HCTF2018 部分 web 题目 Writeup

原创

知道创宇KCSC 于 2019-10-10 13:43:45 发布 1177 收藏
文章标签： 网络安全 漏洞

本文链接：https://blog.csdn.net/qq_43380549/article/details/102397174

作者：LoRexxar'@知道创宇404实验室
时间：2018年11月14日
如果你想第一时间了解漏洞资讯，可以关注我们的知道创宇Paper：https://paper.seebug.org/744/

HCTF2018在出题的时候其实准备了一个特别好的web题目思路，可惜赛前智能合约花了太多时间和精力，没办法只能放弃了之前的web题，在运维比赛的过程中，我发现学弟出的一些题目其实很有意思值得思考。

bottle

bottle是小学弟@luo00出的题目，源码如下 https://github.com/Lou00/HCTF2018_Bottle

整个站几乎只有一个功能就是有一个可控的任意跳转，然后根据题目功能可以判断是一道xss题目。其实技巧挺明确的，就是比较冷门，我第一次见是阿里先知的xss挑战赛。

https://lorexxar.cn/2017/08/31/xss-ali/#05%E8%B7%B3%E8%BD%AC

然后本题的思路主要来自于ph师傅的一篇分析 https://www.leavesongs.com/PENETRATION/bottle-crlf-cve-2016-9964.html

首先这个问题有意思的点在于挺多的，在原本的环境下，bottle有个特殊的鬼畜特性在于，他的header顺序是会变得…

首先我们需要明白一个问题，在流量中，body和header是在一起的，在header的两个换行后内容会被自动识别为body。

所以在`bottle.redirect(path)`中存在location头注入，我们就可以通过传入两个换行来吧header挤到body中，这样就可以控制页面的返回了

```
150.109.53.69:/path?path=//150.109.53.69:0%2f%0D%0A%0D%0Atest
```

正常来说，直接注入script就可以了

```
http://150.109.53.69:3000/path?path=http://150.109.53.69:0%2f%0D%0A%0D%0A<html><head><script>alert`1`</script>
```

原文中说当端口小于80，firefox就会卡住，但我实际测试只有0端口会卡住，可能我环境不同。

这就是题目的原解，这里虽然加入了CSP，但其实没区别，由于bottle头随机的问题，当CSP随机到location下面时，就可以注入js了，但这样就成了一个随机的题目了，学弟想让别人注意到bottle特性而不是随便撞到，这里就设置了脚本定时重启，然后让头更随机一点儿。

攻击者需要意识到这个问题然后不断提交才可以攻击成功，但可惜这种攻击方式就随机了，失去了ctf本身的乐趣，变得太无趣了。

----------下面开始脑洞时间，实际没有作用，不想看可以跳过----------

尝试

仔细思考了一下逻辑我开始想办法改进这题。当然，改进题目的基础必然是想办法减少随机性，所以一些讨论的基础都在于CSP头稳定在location之上。

其实可以发现，CSP特别简单，最简单的self CSP

```
response.add_header('Content-Security-Policy',"default-src 'self'; script-src 'self'")
```

self CSP最大的问题在于如果能找到一个self源内容可控的，那CSP就可以被绕过了。

然后我发现，这个漏洞不是刚好就是可以控制页面内容吗，于是一个漏洞利用链想到了

构造一个CLRF控制内容注入alert，然后构造CLRF，然后构造第二个CLRF注入script，然后src引入前面的链接。

听起来非常完美的利用链。这其中也有几个小坑。

首先构造一个alert

```
http://150.109.53.69:3000/path?path=http://150.109.53.69:0%2f%0D%0A%0D%0Aalert%60%31%60%3b
```

这里想到现代浏览器对content-type可能有要求，所以直接头注入设置content-type为text/javascript

```
http://150.109.53.69:3000/path?path=http://150.109.53.69:0%2f%0D%0AContent-Type%3a+text/javascript%3b+charset%3dUTF-8%0D%0A%0D%0Aalert`1`
```

然后尝试引入这个链接，然后需要注意二次urlencode，不然%0a%0d都会解开

```
http://150.109.53.69:3000/path?path=http://150.109.53.69:0%2f%0D%0A%0D%0A<html><head><script/src=%68%74%74%70%3a%2f%2f%31%35%30%2e%31%30%39%2e%35%33%2e%36%39%3a%33%30%30%30%2f%70%61%74%68%3f%70%61%74%68%3d%68%74%74%70%3a%2f%2f%31%35%30%2e%31%30%39%2e%35%33%2e%36%39%3a%30%25%32%66%25%30%44%25%30%41%43%6f%6e%74%65%6e%74%2d%54%79%70%65%25%33%61%2b%74%65%78%74%2f%6a%61%76%61%73%63%72%69%70%74%25%33%62%2b%63%68%61%72%73%65%74%25%33%64%55%54%46%2d%38%25%30%44%25%30%41%25%30%44%25%30%41%61%6c%65%72%74%60%31%60></script>
```

看上去很有道理，然后访问…然后失败…Orz，被CSP ban了

仔细回想上面的流程，其实有个很重要的问题没有注意到，这个问题我也是第一次重视到。

CSP和cookie的同源策略一样，不但对ip做限制，对端口也有限制。最过分的是，CSP在location头存在的时候，会跟入判断location

也就意味着，我们试图引入`http://150.109.53.69:3000/path?path=http://150.109.53.69:0%2f%0D%0AContent-Type%3a+text/javascript%3b+charset%3dUTF-8%0D%0A%0D%0Aalert`作为目标js引入，会被认为引入http://150.109.53.69:0这个来源的js，端口为0，self的端口为3000，所以被拦截了。

而且值得注意的是，这里如果把CSP改为

```
response.add_header('Content-Security-Policy',"default-src 'self'; script-src 150.109.53.69")
```

CSP中如果不设置端口，默认会认为是80端口。同样没办法绕过。

经过了一番研究我发现这个端口判定没有别的解决办法，如果不用跳0的办法，正常的302是会跟随跳转过去的。

无奈我修改题目把CSP改成了

```
response.add_header('Content-Security-Policy',"default-src 'self'; script-src 150.109.53.69:*")
```

然后我开始继续上面的测试，果然，仍然失败了…这次报错不一样了，阻拦加载的并不是CSP。

firefox的控制台显示script加载失败，仔细研究了一番突然意识到一个事情，是不是浏览器的不加载非200的静态资源。

于是我用flask简单写了一段代码测试了一下

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'alert(1);', 303

if __name__ == '__main__':
    app.run()
```

事实证明的确是这样的，浏览器在这块的安全性已经做的非常好了，这种奇怪的操作完全不成立...

思考到最后，我忽然又意识到了一个问题，假设我把跳转那个页面生生改成200，然后去加载，有一个致命的利用问题。

模拟出来的页面内容是这样的

```
alert`1`;
xxxxxheader: xxxx
```

我没办法改变下面的内容，而js虽然是逐行渲染的，但遇到报错之后整块script都会阻止，不再继续加载了，然后我就又回到了最初的问题，我必须保证locaion是最后一行header才行...我违背了最初想要去除随机性的目的...

最后没办法，还是将题目改回原样了，这里的思考过程挺有意思的，分享给大家，也感谢在试验过程中@Math1as给我出了很多主意。

game

game这道题是我的另一个小学弟@undifined出的题目，后来听他说起这个思路我觉得蛮有意思的，这里出成题目用了明文密码入库虽说是比较强行，但其实用来注其他信息还是不难的，是个很有趣的想法。

第一次见到这种思路是在pwnhub上

https://pwnhub.cn/questiondetail?id=3

这题目当时是上传文件，然后后端展示的时候会有排序，通过不断上传就可以得到目标文件名字。

这里也是一样，这里是一个近似于逻辑漏洞，站内只有两个功能:

1. 打游戏获得积分
2. 排行榜，可以根据不同的字段排行

排行榜的数据都是实时的，而且整个部分没有任何的注入点，完全不能SQL注入，但由于可以根据不同的字段排行，所以order by后面的字段名可控，除了常规的id, username, sex, score以外，也可以更具password来排行，再加上数据库中密码是明文存取的（不是明文也可以，只是获得的是hash）

知道了原理，我们就可以通过不断插入新的账号来逼近目标字符串，因为数据库排序和前面说的linux文件名排序是一样的，很有趣。

```
ac > adf > ad
```

想明白就可以直接写脚本跑起来

欢迎关注我和专栏，我将定期搬运技术文章～

也欢迎访问我们：知道创宇云安全 ：https://www.yunaq.com/?from=CSDN911010
或拨打热线电话：400-833-1123

如果你想与我成为朋友，欢迎加微信kcsc818~