# 《软件安全》实验报告

姓名：李雅帆　　学号：2213041　　班级：信安班

## 一、实验名称：

跨站脚本攻击

## 二、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。

## 三、实验过程：

1.运行示例代码。

```
<!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>
```

这段代码创建了一个简单的网页，显示一个标题和一个包含提交按钮的表单，接收用户输入的 keyword 参数。服务器端通过一系列字符串替换来尝试去除 keyword 中的 script、on、src 等潜在的 XSS 攻击代码，处理后的输入通过 htmlspecialchars 转义后回显到网页上，并生成一个带有处理后输入的新的表单，供用户再次提交数据。

运行代码结果如下：

**--Welcome To The Simple XSS Test--**

2.从黑盒测试角度来进行实验。

访问 URL：http://127.0.0.1/xss_test.php

页面显示效果如下：


输入 XSS 脚本：<script>alert('xss')</script>来进行测试，点击 Submit 按钮


结果发现 Hello 后面出现了我们输入的内容，并且输入框中的回显过滤了 script 关键字，这个时候考虑后台只是简单的一次过滤。于是可以利用双写关键字绕过，构造脚本：<scrscriptipt>alert(‘xss’)</scscriptript>测试，效果如下：


虽然输入框中的回显确实是我们想要攻击的脚本，但是代码并没有执行在黑盒测试情况下，我们并不能看到全部代码的整个逻辑，所以无法判断问题到底出在哪里。

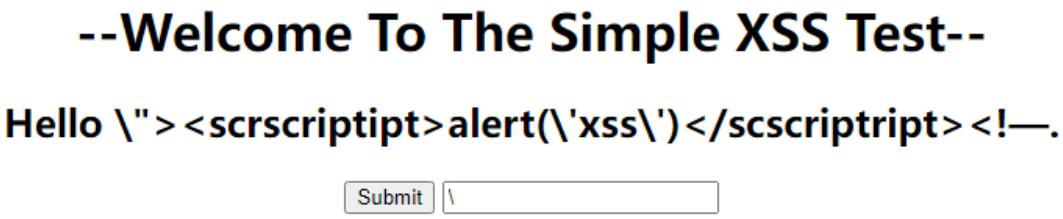
这时可以在页面点击右键查看源码，尝试从源码片段中分析问题。右键源码如下：

```
<!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>
<h2 align=center>Hello &lt;scrscriptipt&gt;alert(\'xss\')&lt;/scscriptript&gt;.</h2><center>
<form action=xss_test.php method=GET>
<input type=submit name=submit value=Submit />
<input name=keyword value="<script>alert(\'xss\')</script>">
</form>
</center>
</body>
</html>
```

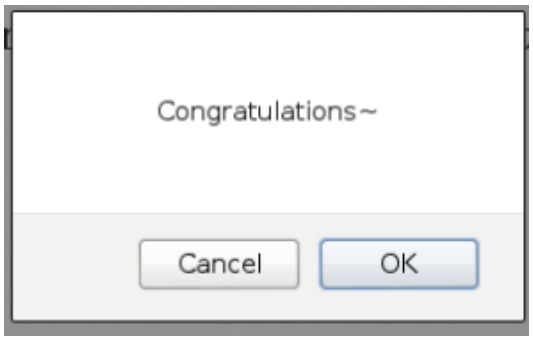
查看 16 行的<input>标签，发现这是唯一能输入且有可能控制的地方。这时候的思路就是想办法将前面的<input>标签闭合，于是构造脚本：

"><scrscriptipt>alert('XSS')</scscriptript><!—

此时弹出：




在 phpnow 安装目录下搜索文件 php-apache2handler.ini，并将“magic_quotes_gpc = On”设置为“magic_quotes_gpc = Off”，弹出确认框，执行成功：




3.看源码，虽然实现了基础的功能，也有没测试到的部分，Hello 后面

显示的值是经过小写转换的。输入框中回显值的过滤方法是将 script、on、src 等关键字都替换成了空。

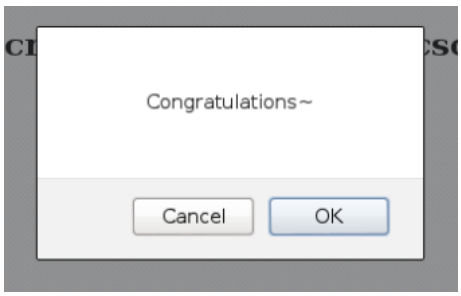
在这里给出一种基于<img>标签的脚本构造方法：<img src=ops! onerror="alert('XSS')">

对于这行脚本：

<img src=ops! onerror="alert('XSS')">：这是一个图片（<img>）元素。其 src 属性指向一个不存在或无效的路径（ops!）。浏览器会尝试加载图片，但由于路径无效，加载失败。

onerror="alert('XSS')": 这是 onerror 事件处理属性，专门用于处理加载失败的事件。当 src 属性指定的路径加载失败时，onerror 会被触发，并执行其值中的 JavaScript 代码。在这个例子中，它弹出了一个警告框（alert('XSS')），这是典型的 XSS 攻击手段，用于证明注入 JavaScript 成功执行。

此时弹出窗口：



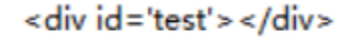

攻击成功。

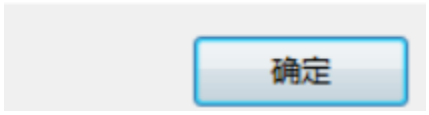
4.如果过滤掉<img></img>和<img />形式的字符串

```
<script type="text/javascript">
var htmlContent = "<div id='test'><img src='aaa' height='4' width='4'></img><img src='ff'
width='44' height='444' /></div>";
var data = htmlContent.replace(/<img.*>.*<\/img>/ig,"");
data = data.replace(/<img.*\/>/ig, "");
alert(data);
</script>
```

这段代码首先定义了一个包含两个图片标签的 HTML 字符串，然后通过两次正则替换操作依次删除含有闭合标签（<img ...></img>）和自闭合标签（<img ... />）的所有图片标签，最后弹出处理后的字符串内容。

弹出窗口：




## 四、心得体会：

通过实验，我学会了如何在 PHP 网页中实施 XSS 攻击，并实现简单的弹窗效果。我掌握了跨站脚本攻击的基本方法，包括利用<script>标签和<img>标签的 onerror 属性来注入并执行恶意代码。通过黑盒测试和源码分析，我了解了绕过简单字符串过滤的方法，如双写关键字和标签属性闭合。实验中还尝试了利用正则表达式过滤标签的方法，进一步提高了对 XSS 攻击的防御能力。

实验让我对 XSS 攻击的原理、绕过过滤机制的技巧以及有效的防御措施有了全面的理解和掌握。我深刻体会到 XSS 攻击的多样性和防御的复杂性。尽管实验中采用了一些基础的防御措施，但要全面防御 XSS 攻击，还需结合多种技术和策略，确保应用程序的安全性。实验不仅让我了解了 XSS 攻击的原理和方法，也提升了我在实际开发中防御此类攻击的能力。