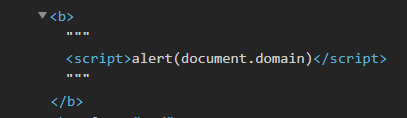
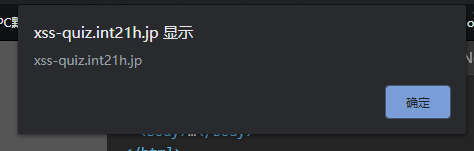
**XSS Challenges解题报告**

1. 题目简述：在输入框输入字符串，点击按钮，会将输入的内容显示在页面上。

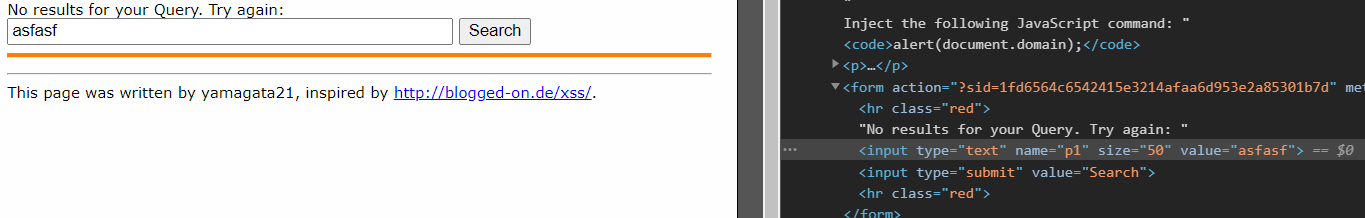


漏洞：如果输入的是相应的代码语言，浏览器会将其视为代码并运行。

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

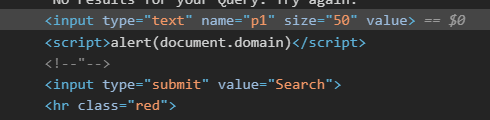


1. 题目简述：在输入框输入字符串，点击按钮，会将输入的内容放入input标签的value属性中。



漏洞：若输入的代码先将input标签闭合，然后加入需要运行的代码，再将尾部的“>”注释掉，则能达到目的。

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

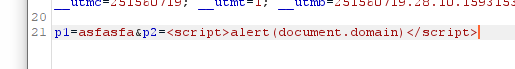


1. 题目简述：表面上与第一题相似，会将输入的内容显示在页面上，但在使用burp抓包之后发现，会将<>等非法符号转码，是的浏览器不会讲其当作代码来运行。

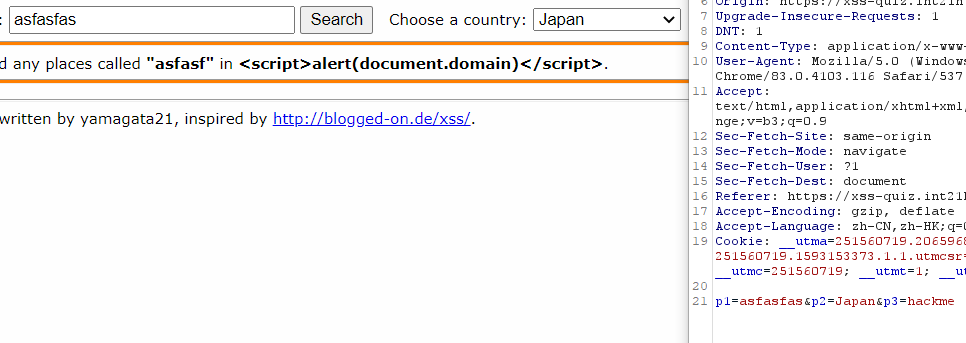


漏洞：除了p1这个参数之外，还会讲p2这个参数显示在页面上，且并未对p2这个参数进行检测，则通过抓包将p2修改，便可达到目的。

输入：将p2修改为<script>alert(document.domain)</script>



1. 题目简述：此题与第3题类似，会将输入的内容显示出来，通过抓包发现，一共有三个参数。而第三个参数的值会放入input标签的value。

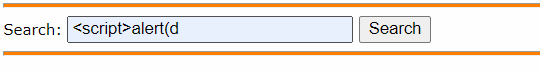


漏洞：虽然前两个参数均会对字符串进行检测，但第3个参数未进行，则对第三个参数进行修改。

输入：将p3修改为"/><script>alert(document.domain)</script><!

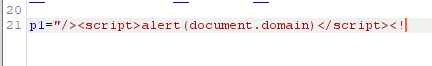


1. 题目简述：此题对输入的内容的长度进行了限制，使得无法直接输入长长的代码进行注入，且输入的内容会变为input的value的值。



漏洞：通过抓包之后发现，在包内将参数的值进行修改，就可以成功运行。

输入：将p1修改为"/><script>alert(document.domain)</script><!



1. 题目简述：此题仍旧会将输入的内容变为input的value的值，但他将<>进行实体编码。

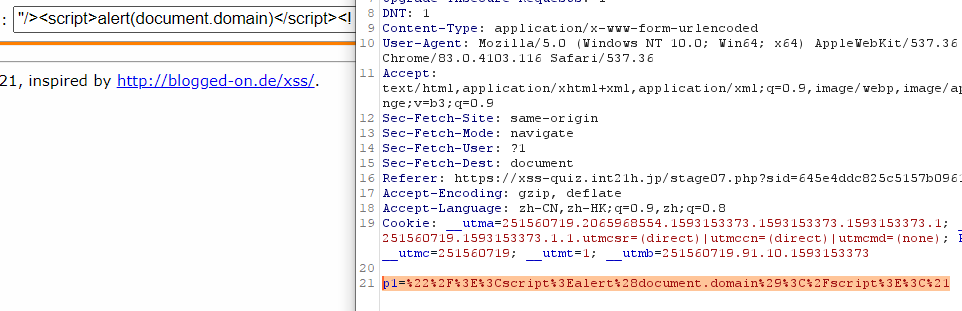


漏洞：未将引号进行实体编码，则可以闭合value属性，构建onclick属性，并运行代码。

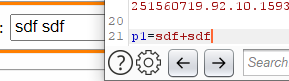
输入："onclick="alert(document.domain)"



1. 题目简述：此题会将输入的内容变为input的value的值，但他将<>和引号等非法符号都进行了转码。



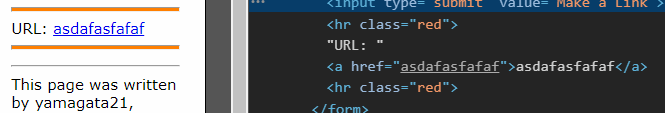
漏洞：未对空格进行检测，先随意输入一段字符串，然后加上空格，再加上我们构造的onclick属性，即可运行代码。



输入：asdas onclick=alert(document.domain)

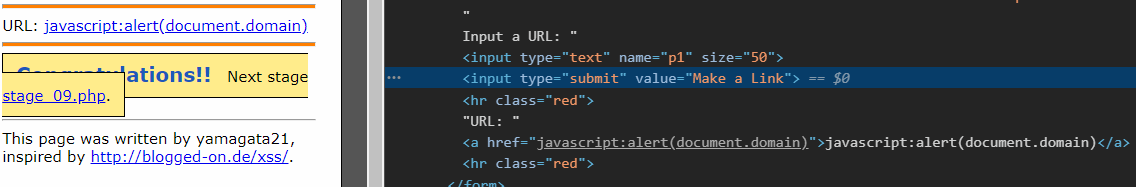


1. 题目简述：输入一段字符串，会将输入的内容放入<a>标签中作为一个链接。

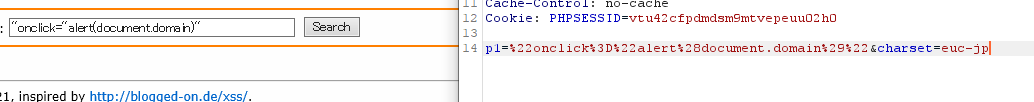


漏洞：对于输入的内容未进行检测，可以在url中加入代码，然后点击链接即可运行。

输入：javascript:alert(document.domain)

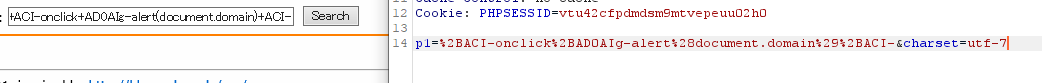


1. 题目简述：输入一段字符串，会将输入的内容放入input的value属性中，同样对输入的字符串的非法字符进行了转码，通过抓包看到其传递了两个参数。

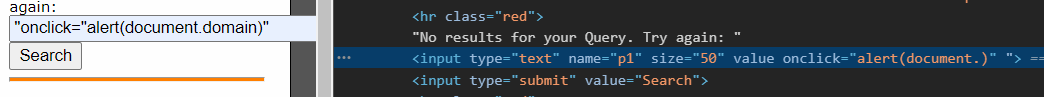


漏洞：可以通过使用utf-7编码进行绕过，则需要将要输入的语句先编码为UTF-7，然后在进行抓包，将charset参数改为UTF-7。但只能在ie浏览器中实现。

输入：+ACI-onclick+AD0AIg-alert(document.domain)+ACI-

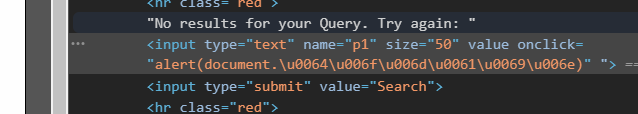


1. 题目简述：输入一段字符串，会将输入的内容放入input的value属性中，但将domain这个名词进行了过滤。



漏洞：可以使用Unicode编码表示domain

输入："onclick="alert(document.\\u0064\\u006f\\u006d\\u0061\\u0069\\u006e)"

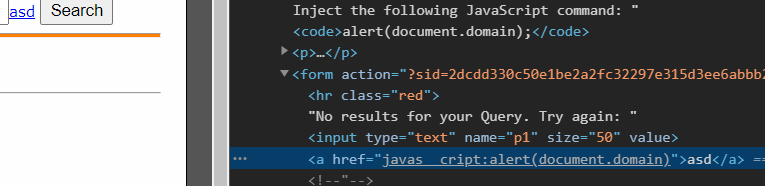


1. 题目表述：输入一段字符串，会将输入的内容放入input的value属性中，但对某些名词进行了检测，将onclick变为onxxx，将script变为xscript，将style变为stxxx。

漏洞：可以通过在名词间加入&Tab;来绕过检测，因为js语句是检测到分号才算是一个语句的结束，所以加入一个制表符，js会继续处理直到语句结束或遇到分号。

输入：

"/><a href="javas&Tab;cript:alert(document.domain)">asd</a><!



1. 题目表述：输入一段字符串，会将输入的内容放入input的value属性中，但其将引号、大于小于号等非法符号均进行过滤处理。

漏洞：未对`进行过滤，而在ie浏览器下，会将`作为引号处理，则可以使用`代替引号进行攻击。

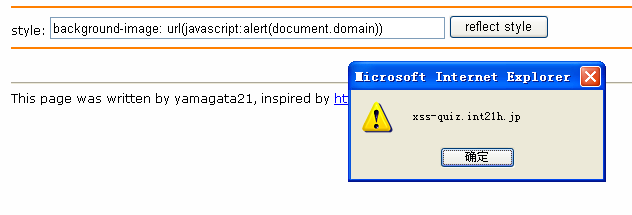
输入：`asdasd`onclick=alert(document.domain)



1. 题目简述：输入一段字符串，会将输入的内容放入input的style属性中。

漏洞：CSS中也可以嵌入js代码进行执行。在旧版的ie浏览器上才可以实现。

输入：background-image: url(javascript:alert(document.domain))

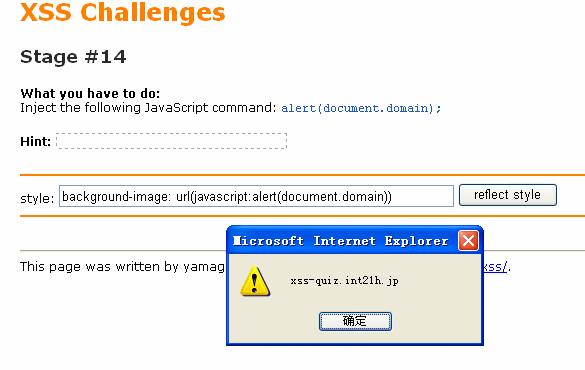


1. 题目简述：输入一段字符串，会将输入的内容放入input的style属性中，但其对url、expression、script、eval等名词进行了检测。

漏洞：可以在名词间插入\0，因为浏览器会忽略掉\0，从而绕过检测。

输入：

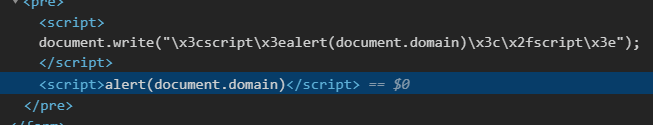
background-image: u\0rl(javascr\0ipt:alert(document.domain))



1. 题目简述：输入一段字符串，会将输入的内容通过document.write()这个函数打印出来，并显示在页面上。且对大于小于号引号等非法符号进行了检测。

漏洞：可以使用16进制表示相应的非法符号，从而绕过检测。

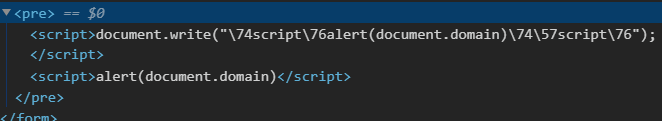
输入：[\\x3cscript\\x3ealert(document.domain)\\x3c\\x2fscript\\x3e](\\\\x3cscript\\\\x3ealert(document.domain)\\\\x3c\\\\x2fscript\\\\x3e)



1. 题目简述：输入一段字符串，会将输入的内容通过document.write()这个函数打印出来，并显示在页面上。且对大于小于号引号等非法符号进行了检测，同时还对\x进行了检测，即禁用了16进制。

漏洞：可以使用8进制来表示相应的非法符号，从而绕过检测。

输入：[\\74script\\76alert(document.domain)\\74\\57script\\76](\\\\74script\\\\76alert(document.domain)\\\\74\\\\57script\\\\76)



1. 题目简述：需要输入两个字符串，分别将输入的内容放在两个input的value中。将引号、大于小于号等非法符号实体编码。

漏洞：可以使用利用双字节字符，将第一个value的尾部的引号变为双字节字符的一部分，如此便会导致第一个value的左引号到第二个value的左引号的内容都成为第一个value的值。然后构造onclick属性，运行js代码，并用同样的方法去掉第二个value的尾部的引号。利用抓包工具将两个参数的值进行修改。该漏洞也只能在旧版ie浏览器上实现。



输入：

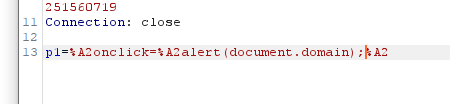
sdfsdf%A7

onclick=alert(document.domain); %A7

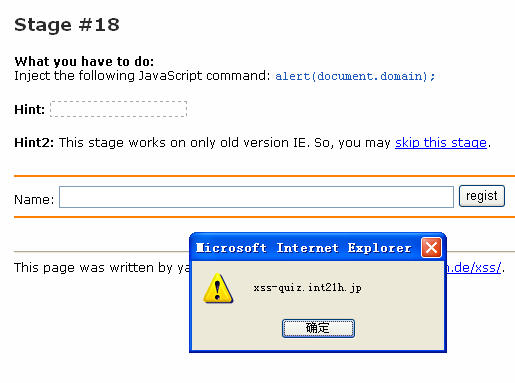


1. 题目简述：输入一段字符串，会将输入的内容放入input的value中，且大于小于号、引号等非法符号均被过滤。

漏洞：利用浏览器会将8bit的2进制当作7bit来辨识的缺陷，我们可以绕过符号过滤。即将<(0x3c)换成0xbc、>(0x3e)换成0xbe、”(0x22)换成0xA2等等绕过其过滤机制。并且要使用抓包工具，抓包进行修改。



输入：%A2onclick=%A2alert(document.domain);%A2



总结：

XSS攻击的基本方式：通过输入或抓包修改参数，将插入部分的前面的原代码闭合，再加入要注入的代码，然后注释掉后方的代码或不理会也可以。从而达到自己的目的。

XSS攻击绕过过滤规则的方式：

1. 使用编码绕过，可以使用unicode码、16进制、8进制码、utf-7等编码绕过其过滤机制。
2. 插入空格、制表符、\0等符号绕过，对script检测时可以在script字母之间插入以上符号，浏览器会自动略过，而过滤机制则不会略过这些符号。

XSS攻击的大致思路：构造<script>xss<script>、onclick=xss、style=background-image：url(xss)、<a href=xss><\a>。