ctf xss利用_TCTF/0CTF2018 XSS Writeup

原创

邹家二哥 于 2021-01-14 15:57:28 发布 118 收藏
文章标签： ctf xss利用

作者：LoRexxar'@知道创宇404实验室

刚刚4月过去的TCTF/0CTF2018一如既往的给了我们惊喜，其中最大的惊喜莫过于多道xss中Bypass CSP的题目，其中有很多应用于现代网站的防御思路。

其中bl0g提及了通过变量覆盖来调用已有代码动态插入Script标签绕过strict-dynamicCSP的利用方式。

h4xors.club2则是通过Script Gadgets和postmessage中间人来实现利用。

h4x0rs.space提及了Appcache以及Service worker配合jsonp接口实现的利用思路。

其中的很多利用思路非常精巧，值得研究。所以我花费了大量时间复现其中题目的思路以及环境，希望能给读者带来更多东西...

bl0g

题目分析

An extremely secure blog

Just focus on the static files. plz do not use any scanner, or your IP will be blocked.

很有趣的题目，整个题的难点在于利用上

站内的功能都是比较常见的xss功能

new 新生成文章

article/xx 查看文章/评论

submit 提交url (start with http://202.120.7.197:8090/)

flag admin可以查看到正确的flag

还有一些隐藏的条件

1、CSP

Content-Security-Policy:

script-src 'self' 'unsafe-inline'

Content-Security-Policy:

default-src 'none'; script-src 'nonce-hAovzHMfA+dpxVdTXRzpZq72Fjs=' 'strict-dynamic'; style-src 'self'; img-src 'self' data:; media-src 'self'; font-src 'self' data:; connect-src 'self'; base-uri 'none'

挺有趣的写法，经过我的测试，两个CSP分开写，是同时生效并且单独生效的，也就是与的关系。

换个说法就是，假设我们通过动态生成script标签的方式，成功绕过了第二个CSP，但我们引入了

从CSP我们也可以简单窥得一些利用思路，base-uri 'none'代表我们没办法通过修改根域来实现攻击，default-src 'none'这其中包含了frame-src，这代表攻击方式一定在站内实现，script-src的双限制代表我们只能通过

2、new中有一个字段是effect，是设置特效的

POST /new HTTP/1.1

Host: 202.120.7.197:8090

Connection: keep-alive

Content-Length: 35

Cache-Control: max-age=0

Origin: http://202.120.7.197:8090

Upgrade-Insecure-Requests: 1

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

Referer: http://202.120.7.197:8090/new

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9

Cookie: BL0G_SID=vV1p59LGb01C4ys4SIFNve4d_upQrCpyykkXWmj4g-i8u2QQzngP5LIW28L0oB1_NB3cJn0TCwjdE32iBt6h

title=a&content=a&effect=nest

effect字段会插入到页面中的，但这里实际上是没有任何过滤的，也就是说我们可以通过闭合这个标签并插入我们想要的标签，需要注意的是，这个点只能插入70个字符。

3、login?next=这个点可以存在一个任意跳转，通过这个点，我们可以绕过submit的限制(submit的maxlength是前台限制，可以随便跳转

4、站内的特效是通过jqery的append引入的，在article.js这个文件中。

$(document).ready(function(){

$("body").append((effects[$("#effect").val()]));

});

effects在config.js中被定义。

回顾上面的几个条件，我们可以简单的整理思路。

在不考虑0day的情况下，我们唯有通过想办法通过动态生成script标签，通过sd CSP这个点来绕过

首先我们观察xss点周围的html结构

在整站不开启任何缓存的情况下，通过插入标签的方式，唯一存在一种绕过方式就是插入

浏览器有一定的容错能力，他会补足不完整的标签

=====>

但这个操作在这里并不适用，因为中间过多无用标签，再加上即使吞了也不能有什么办法控制后面的内容，所以这里只有一种绕过方式就是dom xss。

稍微翻翻可以发现，唯一的机会就在这里

$(document).ready(function(){

$("body").append((effects[$("#effect").val()]));

});

如果我们可以覆盖effects变量，那我们就可以向body注入标签了，这里需要一点小trick。

在js中，对于特定的form,iframe,applet,embed,object,img标签，我们可以通过设置id或者name来使得通过id或name获取标签

也就是说，我们可以通过effects获取到

这个标签。同理，我们就可以通过插入这个标签来注册effects这个变量。
可如果我们尝试插入这个标签后，我们发现插入的effects在接下来的config.js中被覆盖了。

这时候我们回到刚才提到的特性，浏览器有一定的容错能力，我们可以通过插入，那么中间的代码就会被视为js代码，被CSP拦截。

我们成功的覆盖了effects变量，紧接着我们需要覆盖effects[$("#effect").val()]，这里我们选择id属性(这里其实是为了id会使用两次，可以更省位数)，

所以我们尝试传入

effect=id">