《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰　　学号：2113997　　班级：信安2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。有能力者可以自己撰写更安全的过滤程序。

实验过程：

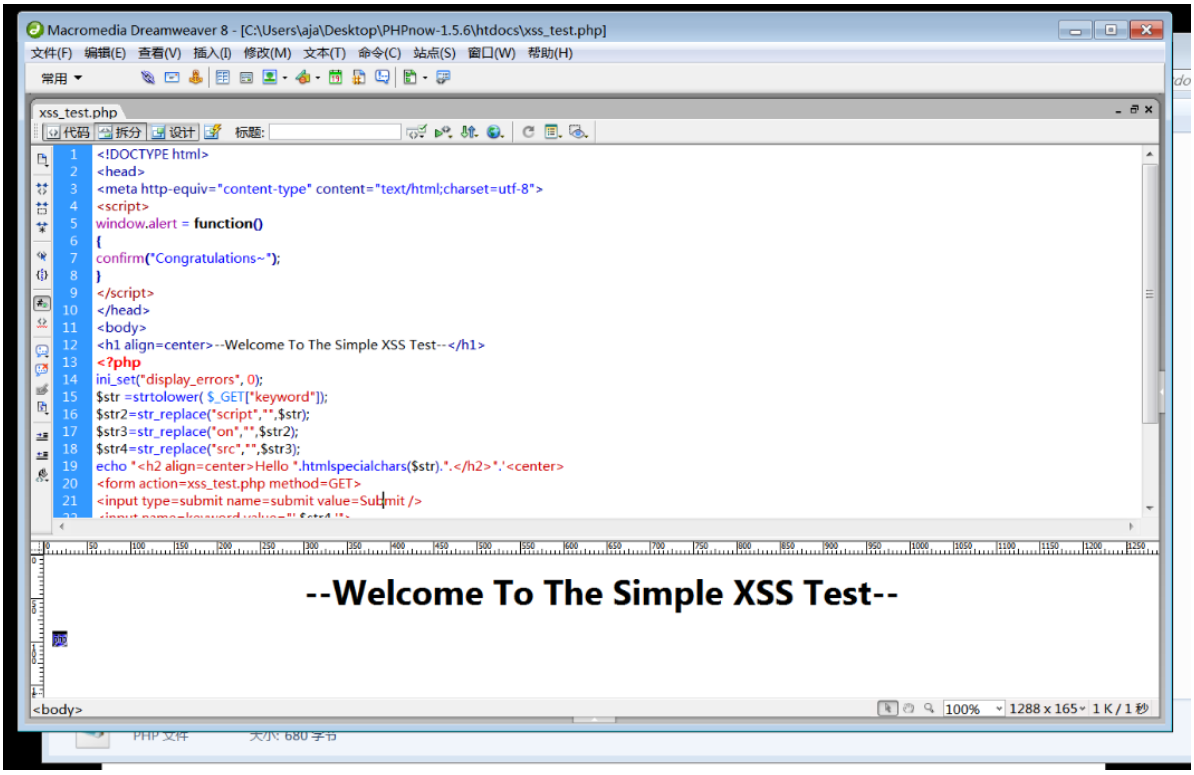
一、 创建XSS攻击测试网站

在`PHPnow-1.5.6/htdocs`目录下新建一个文件`xss_test.php`，写入如下代码：

```
<!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);
```

```
18 $str4=str_replace("src","",$str3);
19 echo "<h2 align=center>Hello ".htmlspecialchars($str).".
   </h2>".'<center>
20 <form action=xss_test.php method=GET>
21 <input type=submit name=submit value=Submit />
22 <input name=keyword value="'.$str4.'">
23 </form>
24 </center>';
25 ?>
26 </body>
27 </html>
```

结果如下图所示：



进入网址`http://127.0.0.1/xss_test.php`，成功进入，并能正常输入和显示：

二、 Script实现XSS攻击

下面将从黑盒和白盒两个角度来进行测试。

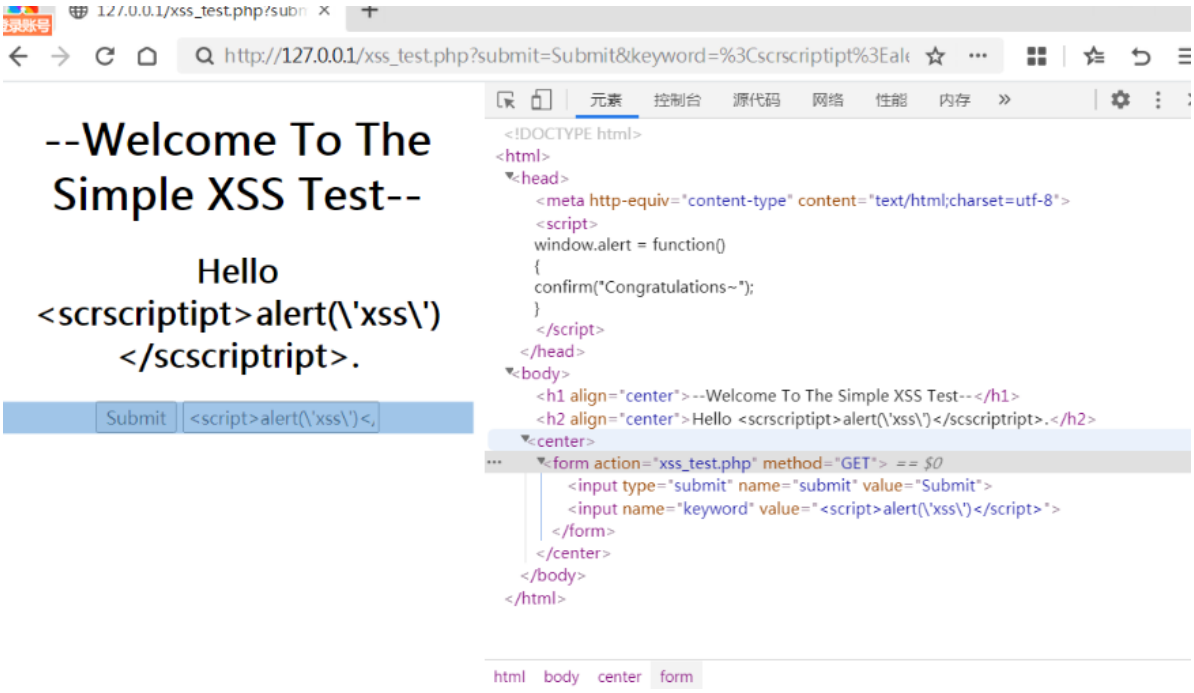
- **黑盒测试**

如图可以看到一个 `Submit` 按钮和输入框，并且还有标题提示 XSS。于是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：

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：

发现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。因为在黑盒测试情况下，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。这个时候我们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：



刚开始就会看到第 5 行重写的 `alert` 函数。如果可以成功执行 `alert` 函数的话，页面将会跳出一个确认框，显示 `Congratulations~`。这应该是我们 XSS 成功攻击的的标志。接着往下查看 16 行的 `<input>` 标签，我们唯一能输入且有可能控制的地方。

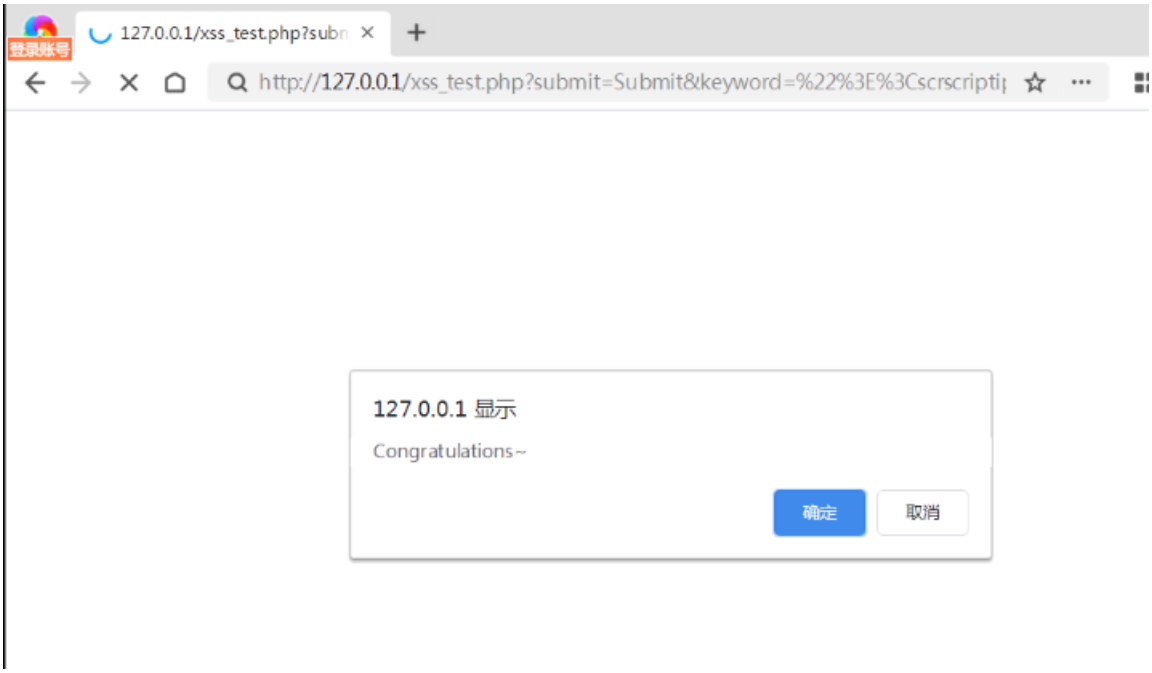
```
<input name=keyword value="<script>alert('xss')</script>">
```

分析这行代码知道，虽然我们成功的插入了 `<script></script>` 标签组,但是我们并没有跳出 `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><!--
```

> **解释：**此时，打开网站的源码可看到如下语句：
>
> ```
> <input name=keyword value=""><script>alert('xss')
> </script><!--">
> ```
>
> 其中，`">` 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 `">`

弹出确认框，**XSS 攻击成功**。执行效果如下：



- **白盒测试**

我们从源码的角度来看一下页面的核心逻辑，源码如下所示：

```
<?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>';
?>
```

发现跟我们上面黑盒测试的情况差不多，但是也有没测试到的地方。比如，Hello后面显示的值是经过小写转换的。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。

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。*

三、 img方式实现XSS攻击

- 用`<img>`标签的脚本构造：

```
<img src="x" onerror="alert('XSS')">
```

> **解析：** `<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。

据此我们修改源码，将 `img` 标签嵌入其中，代码如下所示：

```
<!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='".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>
```

其中， 代码的核心部分如下所示：

```
<img src="non-exist" onerror="var
payload=String.fromCharCode(97,108,101,114,116,40,39,88,83,83,39
,41);eval(payload)">
```

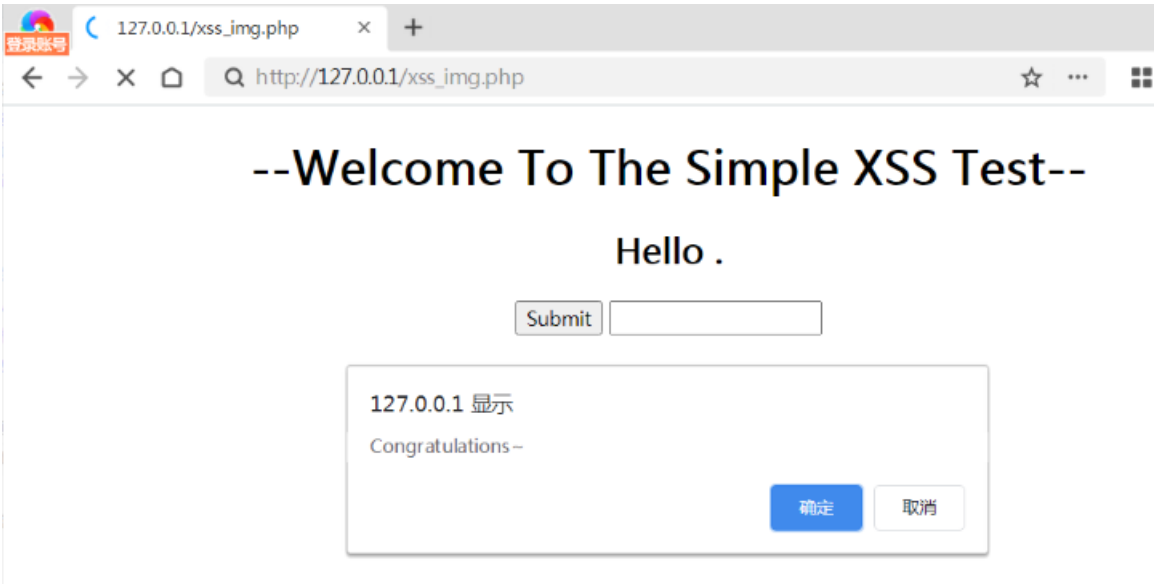
下面来解释这段代码。

> 1. 首先，PHP 代码从 URL 获取 "keyword" 参数，将其转化为小写，然后替换掉其中的 "script", "on", "src" 字符串，尝试防止 XSS 攻击。
> 2. 在 PHP 代码之后，插入一个 <img> 标签，其中 src="non-exist" 是一个不存在的图片资源，用来触发 onerror 事件。
> 3. onerror 事件在图像加载错误时会触发。在这个事件中，我们定义了一个变量 payload，并使用 String.fromCharCode() 函数生成一个字符串。
>
> String.fromCharCode(97,108,101,114,116,40,39,88,83,83,39,41) 会返回字符串 "alert('XSS')"。
> 4. 然后，我们使用 eval(payload) 执行这个字符串。 eval() 函数会执行传入的 JavaScript 代码。由于 payload 的值是 "alert('XSS')"，所以这将弹出一个带有 "XSS" 消息的警告框。

这是一种利用 `onerror` 事件和 `eval()` 函数来实现 XSS 攻击的方式。通过这句核心代码，我们就能实现以 `img` 的方式完成一次跨站脚本攻击。

- **运行结果**

将上述代码保存为 `xss_img.php`，进入网址，运行效果如下：



心得体会：

在这个实验中，我了解到了跨站脚本攻击（XSS）的潜在危险性以及执行这种攻击的一些技术。XSS 攻击发生在当恶意攻击者找到方式在网页中注入未经过滤的或者错误过滤的脚本时。

在实验的第一部分，我利用 `script` 标签来注入 JavaScript 代码。通过巧妙地插入"scrscriptipt"来避开关键词"script"的过滤，成功插入了 `alert('XSS')` 脚本，这表明即使使用了某种过滤措施，也可能因为方法简单或者疏忽导致防御失效。

在实验的第二部分，我学习了如何利用 `<img>` 标签的 `onerror` 事件来执行 JavaScript 代码。当图片资源加载错误时，就会触发这个事件，从而运行注入的代码。这种方法绕过了简单的字符串替换过滤。