《漏洞利用及渗透测试基础》实验报告

姓名： 艾明旭 学号：2111033 班级：信息安全一班

**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：**

**对如下示例代码的 php 网页进行 XSS 攻击，实现简单的弹窗效果即可**

<!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>

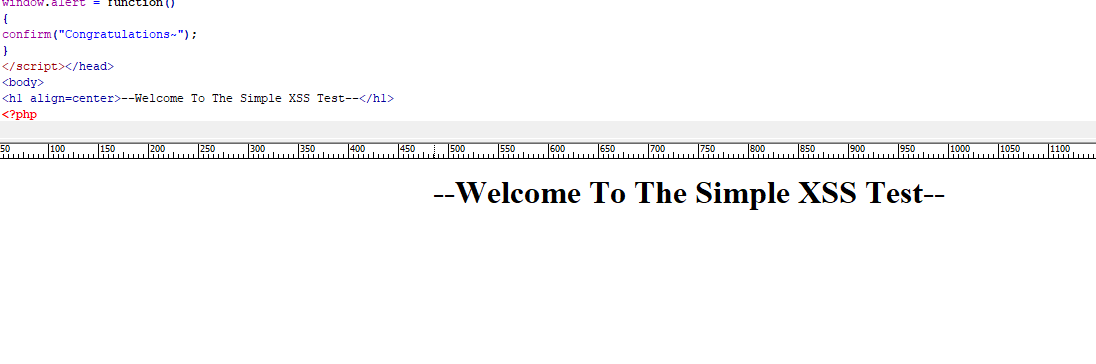
</center>';

?>

</body>

</html>

**代码运行效果：**



首先从黑盒测试的角度来进行实验

访问 URL：http://127.0.0.1/xss\_test.php

页面显示效果如下：



输入上面学过最简单的 XSS 脚本：<script>alert('xss')</script>来进行测试。点击 Submit 按钮以后，效果如下：



发现输入框中的回显过滤了script关键字。

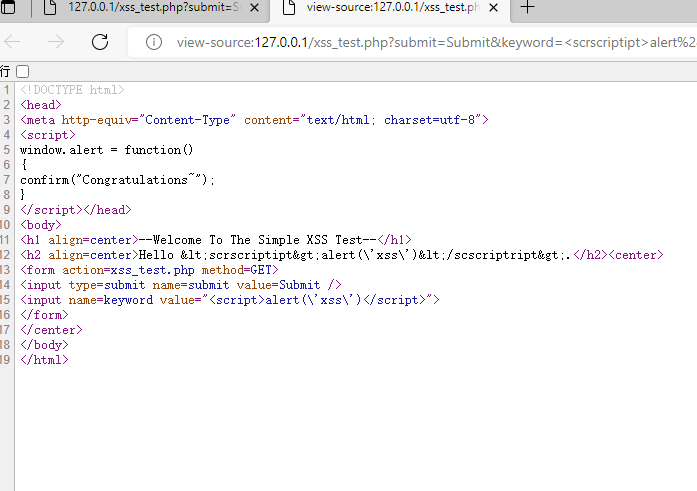
这个时候考虑后台只是最简单的一次过滤。于是可以利用双写关键字绕过，构造脚本：

<scrscriptipt>alert('xss')</scscriptript>测试。执行效果如下：



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：



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。

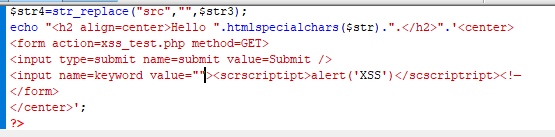
<input name=keyword value="<script>alert('xss')</script>">分析这行代码知道，虽然我们成功的插入了<script></script>标签组,但是并没有跳出 input 的标签，使得我们的脚本仅仅可以回显而不能利用。这个时候的思路就是想办法将前面的<input>标签闭合，于是构造如下脚本：

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

逻辑是用**">** 来闭合前面的**<input>**

标签。而 **<!--** 其实是为了美观，用来注释掉后面不需要的 **">** ,否则页面就会在输入框后

面回显 **">**



弹处确认框，XSS 攻击成功。执行效果如下：

发现出现错误，这是由于输入的双引号不能正常被处理导致的，因为

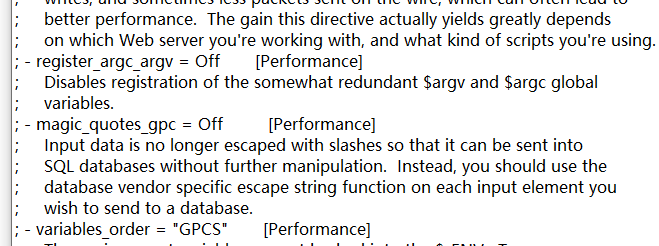
php 服务器自动会对输入的双引号等进行转义，以预防用户构造特殊输入进行攻击



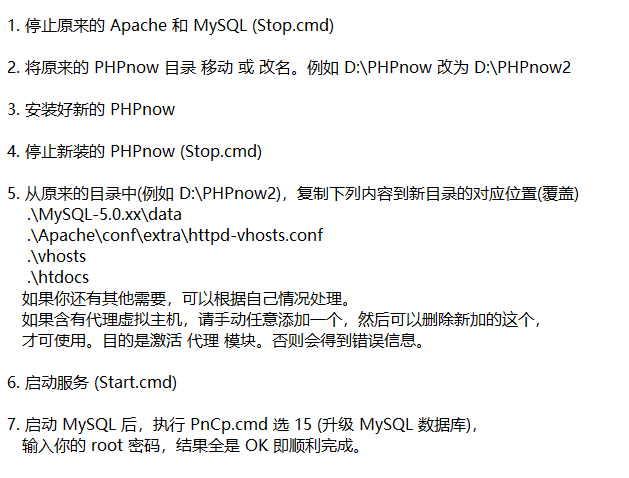
按照书上的方法，在 phpnow 安装目录下搜索文件

php-apache2handler.ini，并将“magic\_quotes\_gpc = On”设置为“magic\_quotes\_gpc = Off”

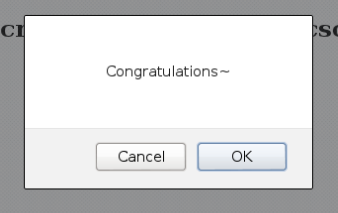
但并没有奏效



这时候要把php版本升级到5.4以上，升级方式如下



弹处确认框，XSS 攻击成功。执行效果如下：



接下来，从源码的角度来看一下页面的核心逻辑

<?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>';

?>

和上面黑盒测试的情况差不多，但是也有没测试到的地方。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。但由于过滤的内容并不是很多，因此攻击脚本的构造方法多种多样。

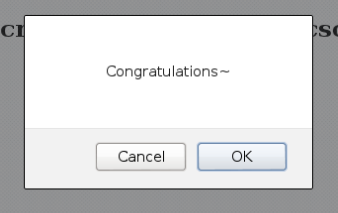
以下是<img>标签的一种脚本构造方法：

<img src=ops! onerror="alert('XSS')">

<img>标签是用来定义 HTML 中的图像，src 一般是图像的来源。而 onerror 事件会在文

档或图像加载过程中发生错误时被触发。所以上面这个攻击脚本的逻辑是，当 img 加载一个错误的图像来源 ops!时，会触发 onerror 事件，从而执行 alert 函数。

弹出弹窗表明攻击成功



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,"");   //过滤如<img></img>形式的图片元素

   data = data.replace(/<img.\*\/>/ig, "");   //过滤如<img />形式的元素

   alert(data);

</script>

效果：



**心得体会：**

通过这次实验，我掌握了从黑盒和白盒两个角度实现跨站脚本攻击，学会了php网页进行XSS攻击，实现简单的弹窗效果。