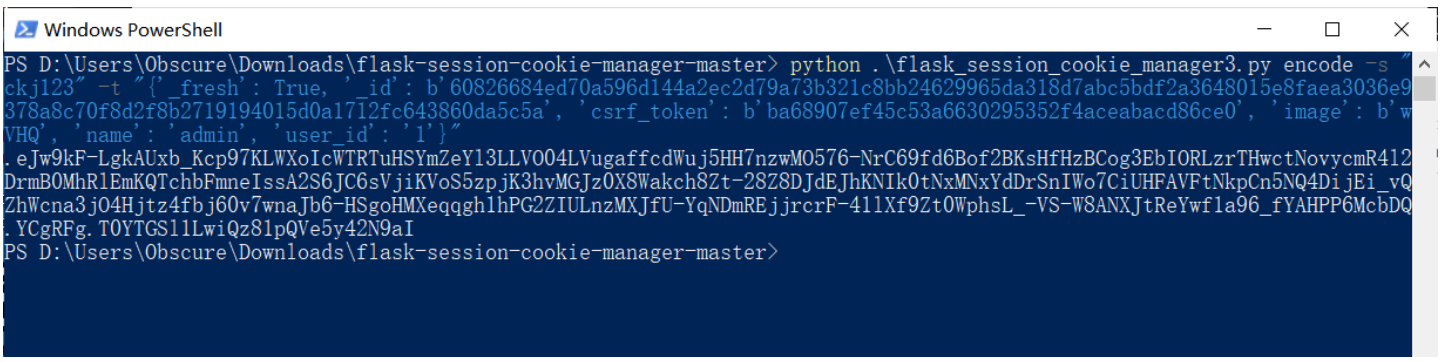


一般这个key可以是随机生成的，这里直接是固定的，也算减小题目难度了吧~
这个时候我们需要再找一个flask的session加密脚本，把admin的session伪造一个。

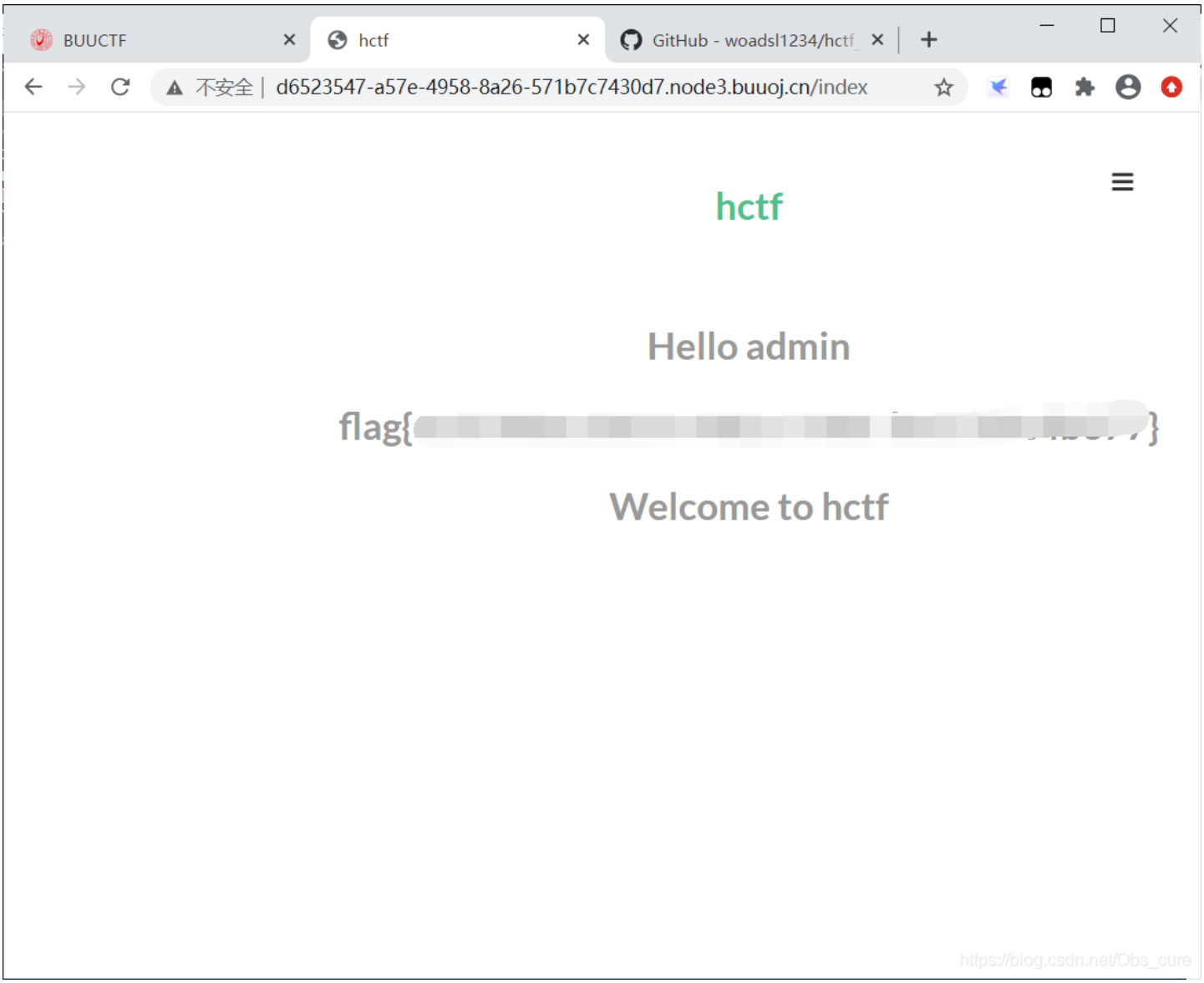
session加密脚本地址

这里要记得把user_id什么的改一下

> python .\flask_session_cookie_manager3.py encode -s “ckj123” -t “{‘_fresh’: True, ‘_id’: b’60826684ed70a596d144a2ec2d79a73b321c8bb24629965da318d7abc5bdf2a3648015e8faea3036e9378a8c70f8d2f8b2719194015d0a1712fc643860da5c5a’, ‘csrf_token’: b’ba68907ef45c53a6630295352f4aceabacd86ce0’, ‘image’: b’wVHQ’, ‘name’: ‘admin’, ‘user_id’: ‘1’}”




https://blog.csdn.net/Obs_cure


BUUCTF
hctf
GitHub - woadsl1234/hctf
不安全 | d6523547-a57e-4958-8a26-571b7c7430d7.node3.buuoj.cn/index
hctf
Hello admin
flag{
}
Welcome to hctf
https://blog.csdn.net/Obs_cure

（2）Unicode欺骗

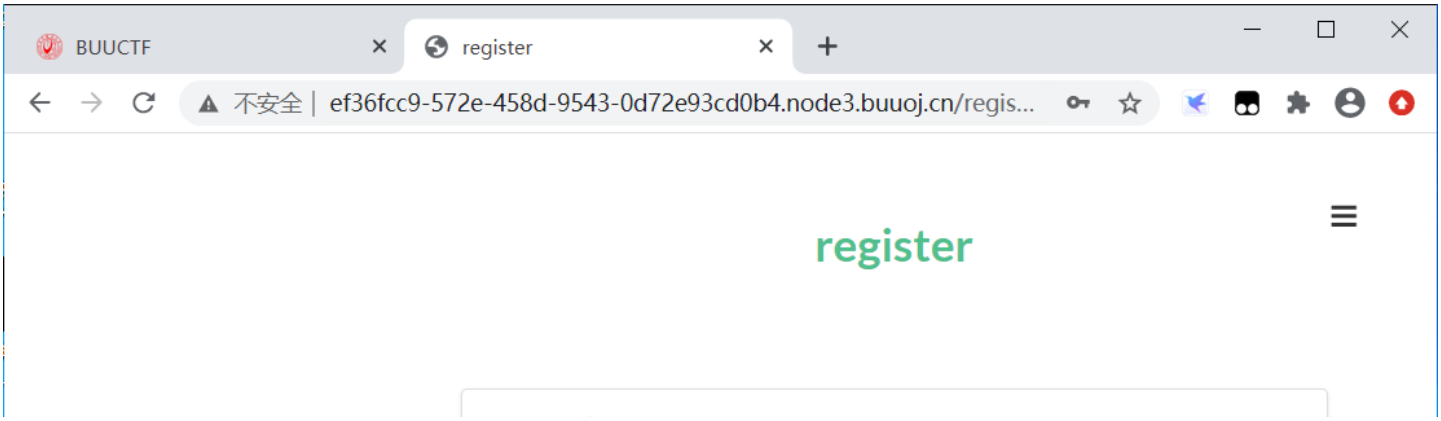
百度上的Unicode欺骗都是这道题，就很有趣

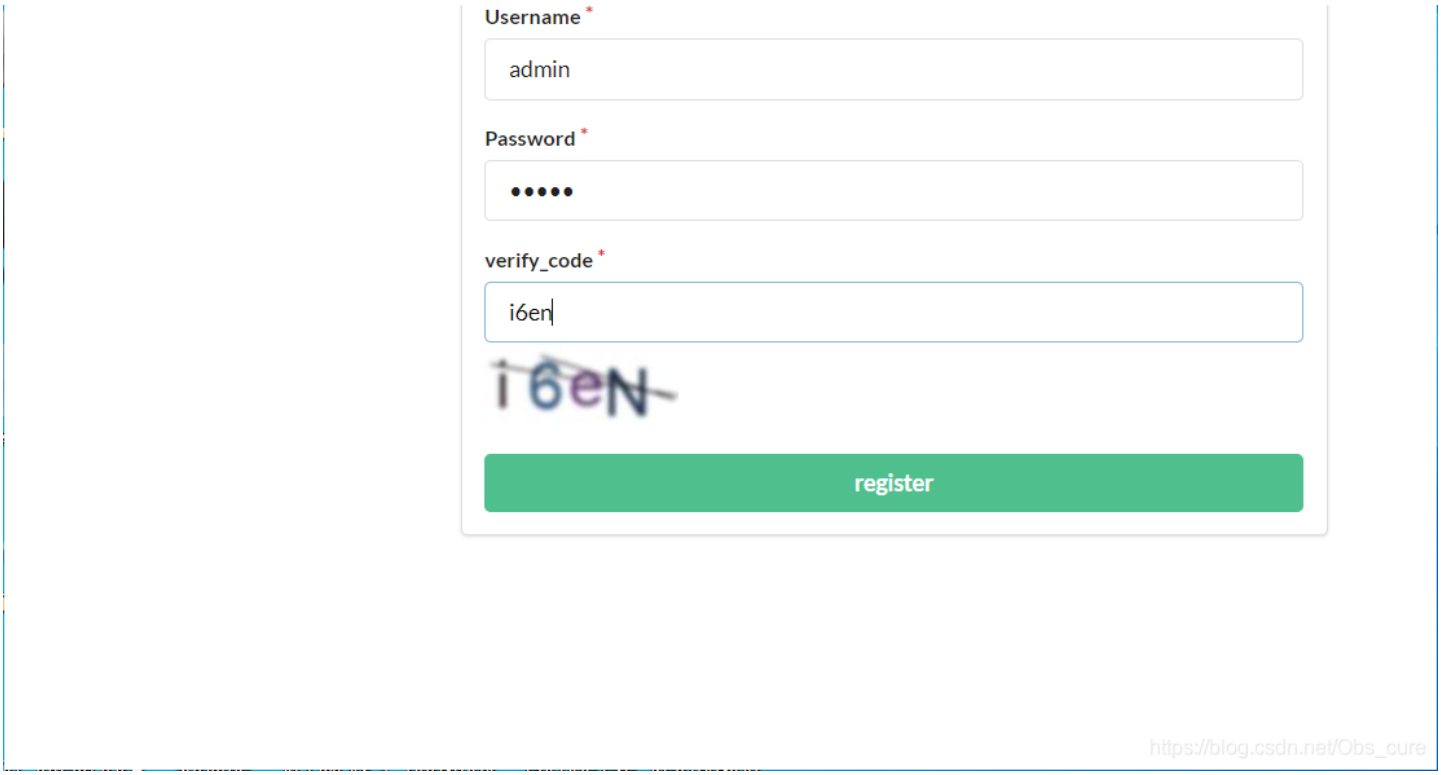
这个大佬的博客讲的很好，先细细的了解了一番。

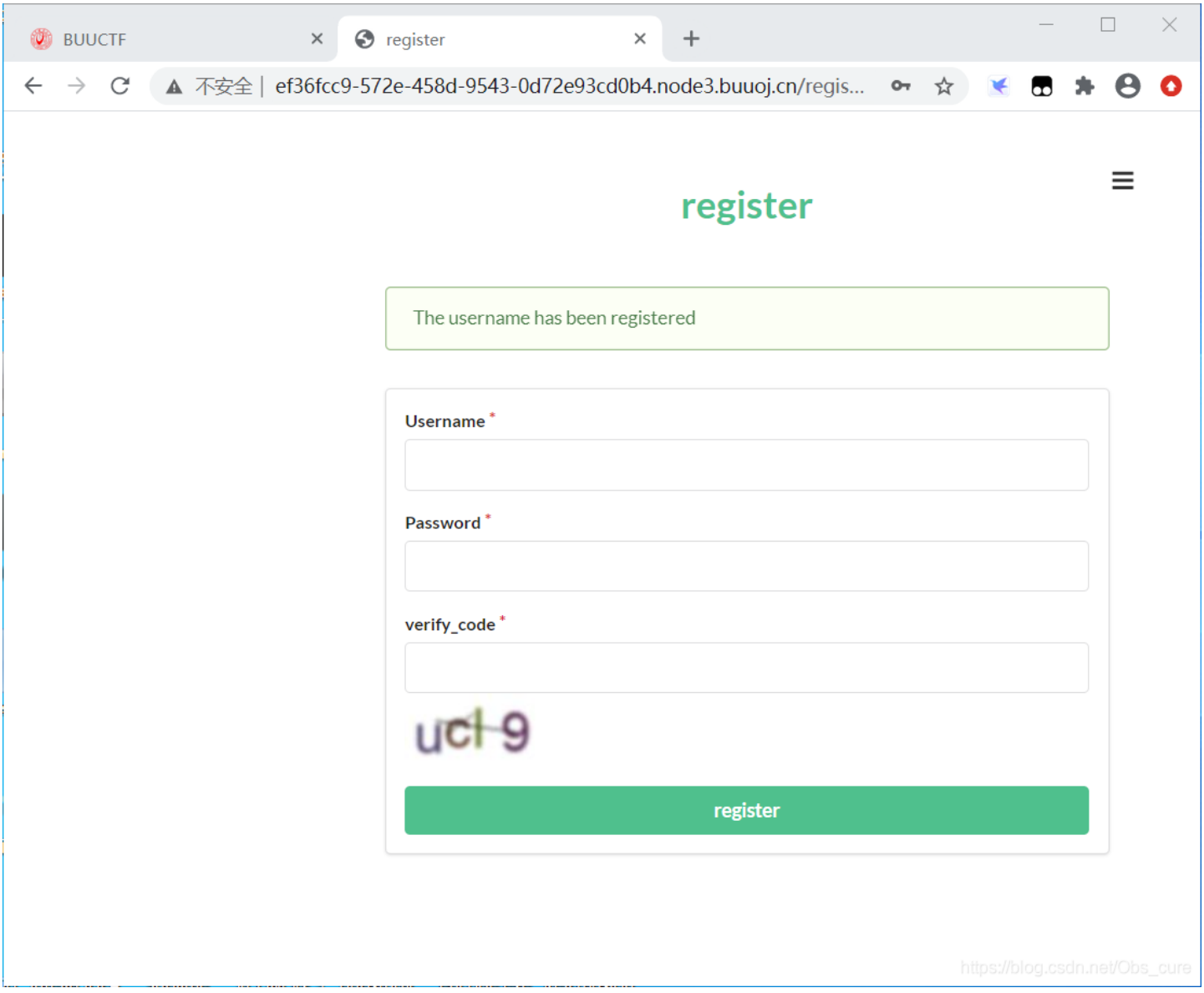
其实这个Unicode欺骗很好理解。就是我们平时英文字母 ABC…在希腊字母中，就变成了αβγ一样。由于占用的空间和使用的用途不同，在转码的时候会有各种方式。

那么在这道题中该如何利用Unicode欺骗呢？简单来说，就是找一个奇怪语言的admin重新注册一个，把之前的admin密码覆盖掉，就可以用我们的密码登陆了。

首先测试一下用admin注册：



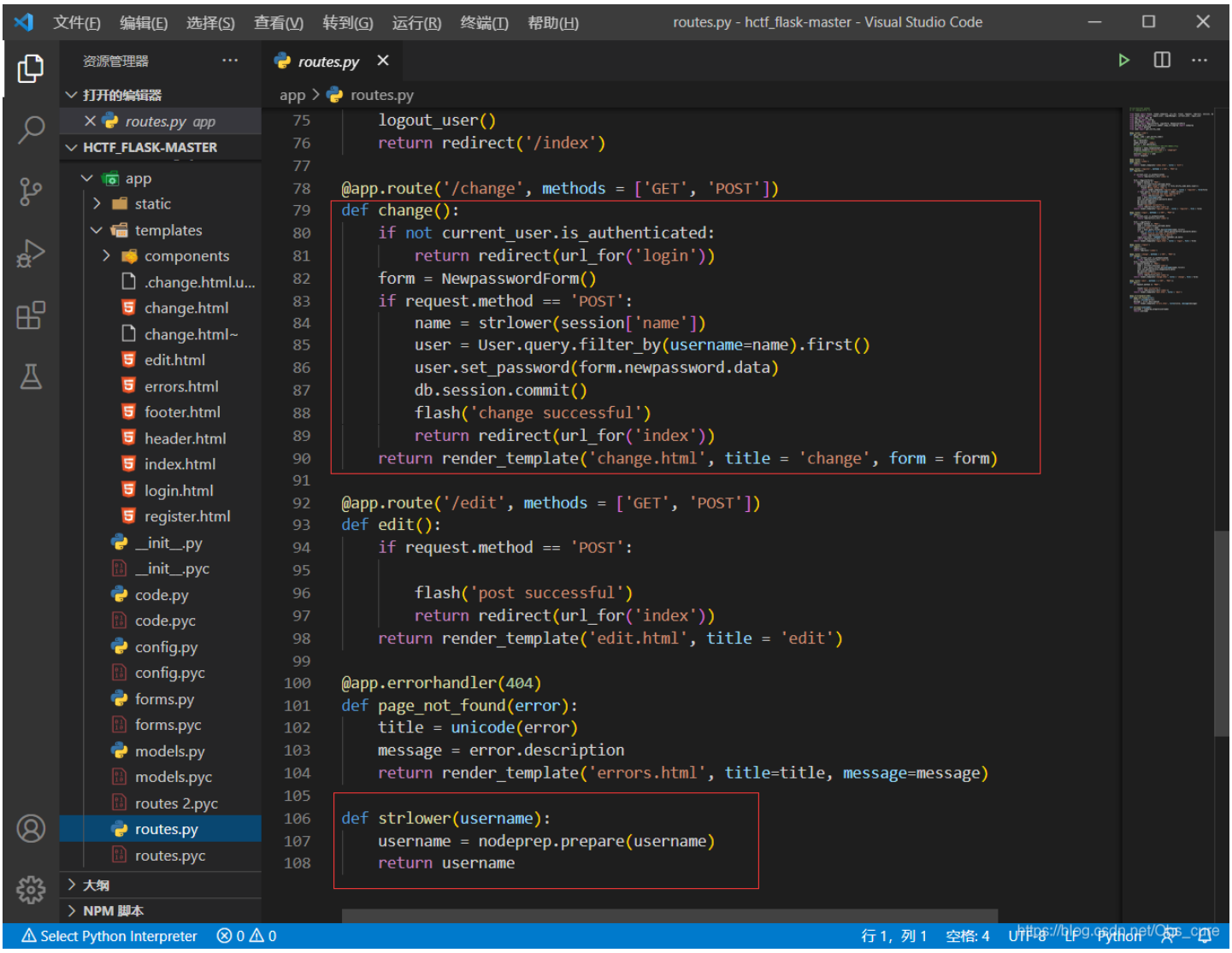

Username *
admin
Password *
•••••
verify_code *
i6en
register

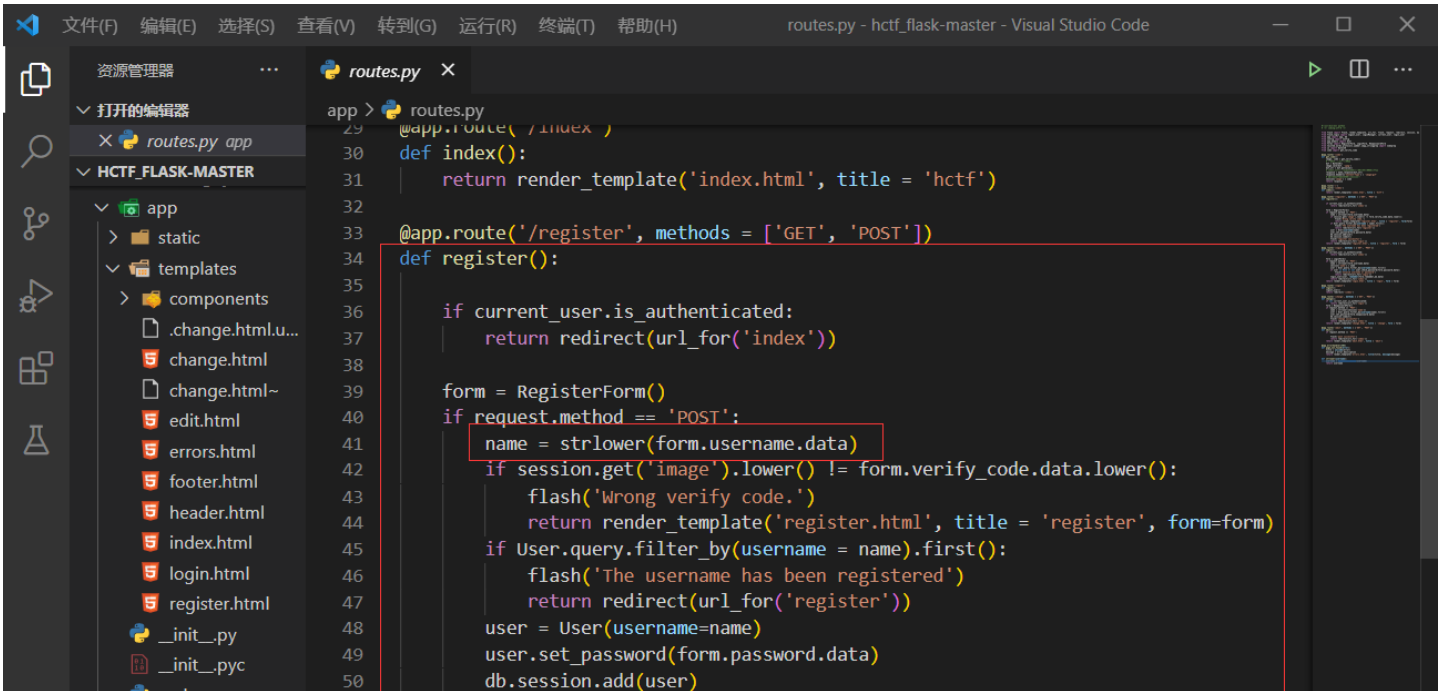

BUUCTF
register
不安全 | ef36fcc9-572e-458d-9543-0d72e93cd0b4.node3.buuoj.cn/regis...
register
The username has been registered
Username *
Password *
verify_code *
register

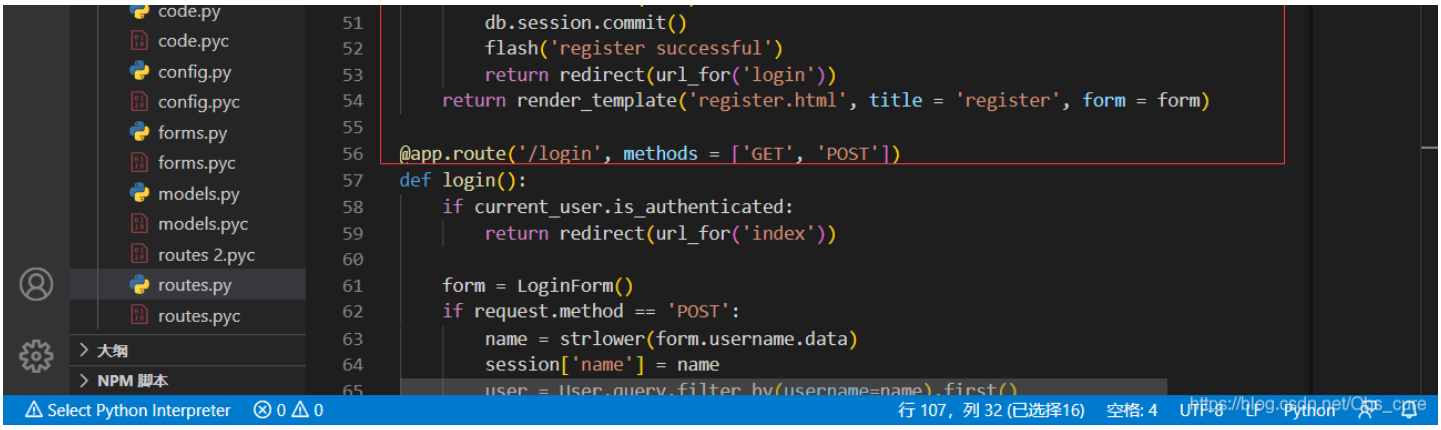
然后大佬们就在unicode表里面掏出来一个ᴬᴰᴹᴵᴺ扣了上去，很快啊~

但是好奇宝宝特别想知道这个Unicode是哪国文字，于是不信邪去找了一下...这个小写的ᴬ的Unicode编码是7468（1D2C），在1D00-1D7F：语音学扩展 (Phonetic Extensions)中。这里指路个博客，ᴬ在第1024行哦~

接下来再看看原理。通过源码可得

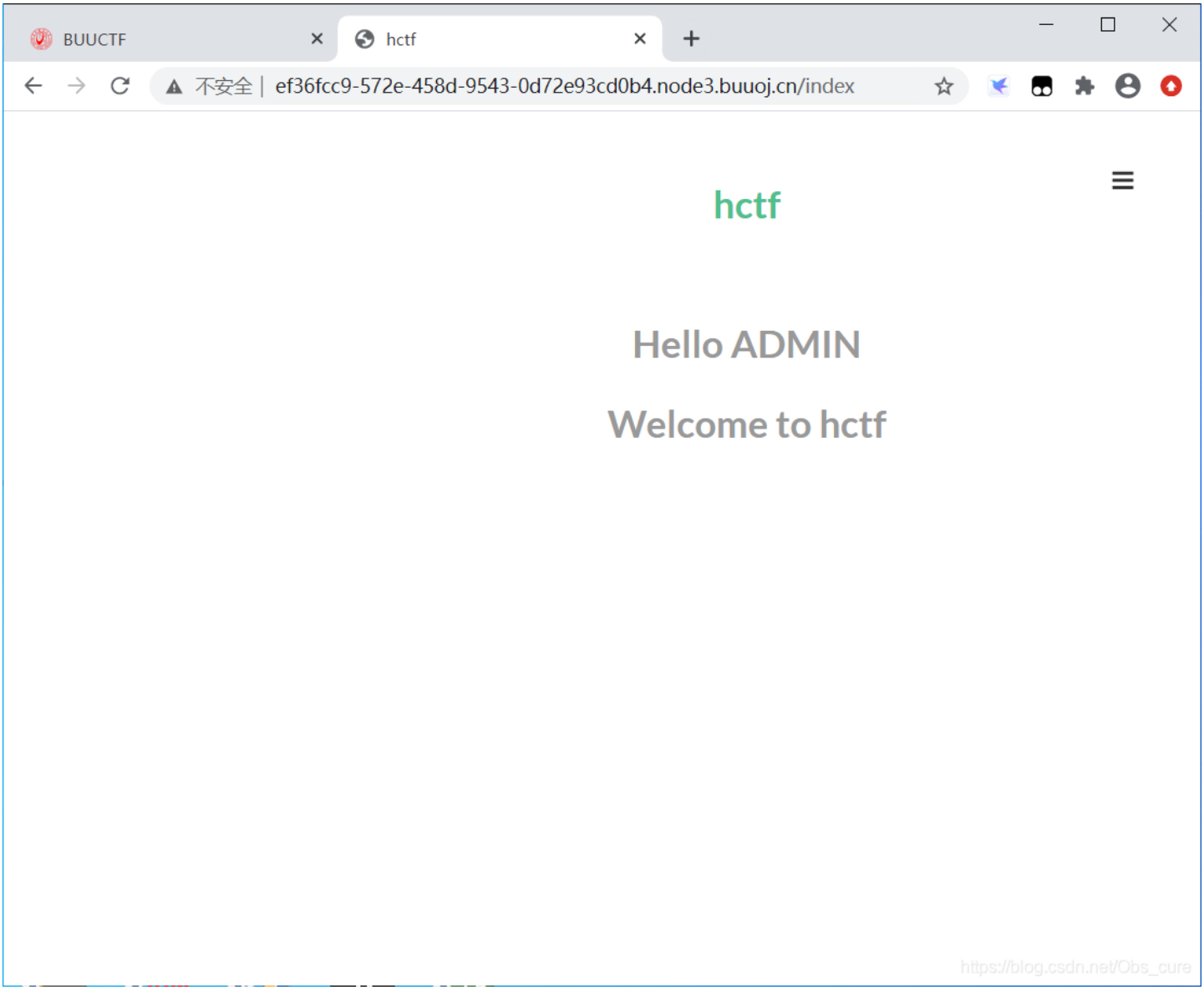




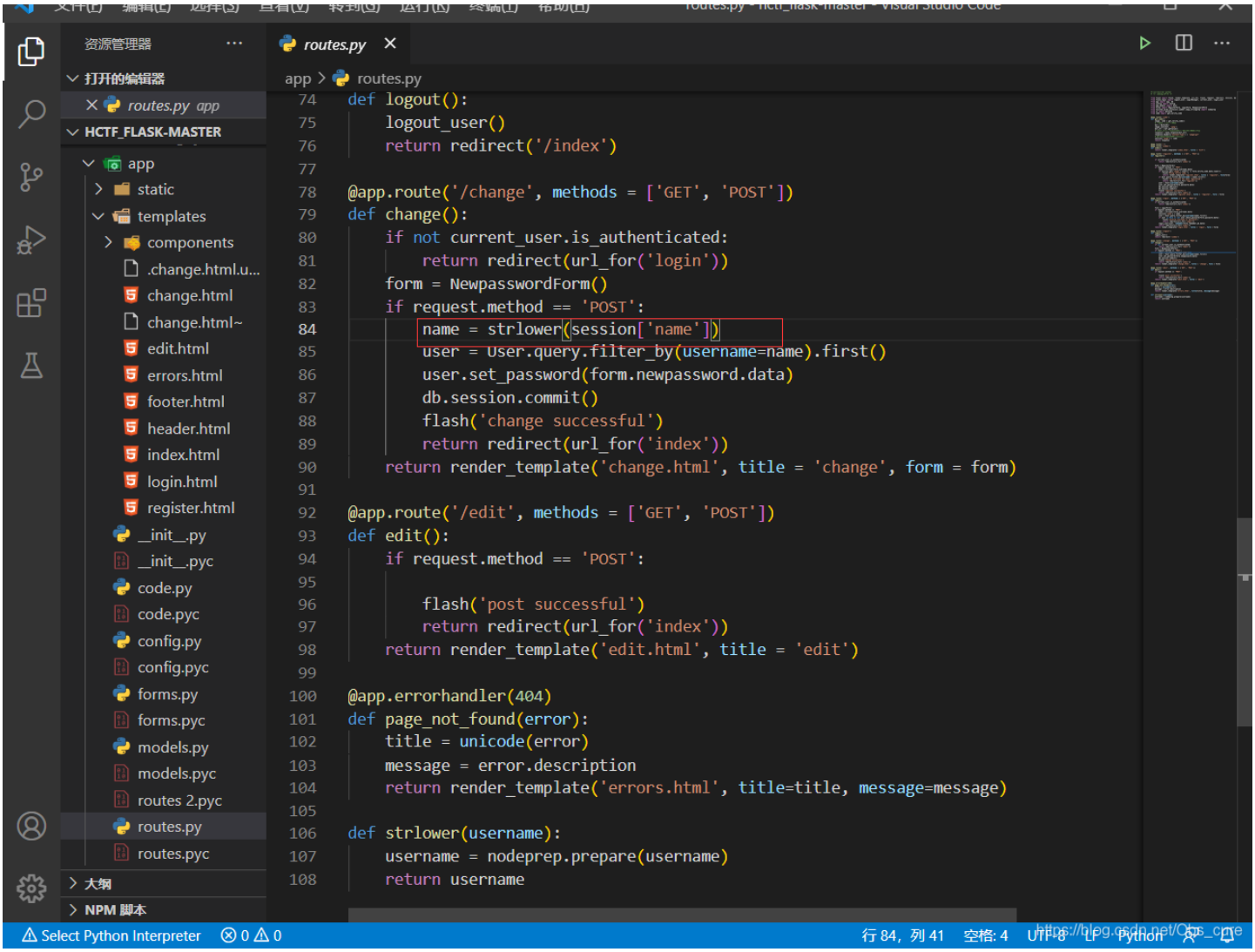


首先是strlower函数。这个函数本意是把大写转换成小写。nodeprep.prepare的本意也是把A转换成a.但他遇见ᴬᴰᴹᴵᴺ时，会转换成ADMIN。这是这个函数的漏洞。

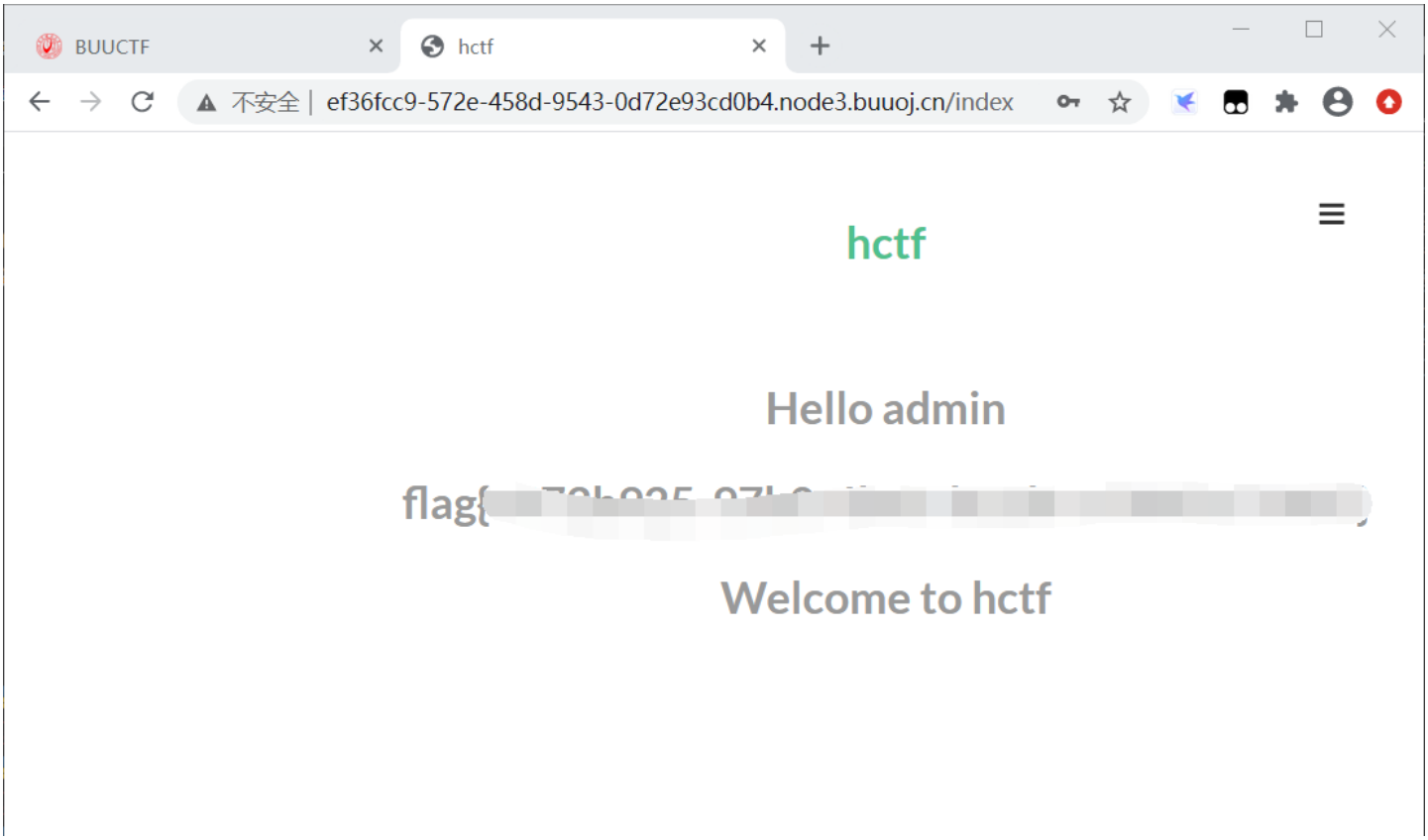
然后先看注册部分，我们的输入的用户名会经过strlower函数。如果没有这个漏洞，我们所有的用户名都是小写。但我们输入转换ᴬᴰᴹᴵᴺ时，会转换成ADMIN，这样就会跳过下面的判断。



注册成功。
我们继续看更改密码的部分函数。



这里又调用了一次strlower函数。我们现在的用户名是ADMIN。再次调用后我们的用户名就会变成admin。这样就会更改掉的密码。



（3）弱口令

这道题账户是admin，密码是123
弱口令yyds：）

3.总结

1. 学会了session的修改和判断
2. 了解了flask框架下的一个函数的一个漏洞
3. 帮助学习了一下python的知识。一度以为strlower函数是重载的，查了半天

4.参考资料

- BUUCTF | [HCTF 2018]admin - !ao!ao - 博客园
- Session机制详解 - lonelydreamer - 博客园
- session（计算机术语）_百度百科
- HCTF2018-admin_迷风小白-CSDN博客