《漏洞利用及渗透测试基础》实验报告

**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。

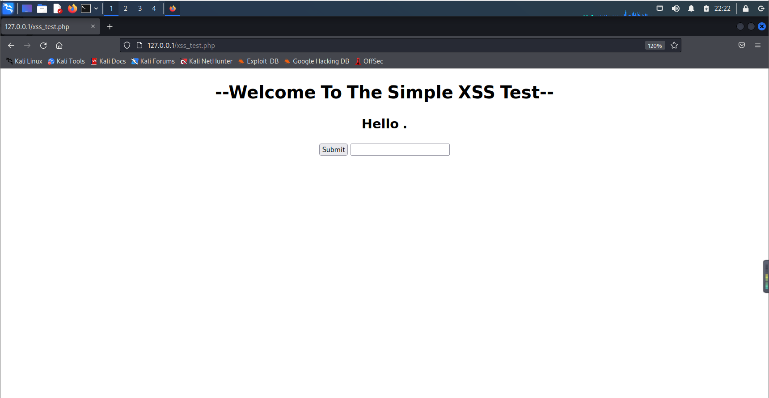
**实验过程：**

1.编写测试代码xss\_test.php如下：

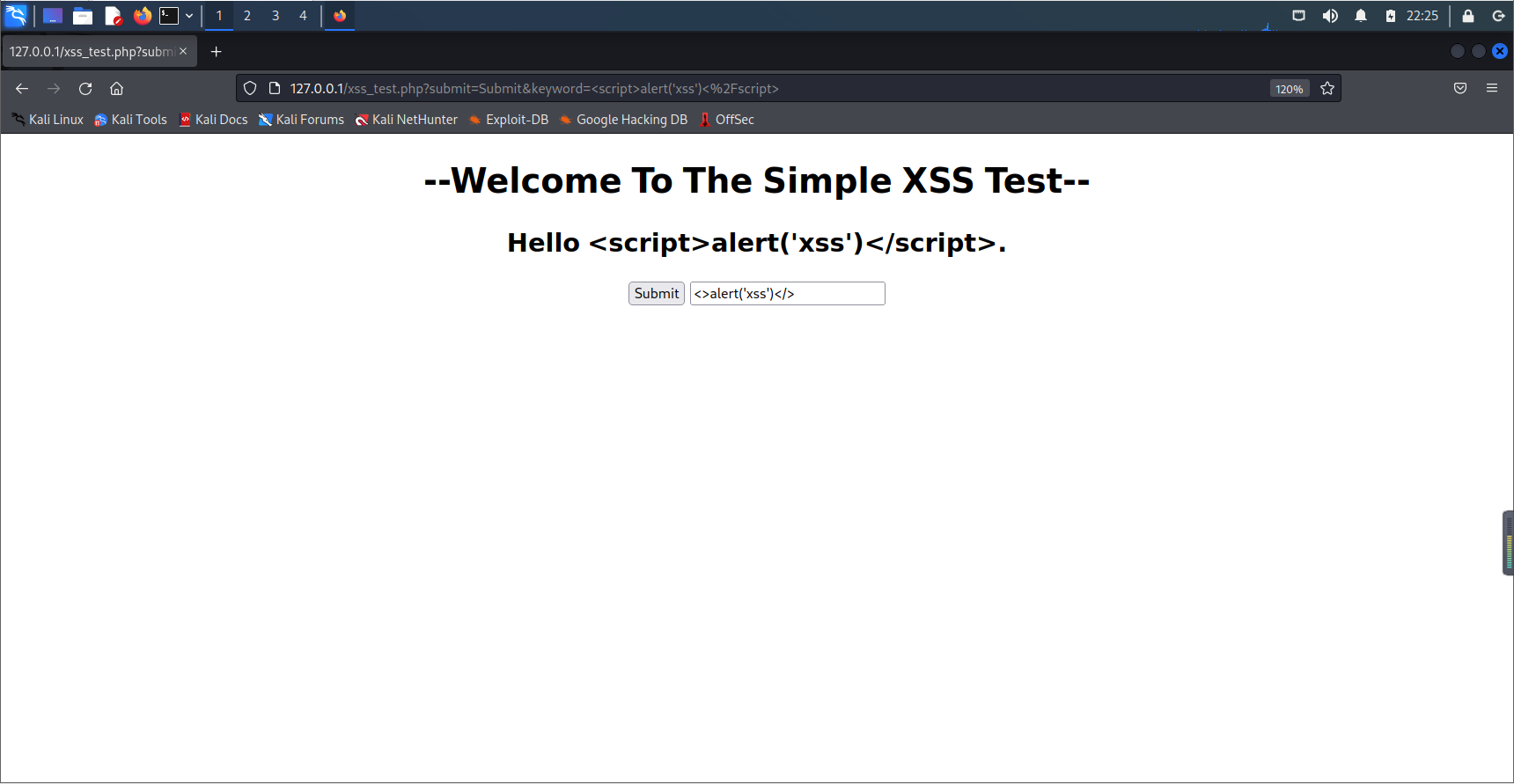
|  |
| --- |
| <!DOCTYPE html>  <head>  <meta http-equiv="content-type" content="text/html;charset=utf-8">  <script>  window.alert = function()  {  confirm("Congratulations~");  }  </script>  </head>  <body>  <h1 align=center>--Welcome To The Simple XSS Test--</h1>  <?php  ini\_set("display\_errors", 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> |

2.使用script方式实现跨站脚本攻击:

**在黑盒测试角度下：**

访问URL：<http://127.0.0.1/xss_test.php>，显示效果如下：

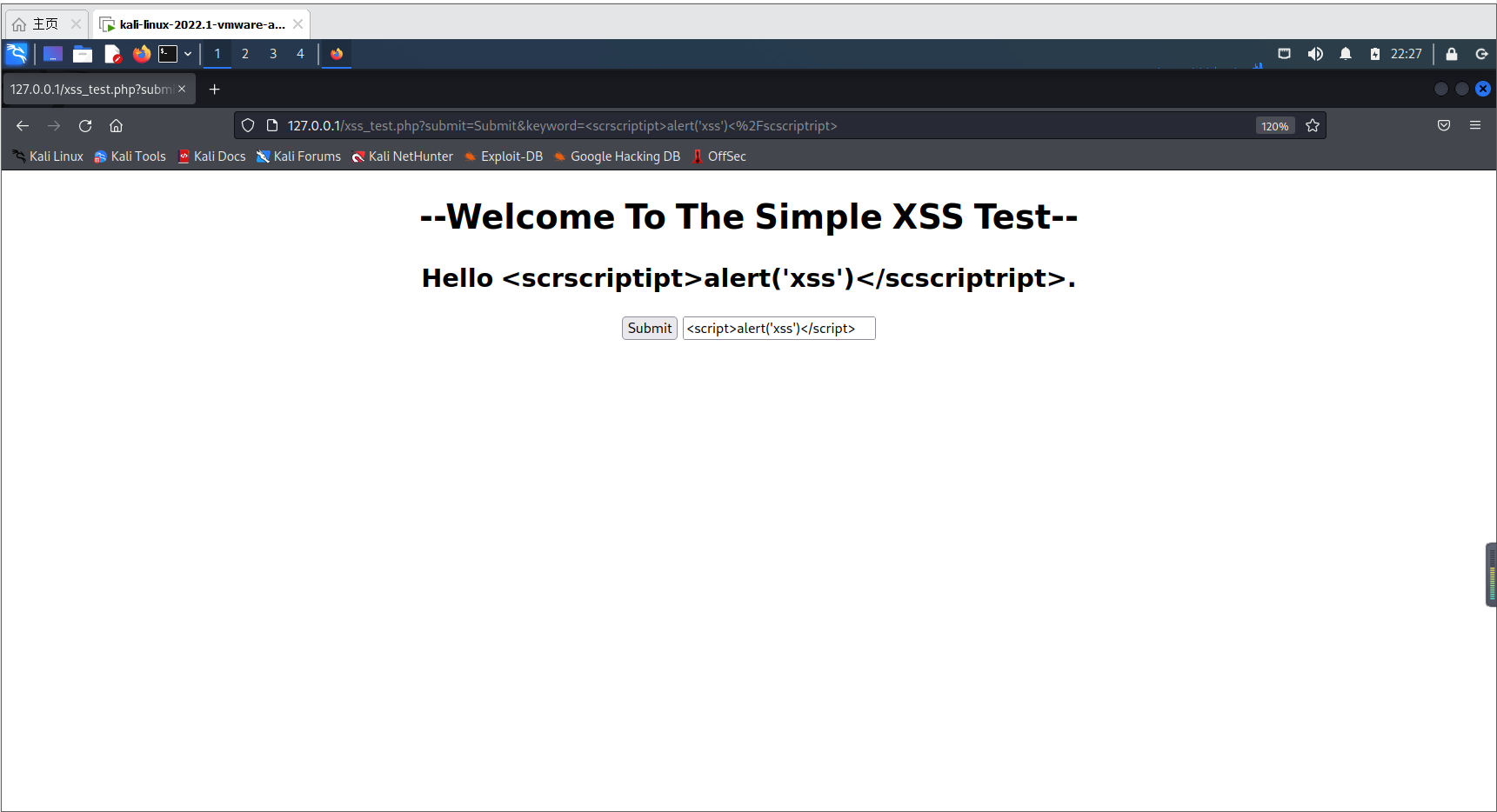
（1）如图可以看到一个 Submit 按钮和输入框，并且还有标题提示 XSS。输入较为简单的XSS 脚本：<script>alert('xss')</script>来进行测试。点击 Submit 按钮以后，效果如下：



结果发现Hello后面出现了输入的内容，并且输入框中的回显过滤了script关键字，此时考虑后台只是最简单的一次过滤。

（2）可以利用双写关键字绕过，构造脚本<scrscriptipt>alert('xss')</scscriptript>

进行测试。执行效果如下：



发现虽然输入框中的回显确实是想要攻击的脚本，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。这时我们尝试从源码片段中分析问题。

**在白盒测试角度下：**

分析源码，源码第5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。

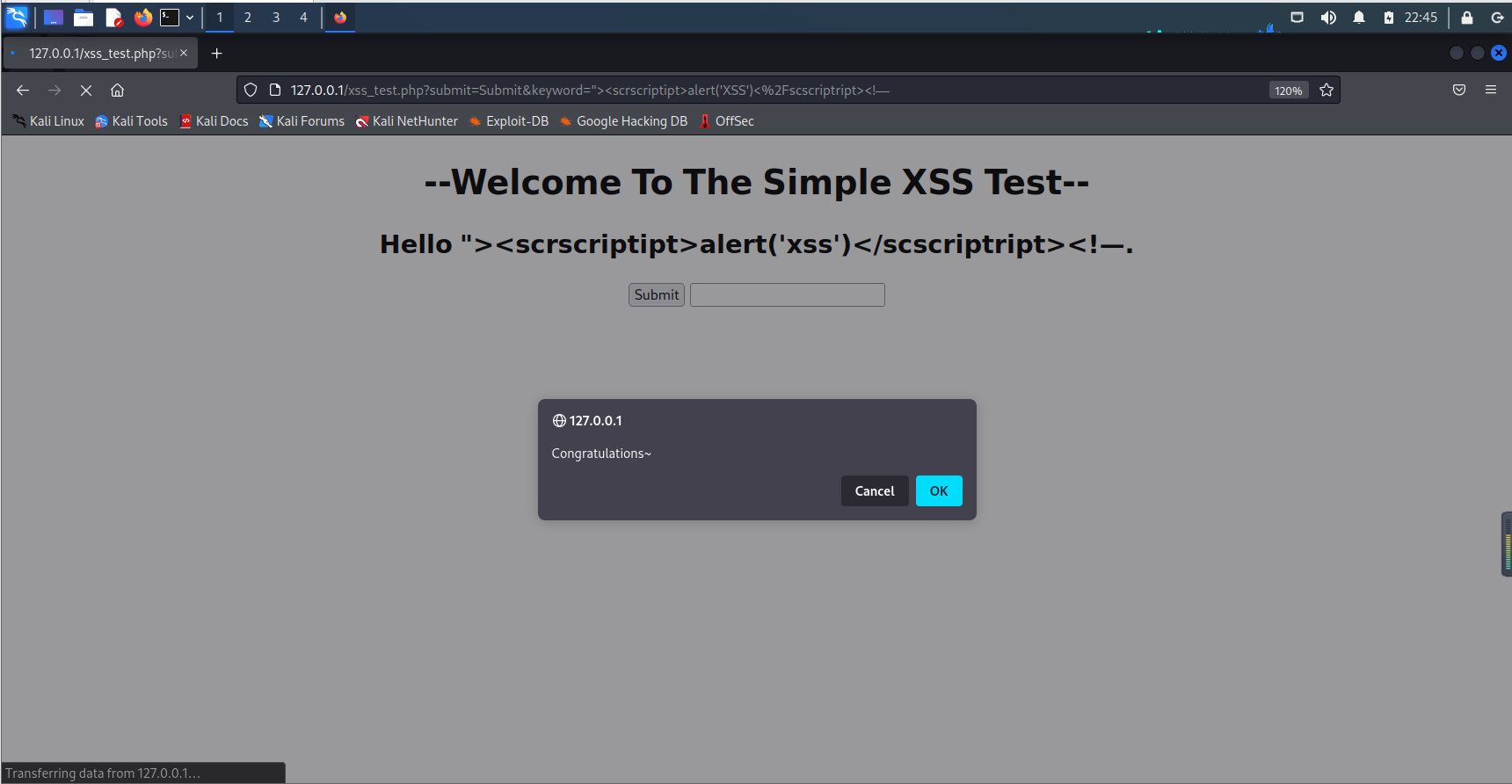
查看源码第16 行的<input>标签，此处为我们唯一能输入且有可能控制的地方：

<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，使得输入脚本仅仅可以回显而不能利用。此时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><!—

弹处确认框，XSS 攻击成功。执行效果如下：



此时再次查看页面源码，仔细查看第16行代码执行的逻辑：

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

如上所示，其中粗体是我们成功构造的脚本，"> 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 ">。

接下来，我们从源码的角度来看一下页面的核心逻辑。

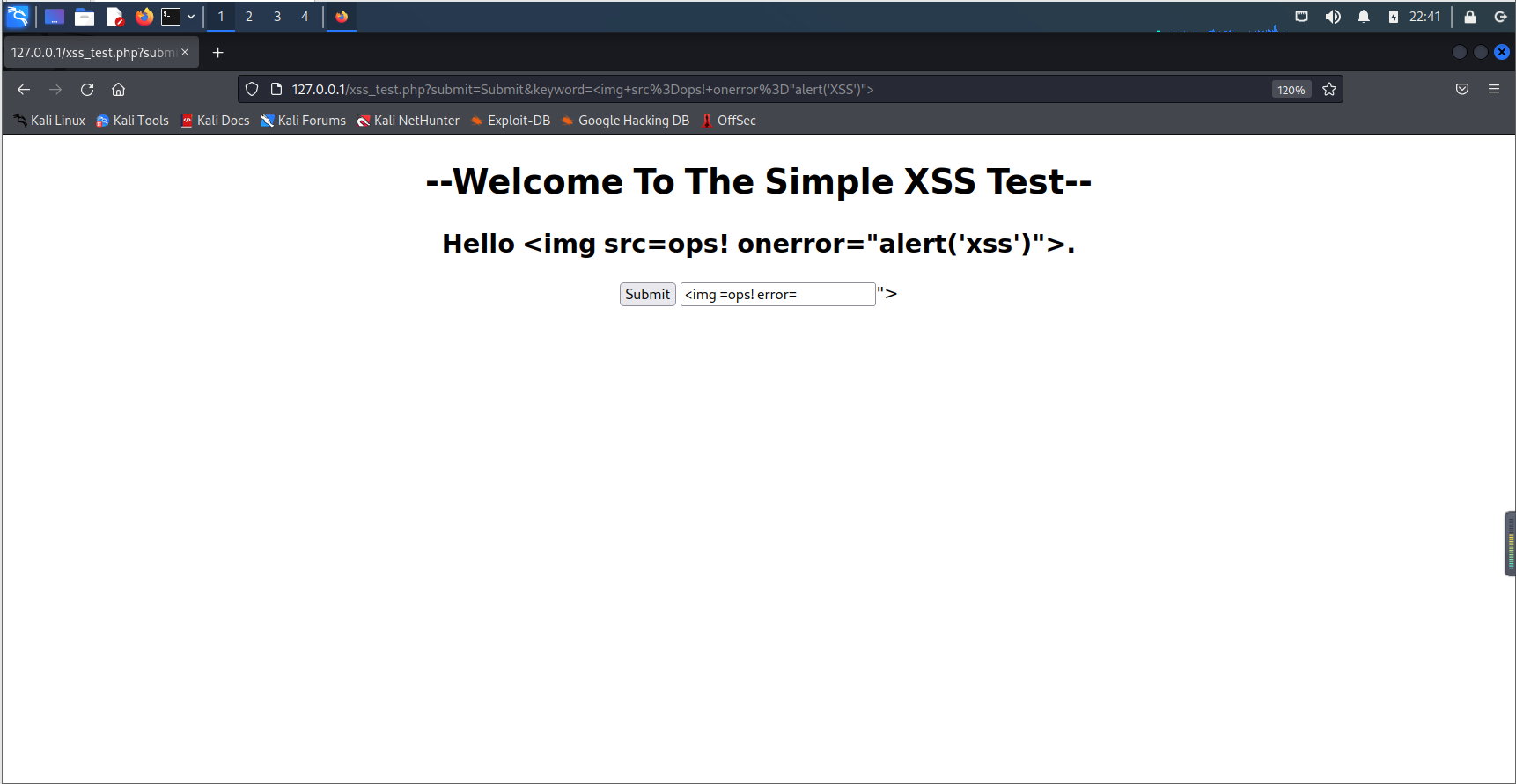
|  |
| --- |
| <?php  ini\_set("display\_errors", 0);  $str =strtolower( $\_GET["keyword"]);  $str2=str\_replace("script","",$str);  $str3=str\_replace("on","",$str2);  $str4=str\_replace("src","",$str3);  echo "<h2 align=center>Hello ".htmlspecialchars($str).".</h2>".'<center>  <form action=xss\_test.php method=GET>  <input type=submit name=submit value=Submit />  <input name=keyword value="'.$str4.'">  </form>  </center>';  ?> |

发现跟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。这也会导致攻击脚本的构造方法多种多样。

3.使用img方式实现跨站脚本攻击:

使用<img>标签构造脚本如下：<img src=ops! onerror="alert('XSS')">，可以看到效果如下：



<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。

**心得体会：**

通过实验，了解了跨站脚本攻击是当前web应用中最危险和最普遍的漏洞之一，掌握了XSS 通常用于发动 cookie 窃取、恶意软件传播(蠕虫攻击)、会话劫持、恶意重定向等，此外一般情况下存储式 XSS 的风险会高于反射式 XSS。