

软件安全实验11

姓名：何叶 学号：2313487 班级：范玲玲班

目录

软件安全实验11

姓名：何叶 学号：2313487 班级：范玲玲班

目录

实验名称：跨站脚本攻击

实验要求：

实验原理：

实验步骤：

一、img方式

- 1.htdocs中新建xss_test.php文件
- 2.在Dreamweaver8打开xss_test.php
- 3.编写代码
- 6.进入127.0.0.1/xss_test.php
- 7.弹出弹窗，正确运行
- 8.代码分析

二、script方式

- 1.编写xss_test.php
- 2.进入127.0.0.1/xss_test.php
- 3.在页面输入xss脚本
- 4.submit后，页面输出
- 5.页面输入双写关键字脚本
- 6.没有congratulations弹窗
- 7.分析代码
- 8.将apache2handler.ini中的magic_quotes_gpc 设置为Off
- 9.再次submit
- 10.弹窗congratulations弹窗，正确运行
- 11.代码逻辑分析
 - (1)、禁用错误显示
 - (2)、获取用户输入：
 - (3)、移除危险字符串：
 - (4)、输出处理过的输入：
 - (5)、构建表单：
 - (6)、重新定义 alert 函数
 - (7)、总结

12.测试用例设计

- (1)、正常输入
- (2)、特殊字符
- (3)、大小写混合
- (4)、空输入
- (5)、长输入
- (6)、XSS攻击向量
- (7)、关键词绕过
- (8)、JavaScript功能
- (9)、表单提交
- (10)、边界字符

13.测试结果

心得体会：

实验名称：跨站脚本攻击

实验要求：

复现课本第十一章实验三，通过img和script两类方式实现跨站脚本攻击，撰写实验报告。有能力者，可以自己撰写更安全的过滤程序。







实验原理：

跨站脚本攻击（XSS）是一种常见的网络攻击方式，攻击者通过在网页中注入恶意脚本，使得其他用户在浏览该网页时执行这些脚本，从而达到窃取用户信息或其他恶意目的。本实验通过模拟XSS攻击，学习如何防范此类攻击。

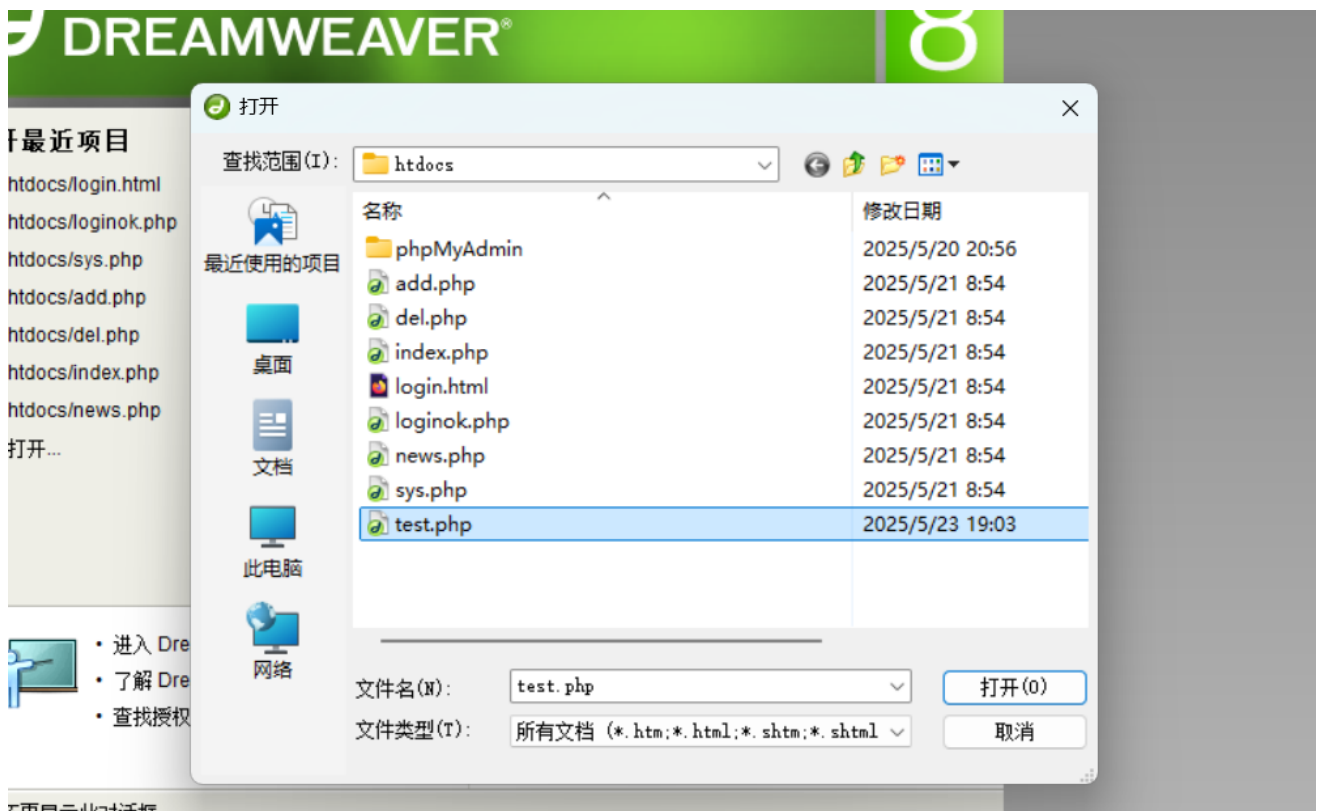
实验步骤：

一、img方式

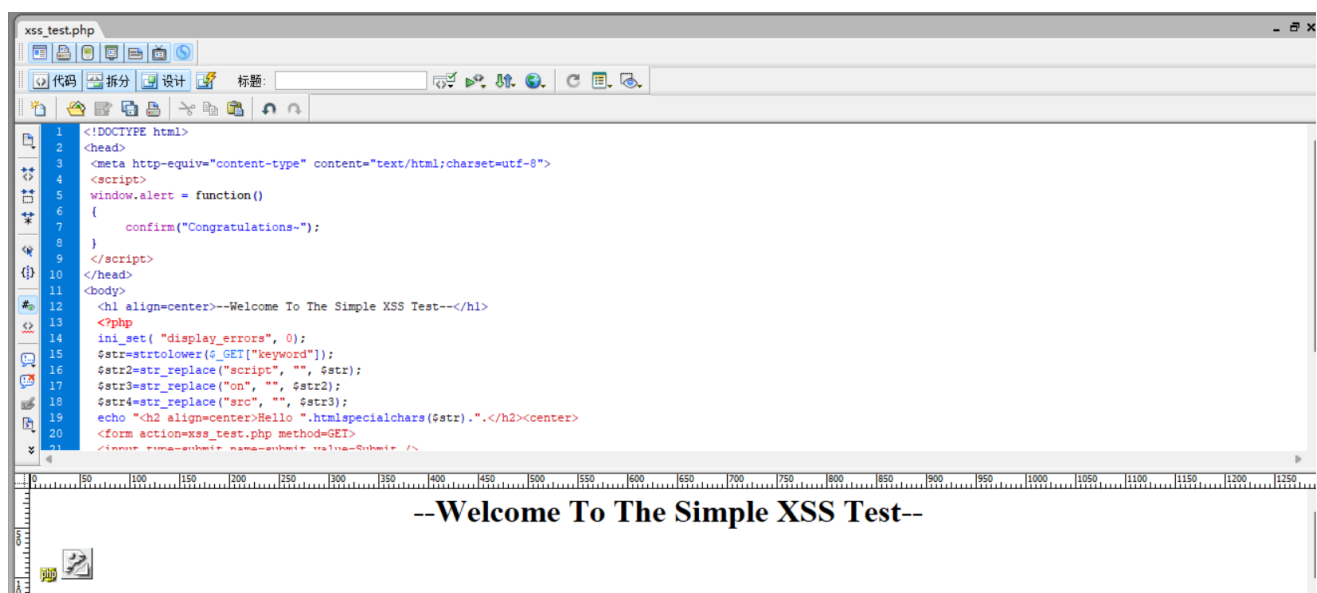
1.htdocs中新建xss_test.php文件

	index.php	2025/5/21 8:54	PHP Script	2 KB
	login.html	2025/5/21 8:54	Firefox HTML D...	1 KB
	loginok.php	2025/5/21 8:54	PHP Script	1 KB
	news.php	2025/5/21 8:54	PHP Script	2 KB
	sys.php	2025/5/21 8:54	PHP Script	2 KB
	test.php	2025/5/23 19:03	PHP Script	0 KB

2.在Dreamweaver8打开xss_test.php



3.编写代码



```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
window.alert = function()
{
    confirm("Congratulations~");
}
</script>
</head>
<body>
<h1 align=center>--welcome To The Simple XSS Test--</h1>
<?php
ini_set( "display_errors", 0);
$str=strtolower($_GET["keyword"]);
```

```
$str2=str_replace("script", "", $str);
$str3=str_replace("on", "", $str2);
$str4=str_replace("src", "", $str3);
echo "<h2 align=center>Hello ".htmlspecialchars($str)."</h2><center>
<form action=xss_test.php method=GET>
<input type=submit name=submit value=Submit />
<input name=keyword value='".htmlspecialchars($str4)."'>
</form>
</center>";
?>

</body>
</html>
```

6.进入127.0.0.1/xss_test.php



7.弹出弹窗，正确运行

it&keyword=

127.0.0.1 显示

XSS

确定

取消

成功输出XSS弹窗，说明正确运行

8.代码分析

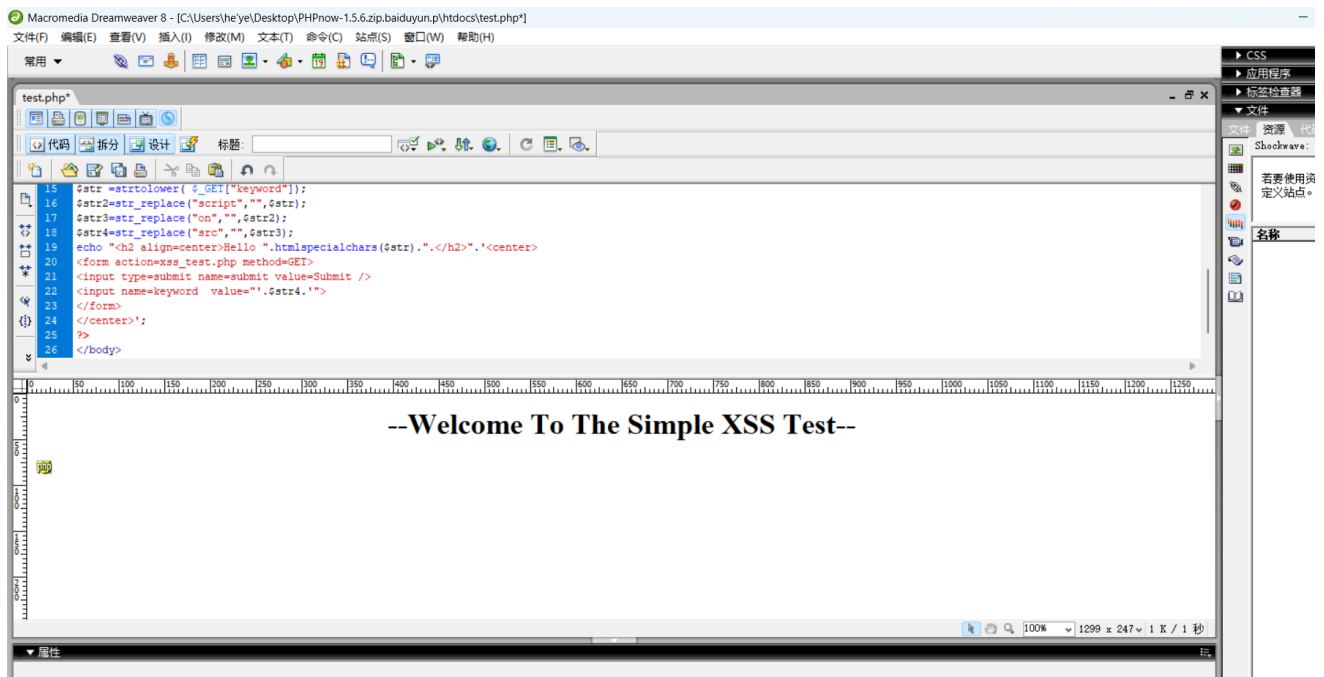
这段代码是一个包含PHP脚本和JavaScript的HTML页面，用于演示XSS测试。

页面加载时，会重新定义alert函数为confirm对话框，并关闭错误报告。用户通过GET请求提交的“keyword”参数经过小写转换和移除敏感词处理，然后通过htmlspecialchars函数进行HTML实体编码以防止XSS攻击，最后显示在页面上。

页面还包含一个表单，允许用户再次提交输入。此外，页面中的一个img标签试图利用onerror事件执行JavaScript代码，这是一个XSS攻击向量。

二、script方式

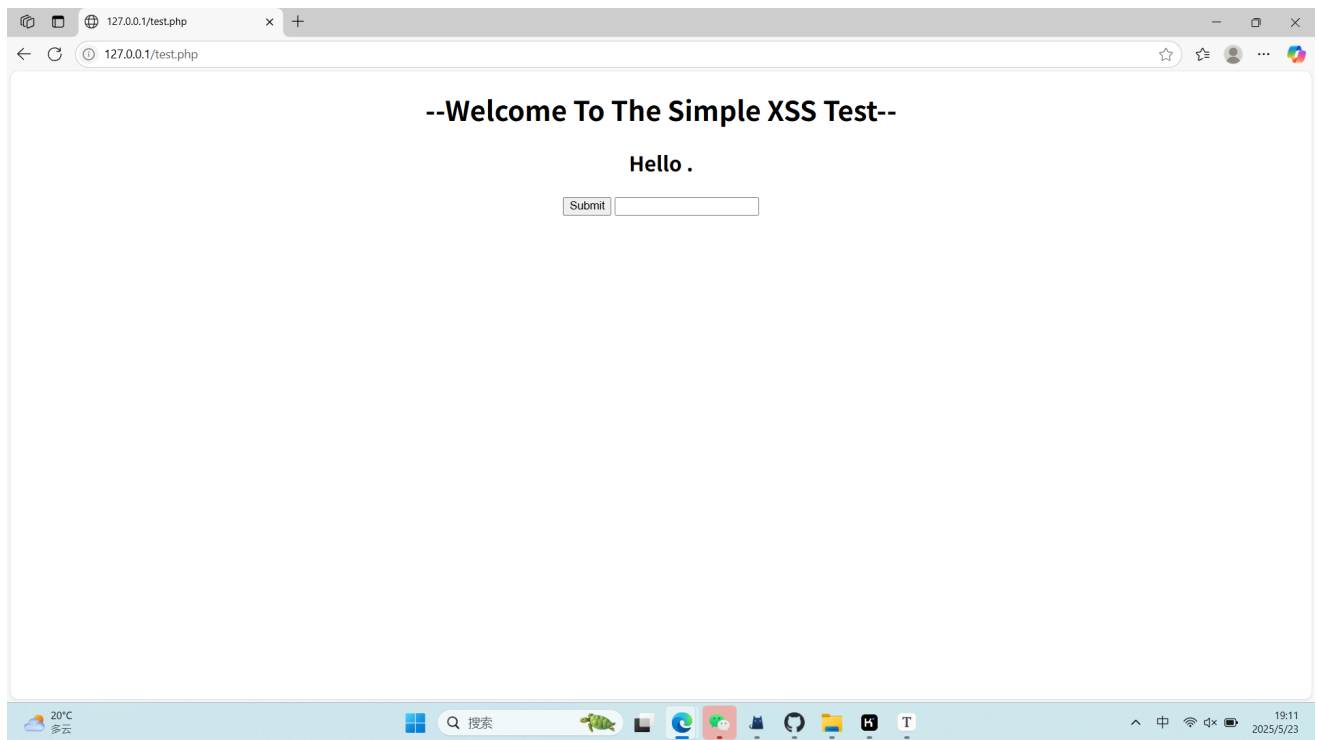
1.编写xss_test.php



```
<!DOCTYPE html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"> <script>
window.alert = function()
{
    confirm("Congratulations~");
}

</script>
</head>
<body>
<h1 align=center>--Welcome To The Simple XSS Test--</h1>
<?php
ini_set("display_errors", 0);
$str = strtolower( $_GET["keyword"]);
$str2=str_replace("script","", $str);
$str3=str_replace("on","", $str2);
$str4=str_replace("src","", $str3);
echo "<h2 align=center>Hello ".htmlspecialchars($str)."</h2>". '<center>
<form action=xss_test.php method=GET>
<input type=submit name=submit value=Submit />
<input name=keyword value="'. $str4. '">
</form>
</center>';
?>
</body>
</html>
```

2.进入127.0.0.1/xss_test.php



3.在页面输入xss脚本

```
<script>alert('xss')</script>
```

--Welcome To The Simple XSS Test--

Hello .

Submit

4.submit后，页面输出

--Welcome To The Simple XSS Test--

Hello <script>alert('\xss\')</script>.

Submit

5.页面输入双写关键字脚本

```
"><script>alert('XSS')</script><!--
```

--Welcome To The Simple XSS Test--

Hello <script>alert(\\'xss\\')</script>.

Submit <script>alert('xss')</script>

6.没有congratulations弹窗

--Welcome To The Simple XSS Test--

Hello <script>alert(\\'xss\\')</script>.

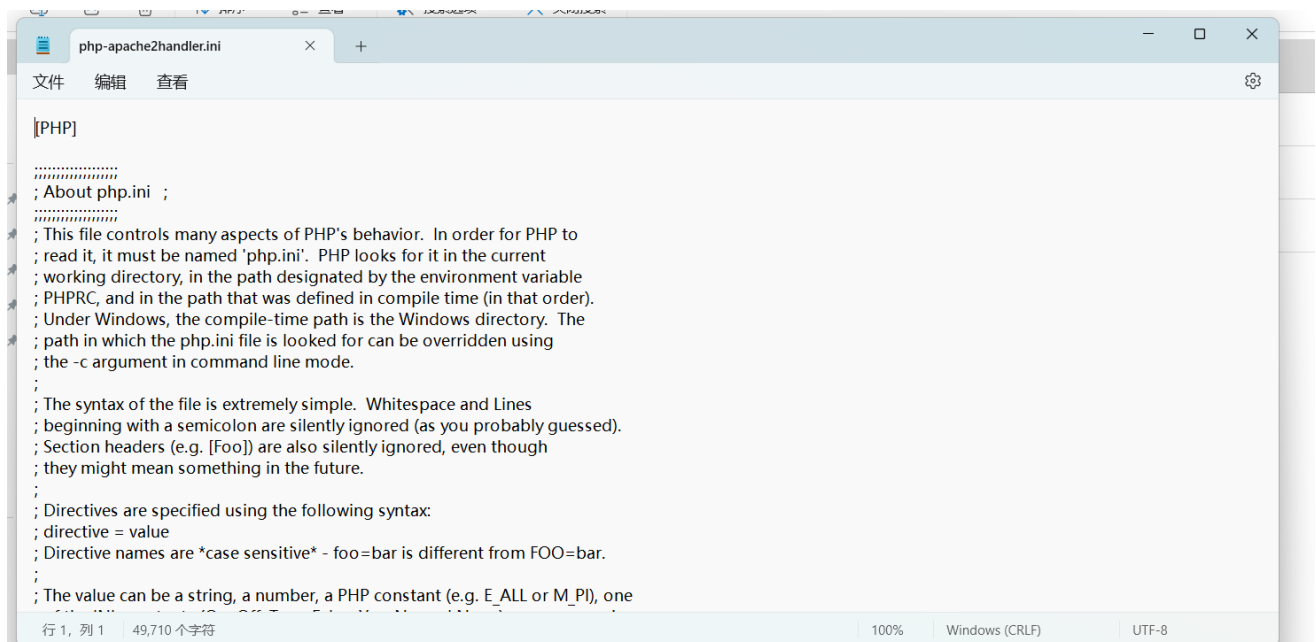
Submit <script>alert(\\'xss\\')</script>

7.分析代码

```
4 window.alert = function()
5 {
6     confirm("Congratulations~");
7 }
8
18 $str4=str_replace("src","", $str3);
19 echo "<h2 align=center>Hello ".htmlspecialchars($str).
20 <form action=xss_test.php method=GET>
21 <input type=submit name=submit value=Submit />
22 <input name=keyword value="'. $str4. '">
23 </form>
24 </center>';
```

JavaScript部分重定义了 alert 函数，使其显示一个 confirm 对话框。PHP部分处理用户输入，通过移除敏感词和HTML实体编码来防止XSS攻击，然后显示处理后的文本并构建一个表单供用户提交新的输入。页面中的一个 img 标签包含一个无效的 src 属性，旨在触发 onerror 事件执行XSS代码。

8.将apache2handler.ini中的magic_quotes_gpc 设置为Off



; Magic quotes for incoming GET/POST/Cookie data.

magic_quotes_gpc = On

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.

magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with " instead of \').

magic_quotes_sybase = Off

magic_quotes_gpc 是 PHP 中一个已经废弃的特性，它在处理从 GET、POST 和 COOKIE 传递的数据时，自动将某些特殊字符（如单引号、双引号、反斜杠和 NUL 字符）转义为它们的对应转义序列，以防止跨站脚本（XSS）攻击。

9.再次submit

--Welcome To The Simple XSS Test--

Hello .

10.弹窗congratulations弹窗，正确运行

127.0.0.1 显示

Congratulations~

确定

取消

运行成功

11.代码逻辑分析

(1)、禁用错误显示

```
ini_set("display_errors", 0);
```

这行代码关闭了 PHP 的错误报告功能，以避免在页面上显示错误信息，这可能会泄露敏感信息。

(2)、获取用户输入：

```
$str = strtolower($_GET["keyword"]);
```

(3)、移除危险字符串：

```
$str2 = str_replace("script", "", $str);  
$str3 = str_replace("on", "", $str2);  
$str4 = str_replace("src", "", $str3);
```

这三行代码使用 `str_replace` 函数移除可能用于 XSS 攻击的字符串（`script`、`on` 和 `src`）。这是为了防止用户输入包含这些关键字的脚本代码。

(4)、输出处理过的输入：

```
echo "<h2 align=center>Hello ".htmlspecialchars($str)."</h2>";
```

使用 `htmlspecialchars` 函数对处理过的字符串进行 HTML 实体编码，以防止任何剩余的特殊字符被解释为 HTML 或 JavaScript 代码。这样可以进一步增强安全性。

(5)、构建表单：

```
echo '<center>  
<form action=xss_test.php method=GET>  
<input type=submit name=submit value=Submit />  
<input name=keyword value="'.htmlspecialchars($str4).'">  
</form>  
</center>';
```

构建一个表单，允许用户再次提交输入。表单的 `action` 属性设置为当前脚本（`xss_test.php`），这意味着表单提交将重新加载同一页面，并显示新的输入结果。

(6)、重新定义 alert 函数

```
<script>
window.alert = function()
{
    confirm("Congratulations~");
}
</script>
```

这段 JavaScript 代码重新定义了 window.alert 函数，使其调用 confirm 函数。这意味着任何尝试调用 alert 的脚本都会显示一个确认对话框，而不是弹出一个警告框。

(7)、总结

这段PHP代码创建了一个简单的XSS测试页面，通过GET请求获取用户输入的“keyword”，对其进行小写转换和关键字移除处理，然后使用htmlspecialchars函数进行HTML实体编码以防止XSS攻击，最后将处理后的结果显示在页面上。页面中还包含一个表单，允许用户再次提交输入。同时，JavaScript代码重新定义了window.alert函数，使其调用confirm对话框显示“Congratulations”。

12.测试用例设计

(1)、正常输入

用例1：输入简单的文本，如“Hello”

预期结果：页面显示“Hello”，没有执行任何脚本。

(2)、特殊字符

用例2：输入包含特殊字符的字符串，如“!@#\$\$%^&*()”

预期结果：页面显示这些特殊字符，没有执行任何脚本。

(3)、大小写混合

用例3：输入大小写混合的字符串，如“HeLLo”

预期结果：页面显示“hello”，没有执行任何脚本。

(4)、空输入

用例4：不提供keyword参数

预期结果：页面显示默认问候语，如“Hello ”。

(5)、长输入

用例5：输入非常长的字符串

预期结果：页面正确显示处理后的长字符串，没有性能问题。

(6)、XSS攻击向量

用例6：输入尝试执行XSS攻击的字符串，如"<script>alert('xss')</script>"

预期结果：页面显示“alert('XSS')”，没有执行脚本。

(7)、关键词绕过

用例7：尝试使用变体或编码来绕过关键词过滤，如scr1pt

预期结果：页面不显示任何脚本代码，只显示处理后的文本。

(8)、JavaScript功能

用例8：输入触发JavaScript的字符串，如";alert('xss');"

预期结果：页面显示“alert('XSS')”，但不会执行脚本。

(9)、表单提交

用例9：提交表单并输入不同的值

预期结果：每次提交后，页面正确显示处理后的输入。

(10)、边界字符

用例10：输入包含边界字符的字符串，如"!@#\$\$%^&*()_+==[]{}|;:'\"",./<>?"

预期结果：页面显示这些边界字符，没有执行任何脚本。

13.测试结果

--Welcome To The Simple XSS Test--

Hello !@#\$\$%^&*()_+==[]{}|;:'\"",./<>?.

--Welcome To The Simple XSS Test--

Hello ;alert(\\'xss\\');

测试结果表明，系统对各种输入的处理均符合预期，包括正常文本、特殊字符、大小写混合、空输入、长输入、XSS攻击向量、关键词绕过、JavaScript功能触发、表单提交以及边界字符。系统能够有效防止XSS攻击，正确显示处理后的输入，展现出良好的安全性和稳定性。

心得体会：

通过本次实验，我对XSS攻击有了更深刻的理解。我学习了如何利用 `img` 和 `script` 标签执行XSS攻击，并测试了系统对不同攻击向量的防护能力。实验过程中，我尝试了多种方法来绕过系统的防护措施，包括使用特殊字符和编码技巧。尽管系统能够防御一些基础攻击，但在面对更复杂的攻击时仍有不足。这让我意识到，网络安全防护是一个持续的过程，需要不断更新和完善策略。

总结来说，这次实验不仅增强了我的安全意识，也让我掌握了实用的安全防护技巧。我认识到，作为软件开发者，了解和防范安全漏洞至关重要。在未来的开发工作中，我将更加注重代码的安全性，采取更多措施保护用户数据和隐私。