《漏洞利用及渗透测试基础》实验报告

姓名： 赵悦蛟 学号：2313650 班级： 1071

**实验名称：**

XSS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。

**实验过程：**

1. 创建XSS攻击测试网站

安装 PHP 集成环境（如 PHPnow-1.5.6），确保本地服务器（Apache/Nginx）和 PHP 正常运行。在 PHPnow 安装目录下的htdocs文件夹中，新建文件xss\_test.php，用于搭建测试页面。

1. 编写测试页面代码

<!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>';

?>

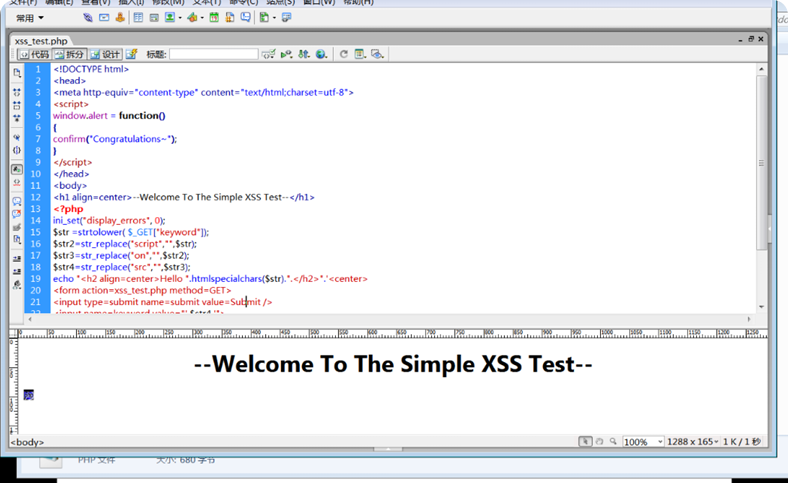
</body>

</html>

这段代码的主要功能是将用户输入的keyword参数转为小写（strtolower），并过滤script、on、src关键词（str\_replace）。其中使用htmlsp{insert\\_element\\_1\\_}ecialchars对输出内容进行编码，防止 HTML 注入，但输入框的值（str4）未编码，存在漏洞。

1. 访问测试页面

在浏览器中，输入网址http://127.0.0.1/xss\_test.php，确认页面正常显示，输入框和提交按钮功能可用。在输入框中输入123456，点击提交，观察页面回显Hello 123456，确认参数传递正常。





1. 使用 Script 标签实现 XSS 攻击
2. 黑盒测试

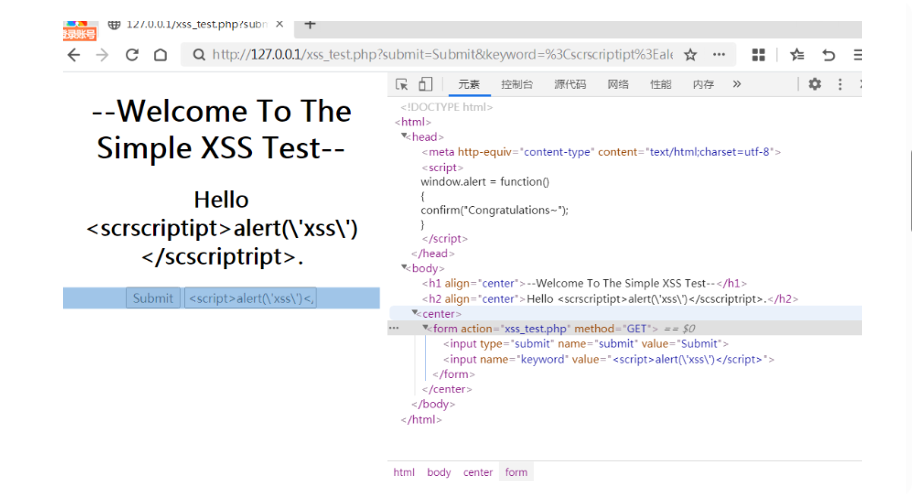
输入一个简单的XSS脚本：<script>alert（‘xss’）</script>提交，进行测试。



Hello 后面出现了输入的内容，并且输入框中的回显过滤了 script 关键字。接下来尝试利用双写关键字绕过过滤，构造脚本：<scrscriptipt>alert('xss')</scscriptript>结果：



页面回显脚本，但未执行。查看源码：



在源码中第5行通过JavaScript重写了`window.alert`函数，将其修改为触发`confirm("Congratulations~")`弹窗，因此若XSS攻击成功执行脚本，页面会弹出显示“Congratulations~”的确认框，这成为判断攻击是否成功的标志。 接着分析第16行的`<input>`标签代码，其`value`属性直接回显用户输入的内容。当尝试注入`<script>alert('xss')</script>`时，虽然表面上插入了`<script>`标签组，但由于整个输入内容被包裹在`<input name="keyword" value="...">`的`value`属性中，未使用`">`等符号闭合该属性标签，导致注入的脚本被限制在`value`属性的字符串范围内，仅作为文本内容回显在输入框中，无法脱离`<input>`标签的上下文环境。浏览器会将`value`属性内的内容视为普通文本，而非可执行的JavaScript代码，因此脚本无法触发执行，仅能看到输入内容的回显，无法实现攻击效果。这一现象揭示了XSS攻击中“标签闭合”的关键作用——若未正确突破目标标签的限制，即使绕过关键词过滤，脚本也无法获得执行上下文。

因此我们需要闭合<input>标签。构造新的脚本：

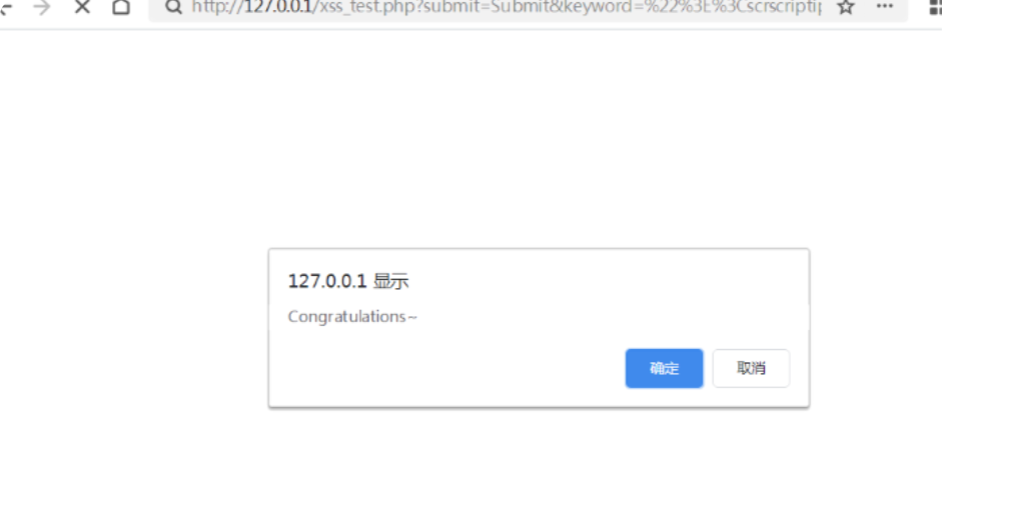
"><script>alert('XSS')</script><!--

">：闭合 <input> 标签的 value 属性，使后续脚本成为独立的 HTML 代码。

<script>...</script>：注入真正的脚本标签。

<!--：注释掉多余的 "（避免破坏页面结构）。

再次执行，可以看到XSS攻击成功：



1. 白盒测试

查看xss\_test.php源码中的 PHP 逻辑：



发现过滤仅针对script、on、src，且输入框value未编码，可通过闭合标签注入脚本。161718直接使用闭合标签脚本，确认与黑盒测试结果一致。

五．使用 img 标签实现 XSS 攻击

在源代码的基础上添加<img>标签：

<!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><?phpini\_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>';?><img src="non-exist" onerror="var payload=String.fromCharCode(97,108,101,114,116,40,39,88,83,83,39,41);eval(payload)"></body></html>

首先，PHP 代码从 URL 获取 "keyword" 参数并进行预处理，将其转换为小写后过滤掉 "script""on""src" 等可能用于 XSS 攻击的关键词，试图以此防范攻击。但在 PHP 代码执行完毕后，页面中插入了一个<img>标签，其src属性被设置为 "non-exist" 这样一个不存在的图片资源，目的是触发浏览器的onerror事件 —— 当图片加载失败时，该事件会被自动触发。

在onerror事件的处理代码中，首先定义了一个变量payload，通过String.fromCharCode()函数将一组 ASCII 码（97,108,101,114,116,40,39,88,83,83

,39,41）转换为对应的字符串，这串 ASCII 码对应的正是 "alert ('XSS')"。接着，通过eval(payload)执行该字符串，由于eval()函数会将传入的字符串作为 JavaScript 代码运行，因此最终会弹出带有 "XSS" 消息的警告框。

这种攻击方式利用了浏览器在图片加载失败时触发onerror事件的机制，绕过了 PHP 代码对部分关键词的过滤（如未过滤onerror事件本身及eval函数），通过编码转换和动态执行的方式注入并运行恶意脚本，从而实现跨站脚本攻击。其核心在于利用合法的 HTML 标签事件和 JavaScript 函数，绕过服务端的过滤逻辑，完成攻击代码的执行。

核心是：<img src="non-exist" onerror="var payload=String.fromCharCode(97,108,101,114,116,40,39,88,83,83,39,41);eval(payload)">利用图片加载失败触发onerror事件，通过eval执行alert('XSS')。

访问http://127.0.0.1/xss\_img.php，页面自动触发onerror事件，弹出Congratulations~确认框，攻击成功。



**心得体会：**

在本次实验中，我深入理解了跨站脚本攻击（XSS）的实际威胁与技术实现逻辑。通过亲手操作，我认识到XSS攻击的核心在于利用网页对用户输入过滤的缺陷——当应用未对动态内容进行严格校验或采用简单过滤策略时，攻击者便能借机注入恶意脚本。

例如在第一部分实验中，通过双写“script”关键词为“scrscriptipt”，成功绕过了服务端对“script”标签的过滤限制，使得alert('XSS')脚本得以执行。这一过程揭示了防御措施若存在逻辑漏洞（如仅做单次字符串替换），即便看似启用了过滤机制，仍可能被攻击者通过构造特殊 payload 轻易绕过。

实验的第二部分则展现了攻击手法的多样性。利用<img>标签的onerror事件，在图片资源加载失败时触发恶意代码执行，这种方式巧妙规避了对“script”“on”等关键词的过滤。通过ASCII码转义（String.fromCharCode）和eval函数动态执行脚本的技巧，进一步说明攻击者可通过编码转换、事件滥用等手段绕过常规防护。此次实践让我深刻体会到，XSS攻击不仅依赖于代码注入的机会，更考验对浏览器机制、编程语言特性的理解深度。

从安全防御角度反思，实验中暴露的过滤策略单一、输出编码不彻底等问题极具警示意义。