《安全程序设计与实践》
实验报告

实验名称：XSS 跨站攻击

实验类型：实操型

指导教师：孙海峰

专业班级：

姓　　名：

学　　号：

实验地点：东 11C227

实验日期：2025/5/28

一、 实验目的

1. 可以理解 XSS 跨站攻击产生的原理和危害
2. 掌握 HttpOnly 属性、HTML 转义和 JavaScript 转义防护 XSS 攻击的方法

二、 实验过程

任务一：创建数据库

导入数据库脚本，新建一个表 messages

```
C:\Users\Administrator\Desktop\Web\apache2.4\httpd-2.4.33-o102o-x86-vc14-r2\Apache24\htdocs\xss>mysql -uroot -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 58
Server version: 5.5.59 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> source lab.sql
Query OK, 1 row affected, 1 warning (0.09 sec)

Database changed
Query OK, 0 rows affected, 1 warning (0.10 sec)
```

查看表结构，messages 表建立成功

```
mysql> use lab;
Database changed
mysql> desc messaages;
ERROR 1146 (42S02): Table 'lab.messaages' doesn't exist
mysql> desc messages;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| message | varchar(256) | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

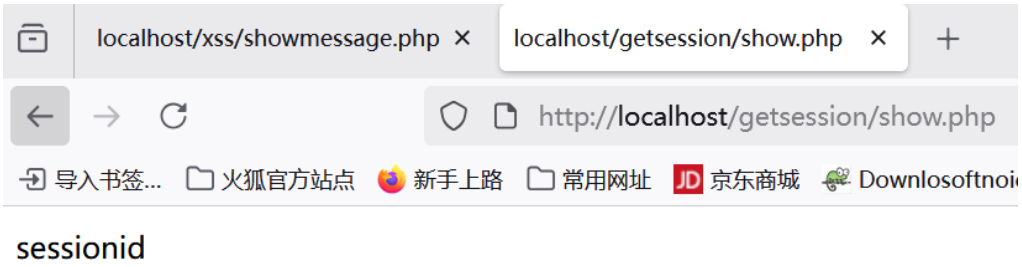
任务二：建立具有接收 SessionID 功能和具有留言功能的网站

2-1 任务实现

将网站代码回滚到初始版本

```
pyp@LAPTOP-6NGOAOTJ MINGW64 /e/作业/web/代码-git/代码-git/项目10/xss (master)
$ git reset --hard 5bf099431c21e6dfb01db202ed48b58f94e4469c
HEAD is now at 5bf0994 json_encode for test.html
```

完成建立具有接收 SessionID 功能网站 getsession



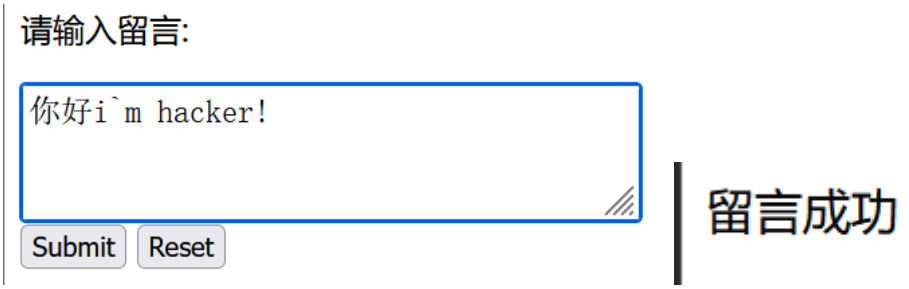
将 xss 网站中的 showmessage.html 第 21 行修改为 echo "<tr><td>$row[1]</td></tr>";

```
echo "<table>";
echo "<tr><td>sessionid</td></tr>";
while($row = mysqli_fetch_array($rs))
    echo "<tr><td>$row[1]</td></tr>";    //显示数据

echo "</table>";

// 释放结果集
```

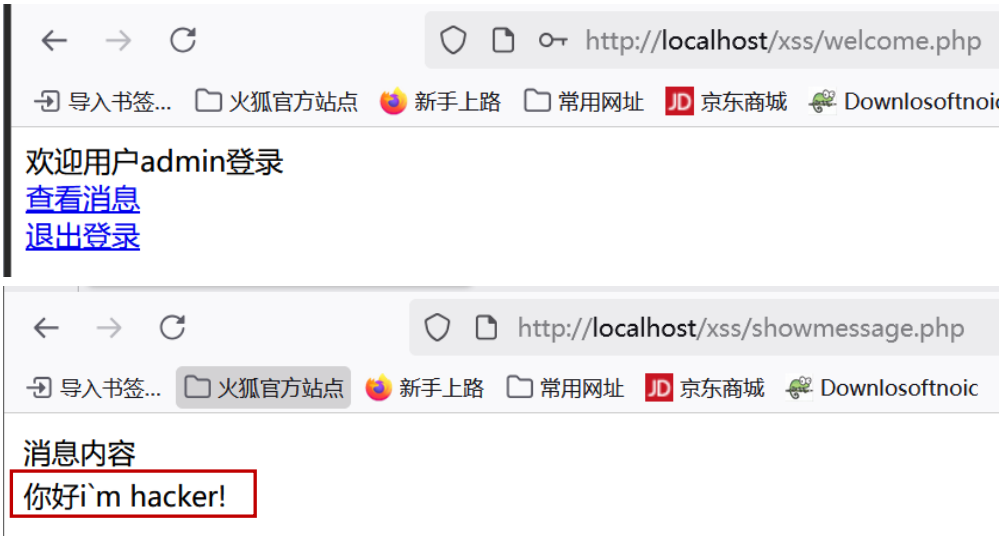
完成建立具有留言功能网站 xss

2-2 功能测试

访问 messages.html 留言板进行留言



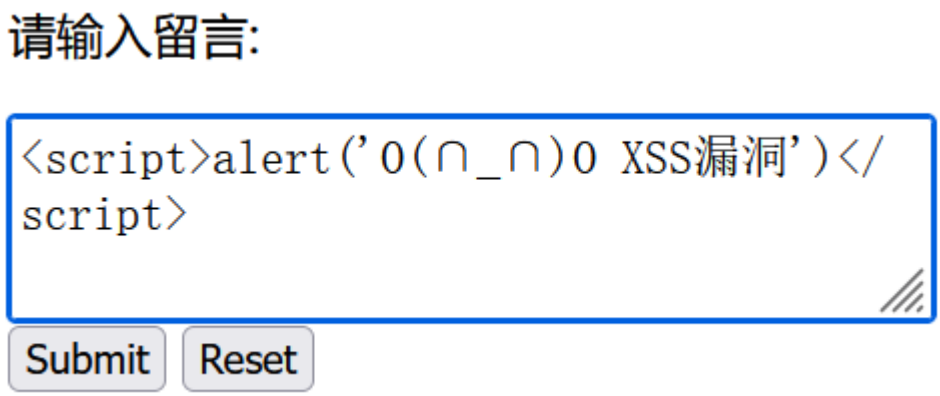
访问 login.php 登录管理员用户“admin/admin123”，访问 showmessage.php 查看留言板，留言显示正常，留言功能实现成功。



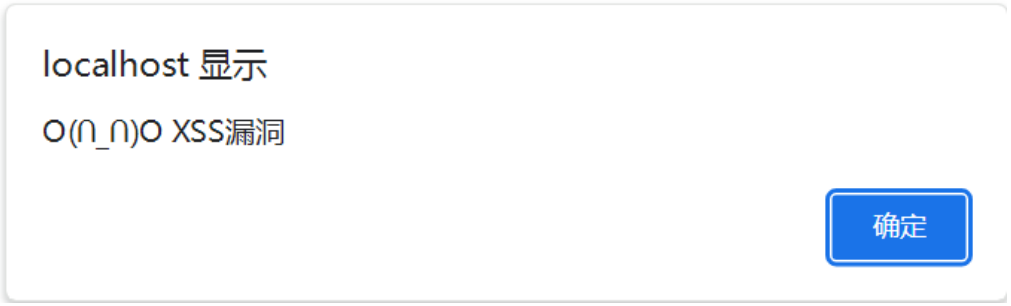
任务三：XSS 攻击测试

3-1 弹窗式 XSS 攻击测试

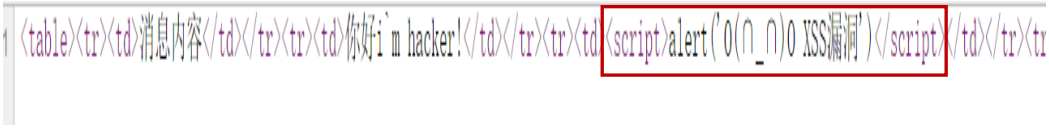
在火狐浏览器访问留言板界面输入 xss 恶意脚本，将内容显示在弹出的提示框上



打开 360 浏览器使用管理员用户查看留言，弹出提示框，与预期效果一致

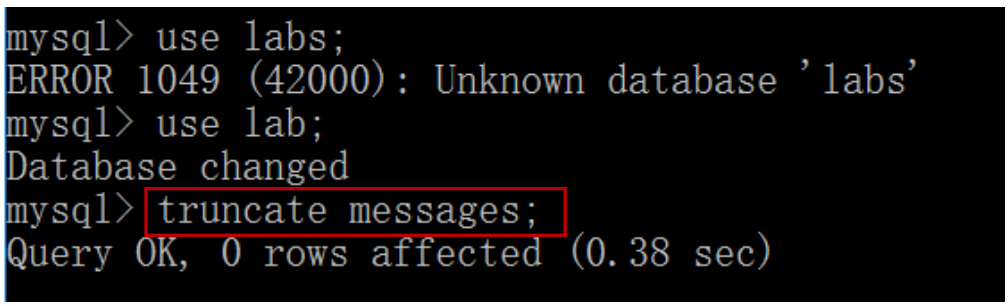


查看源代码，发现 alert 语句被当作脚本执行，因此弹出提示框输出内容



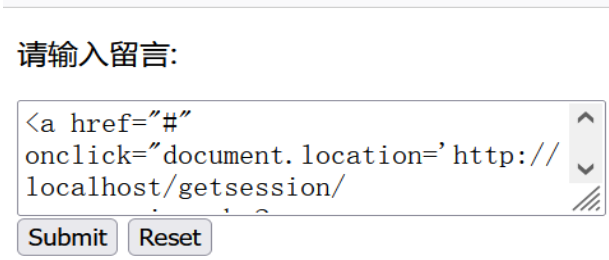
3-2 窃取 SessionID 攻击测试

Get 方式提交的表单只支持 ascii 字符，先使用 truncate 清空 messages 表数据

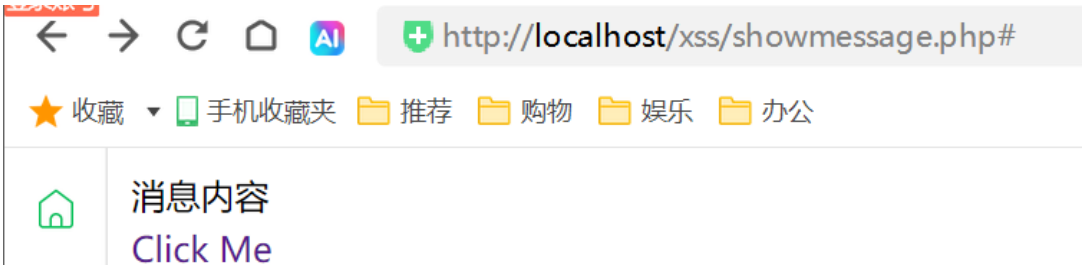


然后清空浏览器数据，防止先前的 cookie 干扰。

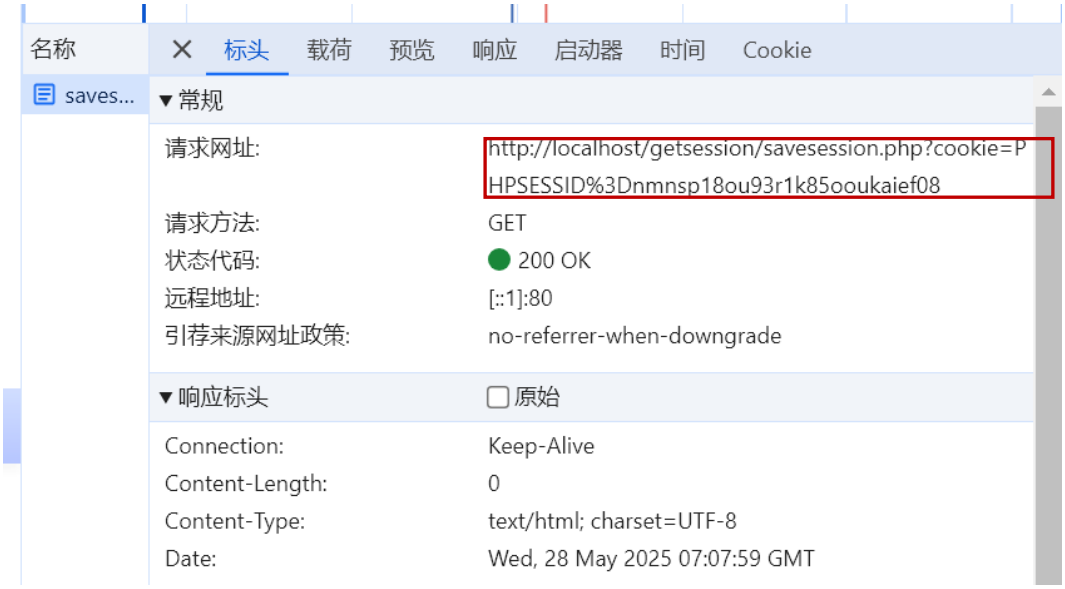
清理完后我们再次使用火狐浏览器进行恶意留言，这次使用超链接形式进行会话劫持，诱使管理员点击该超链接访问 getsession/savesession.php 获取其 cookie 并保存。



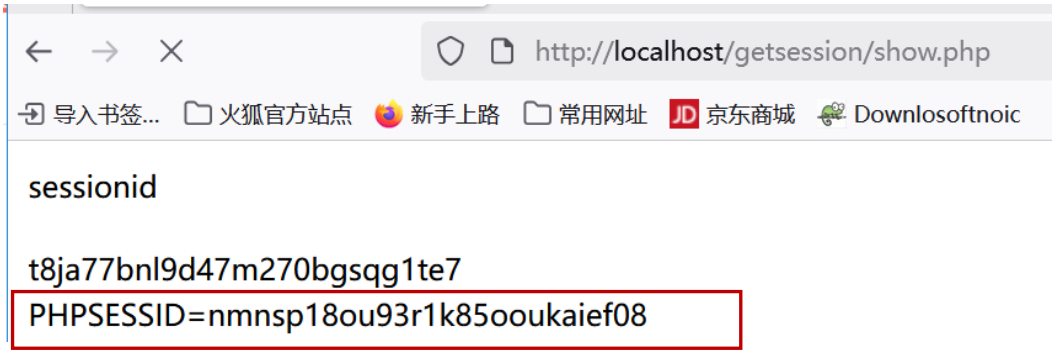
使用 360 浏览器登录管理员账号查看留言板，发现 Click Me，点击



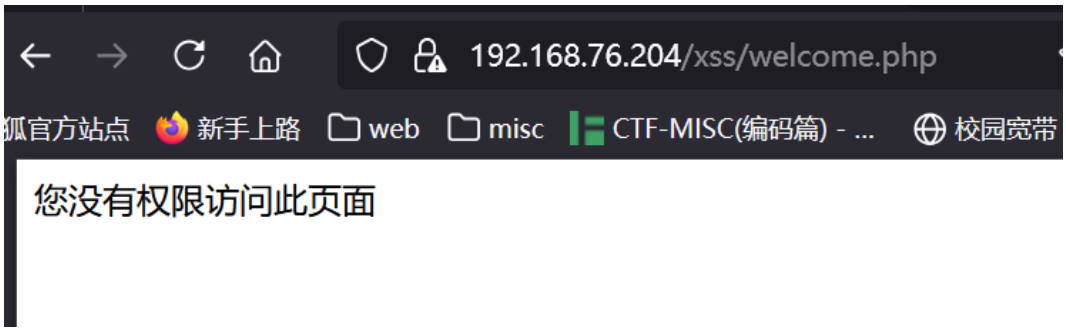
没有任何的回显但是可以查看请求头，发现跳转到 savesession.php 页面，并且 cookie 传参 PHPSESSID，我们保存下其对应 UserAgent：Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95 Safari/537.36



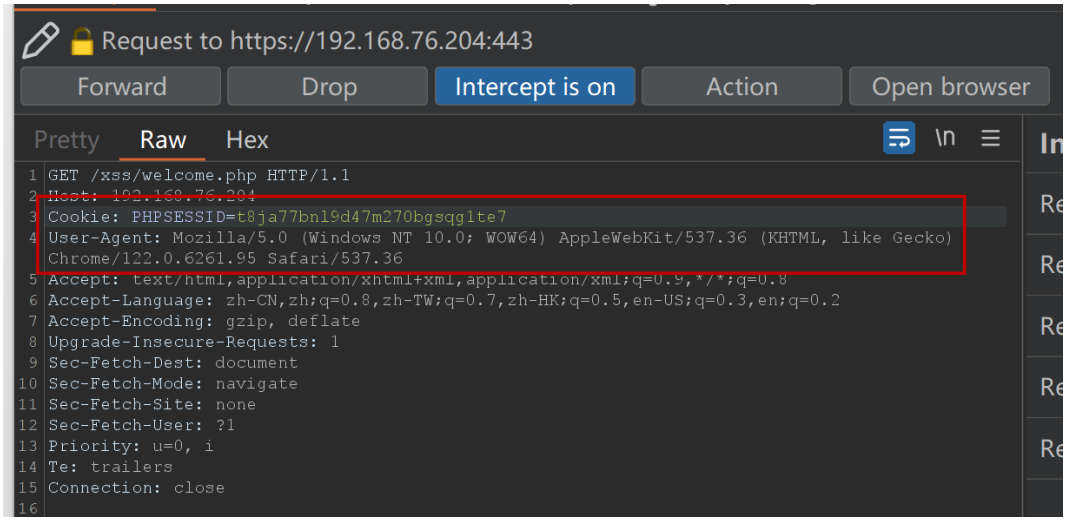
访问 getsession/savesession.php 发现存在刚刚窃取到的管理员 cookie



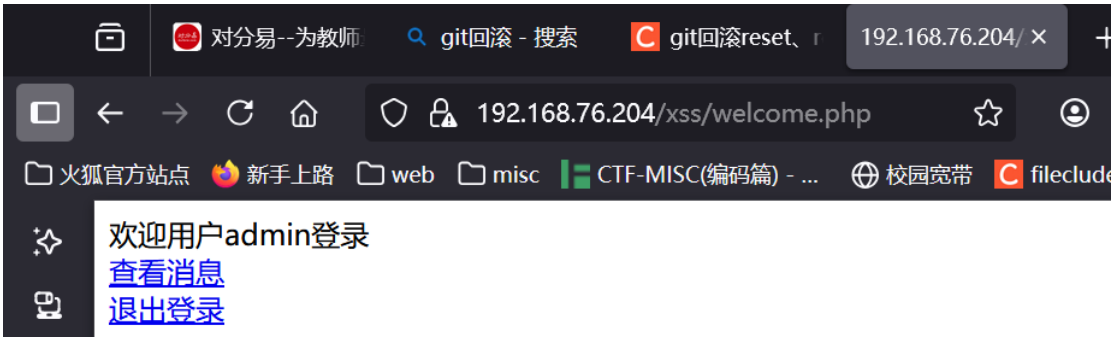
我们使用窃取到的 Cookie 进行 Cookie 欺骗，首先使用火狐访问 welcom.php 界面，提示没有权限



抓包然后改 cookie 和 User-Agent 后放包



访问成功



3-3 测试分析

由于页面 showmessages.php 中显示留言板消息没有经过任何的过滤（echo "<tr><td>$row[1]</td></tr>";），而是原样输出，导致攻击者可以构造恶意 js 脚本进行攻击（访问 document.cookie 或者进行会话劫持）从而获取 cookie。

任务四：XSS 攻击防护

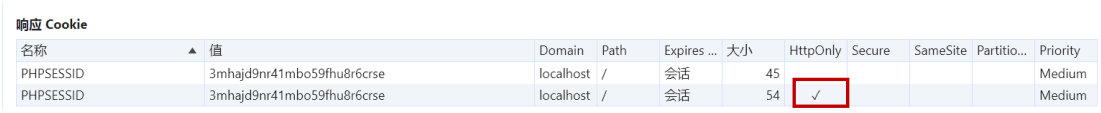
4-1 设置 Cookie 的 HttpOnly 属性：

HttpOnly 属性为 true 时 Cookie 无法被 document.cookie 访问，攻击者从而无法获取到 cookie 值。

在 function.php 中设置 HttpOnly，第 35 行后加入 setcookie 语句，关闭浏览器 cookie 即失效且将作用域布置到 xss 目录下

```
34      check_user_agent();
35      session_regenerate_id(true);
36      setcookie(session_name(), session_id(), NULL, '/', NULL, FALSE, TRUE);
```

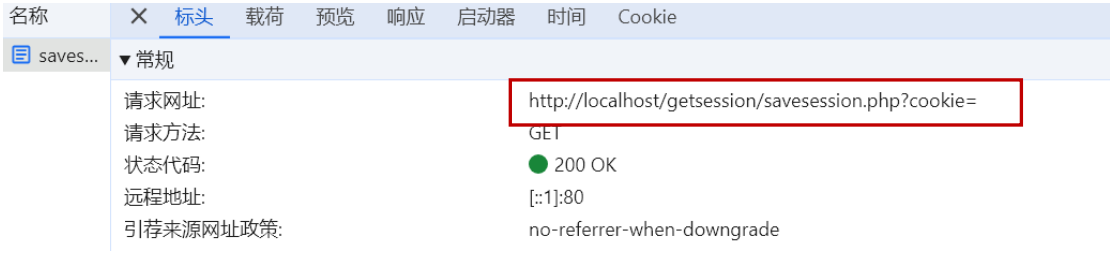
我们再次使用 360 登录后，能发现已经开启了 HttpOnly 属性



响应 Cookie

名称	值	Domain	Path	Expires ...	大小	HttpOnly	Secure	SameSite	Partitio...	Priority
PHPSESSID	3mhajd9nr41mbo59fhu8r6crse	localhost	/	会话	45					Medium
PHPSESSID	3mhajd9nr41mbo59fhu8r6crse	localhost	/	会话	54	✓				Medium

再次访问留言板，点击 Click 发现无法向 cookie 传参，这是因为攻击语句是访问的 document.cookie 获取 cookie 值传参过去，无法读取 document.cookie 时就为空。

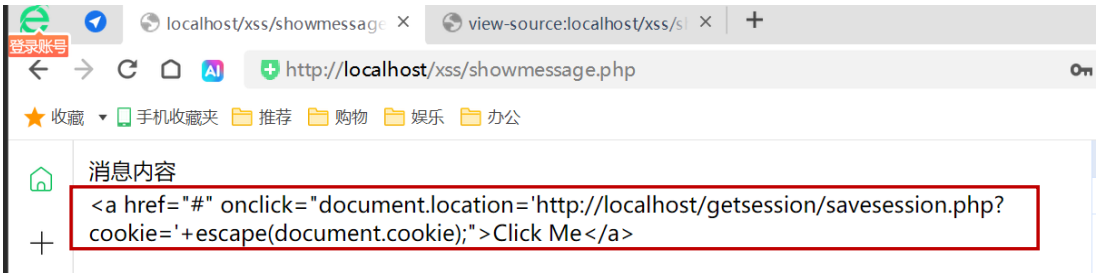


4-2 HTML 转义

由于一开始查看留言板时留言内容不会经过任何过滤就输出，导致攻击者构造的特殊语句得以运行，我们在 showmessage.php 中将留言板数据进行输出转义

```
16  // 用 HTML 显示结果
17  echo "<table>";
18  echo "<tr><td>消息内容</td></tr>";
19  if(mysqli_affected_rows($con)>0)
20      while($row = mysqli_fetch_array($rs))
21          echo "<tr><td>" .htmlentities($row[1]) ."</td></tr>";   //显示数据
22          //echo "<tr><td>$row[1]</td></tr>";
23  echo "</table>";
```

清空数据库后再次进行攻击测试后查看留言板，能发现“<”转义为了&lt，“>”转义成立%gt，“”“转义成了&quto;，防止了语句被当作 html 标签执行，而是原样输出内容，就不会被解释为超链接。



4-3 JavaScript 转义

创建 test.html 进行测试，应用 JQuery 框架快速测试。

定义了一个名为$username 变量，其值是一段包含 JavaScript 代码，\u003c 和\u003e 分别是<和>的 Unicode 转义形式

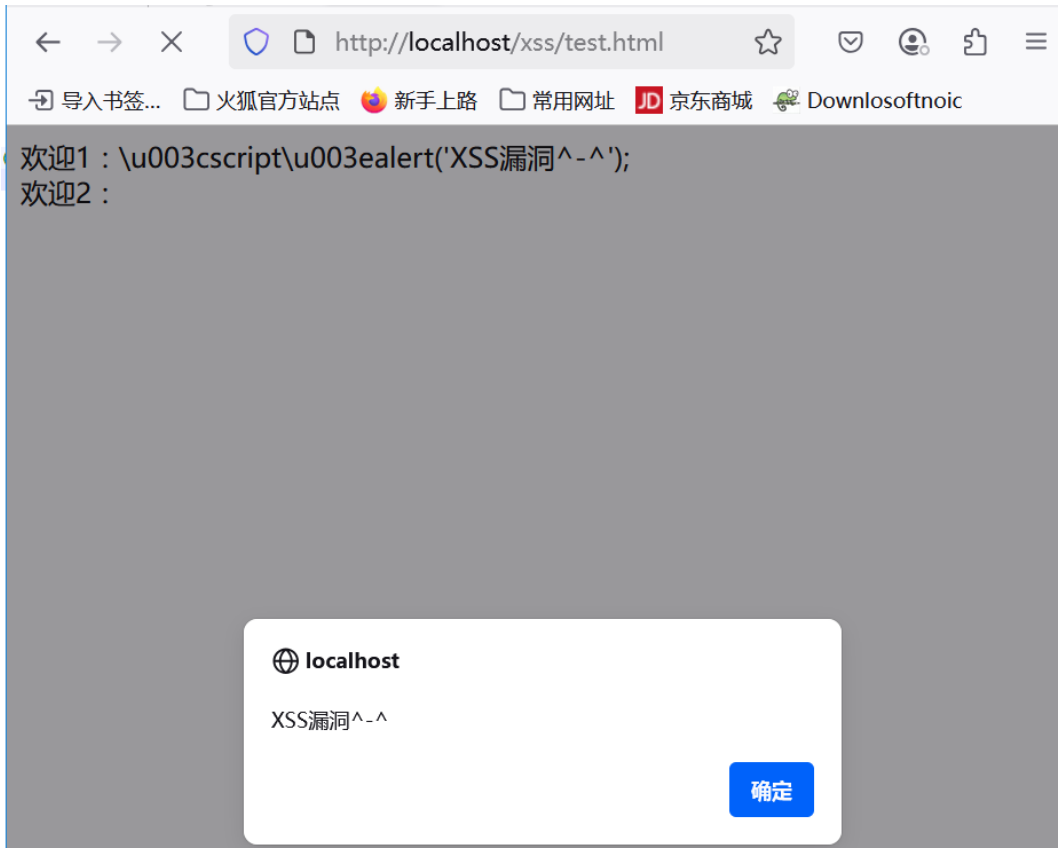
第一条使用 htmlentities 函数对$username 进行转义，把特殊字符转换为 HTML 实体。

第二条语句没有进行任何处理直接输出。

选中页面中 id 为 username_info 的元素， 使用 htmlentities 转义后的$username 内容设置到选中元素中。

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script src="http://libs.baidu.com/jquery/2.1.4/jquery.min.js"></script>
</head>
<body>

<?php $username="\u003cscript\u003ealert('XSS漏洞^-^');"; ?>
<div>欢迎1：<?php echo htmlentities($username);?> </div>
<div>欢迎2：<span id="username_info"></span></div>
<script>
//$('#username_info').html(<?php echo json_encode(htmlspecialchars($username));?>);
$('#username_info').html("<?php echo htmlentities($username);?>");
</script>
</body>
</html>

```

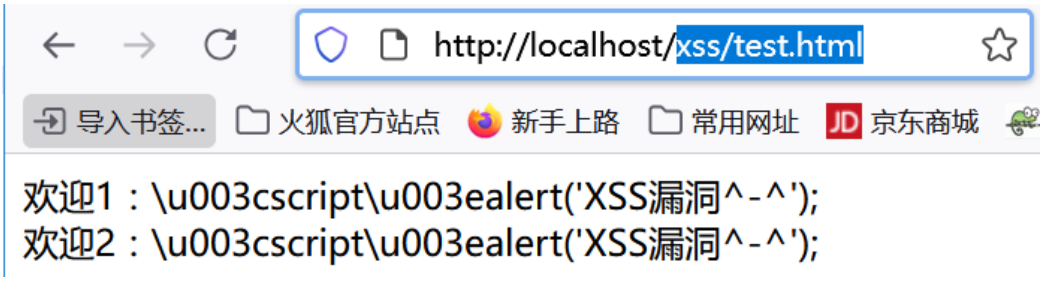
测试发现，htmlentities 不会对“\”进行处理但是 javascrpt 会，还原并执行其转义后对应的字符，所以语句 1 防止了 xss 攻击，而语句二存在 xss 攻击。



我们再次修改，防范 javascript 攻击：先使用 htmlspecialchars 函数对$username 进行转义，再通过 json_encode 转换为 JSON 格式，最后用 html 方法将内容设置到选中元素中

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script src="http://libs.baidu.com/jquery/2.1.4/jquery.min.js"></script>
</head>
<body>

<?php $username="\u003cscript\u003ealert('XSS漏洞^-^');"; ?>
<div>欢迎1：<?php echo htmlentities($username);?> </div>
<div>欢迎2：<span id="username_info"></span></div>
<script>
$('#username_info').html(<?php echo json_encode(htmlspecialchars($username));?>);
</script>
</body>
</html>

```

测试发现，防范成功



三、 实验结论

四、 思考题

1. **你还知道哪些 XSS 跨站攻击形式？如何防御？**

攻击形式：

- 存储型 XSS ：恶意脚本被存储在服务器端（如数据库），常见于用户可提交内容并展示的功能模块，像论坛留言、评论区、用户私信等。当其他用户访问包含恶意脚本的页面时，脚本就会在其浏览器中执行。
- 反射型 XSS ：恶意脚本存在于 URL 中，用户访问带有恶意脚本参数的 URL 时，服务器将脚本反射回页面，在用户浏览器执行。常出现于网站搜索、跳转等通过 URL 传递参数的功能。
- DOM 型 XSS ：不涉及服务器端，攻击者利用前端 JavaScript 代码对 DOM 操作的漏洞，通过修改本地 DOM 环境执行恶意脚本。比如，攻击者构造特殊 URL，其参数被用于修改页面 DOM 时触发恶意代码执行，
- 基于富文本编辑器的 XSS ：富文本编辑器若存在安全漏洞，攻击者可利用其插入恶意脚本。
- MHTML XSS（HTML 邮件 XSS） ：一种将 HTML 页面及相关资源整合在单个文件的格式。攻击者构造包含恶意脚本的 MHTML 格式邮件，当用户使用支持 MHTML 的邮件客户端打开时，恶意脚本可能被执行。

防御方法：

1) 输入验证与过滤：在服务器端对用户输入进行严格检查，拒绝或过滤掉不合法字符及可能构成恶意脚本的内容。
2) 输出编码：将动态输出到 HTML 页面的数据进行转义，把特殊字符转换为 HTML 实体。
3) Content Security Policy（CSP）：通过设置 HTTP 响应头或 HTML 的 <meta> 标签，定义页面允许加载的资源来源，限制可执行脚本的来源
4) HttpOnly Cookie ：设置 Cookie 的 HttpOnly 属性，使 JavaScript 无法访问该 Cookie，防止攻击者利用 XSS 窃取 Cookie 进行会话劫持等攻击。
5) 避免危险的 DOM 操作：谨慎使用 .innerHTML、document.write() 等可能将不可信数据当作代码执行的 DOM 操作方法，多用 .textContent、.setAttribute() 等安全方式操作 DOM。
6) 安全的富文本编辑器配置：对富文本编辑器进行合理配置和安全加固，启用其

自带的安全过滤功能，或补充额外的过滤规则，确保用户输入的内容经过严格净化后再存储和展示。

7) 及时更新软件和系统：及时修复 Web 应用程序、框架、服务器软件等的安全漏洞，关注官方发布的安全补丁并及时部署。

2. 怎样利用 XSS 跨站攻击对 Cookie 验证进行欺骗？

攻击者在注入的恶意脚本中，使用 document.cookie 获取用户当前页面的 Cookie 信息，然后，通过 XMLHttpRequest 或 Fetch 等方式，将获取到的 Cookie 信息发送到攻击者控制的服务器。攻击者拿到用户的 Cookie 后，在其他设备或会话中，将该 Cookie 添加到请求头中，向目标网站发送请求，服务器基于 Cookie 验证认为是合法用户，攻击者就能以用户身份进行操作，实现 Cookie 欺骗，比如进行转账、修改用户信息等操作。

3. 怎样实现在 XSS 攻击的同时获得网页地址、User-Agent 和 Cookie？

在恶意脚本中，使用 document.location.href 可获取当前页面的完整 URL 地址。

通过 navigator.userAgent 可获取用户浏览器的 User - Agent 信息。

使用 document.cookie 获取 Cookie 信息。

我们可以构造语句进行测试：Click Me(promote)