WEB安全Permeate漏洞靶场挖掘实践

原创

汤青松 于 2018-08-15 00:00:00 发布

简介

最近在逛码云时候发现permeat靶场系统,感觉界面和业务场景设计的还不错.所以过来分享一下.

同时也是分享一下我平时挖掘漏洞的一些思路吧,这篇文章里虽然只简单介绍其中三种漏洞类型，但也是想是一个抛转引玉吧,给web安全新手提供一些挖掘思路.

下载地址:

```
GitHub地址: https://github.com/78778443/permeate
国内地址: https://gitee.com/songboy/permeate
```

这篇文章里主要介绍其中的,SQL注入挖掘,xss跨站挖掘,以及csrf漏洞把

在挖掘一网站的漏洞时候,我们脑海里要知道什么漏洞在什么场景下容易出现,那些漏洞出现的比较频繁,我脑海里的web安全漏洞有三种类型吧:

1. 编码型漏洞
2. 业务逻辑漏洞
3. 运行环境漏洞

笔者之前给别人做代码审计有一个习惯,通常希望给的源码能够正常运行,还不是光从代码来分析问题.
为什么这么做呢,因为觉得如果代码不能运行其实很多漏洞是无法光从代码层面发现问题的.
而代码能运行起来,其实不仅能验证问题,也可以从系统的业务功能来找出更多问题.

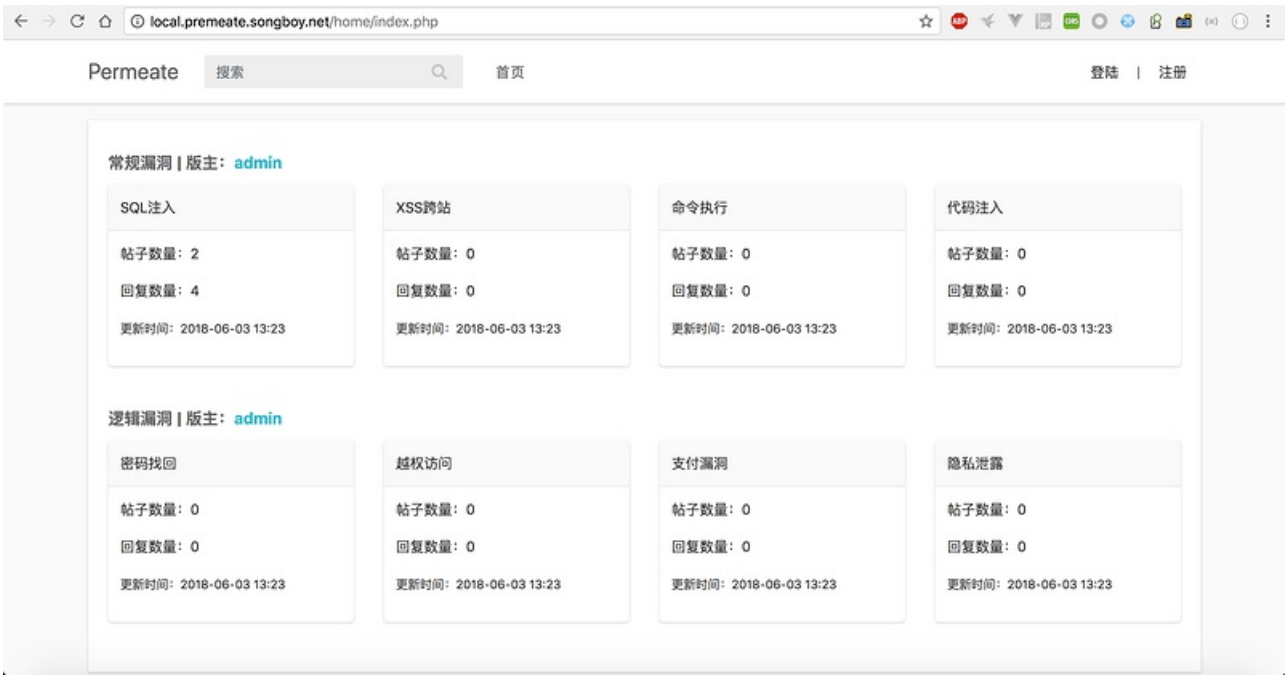
比如说很多网站提供站内搜索功能,在搜索的时候通常会把用户搜索的关键词返回在页面当中,比如"你搜搜的关键词'关键词'结果如下",那在这里就很有可能存在反射型XSS漏洞.

另外很多网站都存在用户体系,而在修改个人资料的时候很有可能存在越权问题,比如修改的个人资料的时候查看是否又提交uid参数,如果有,修改uid值,看是不是把别人的资料给修改了,这些其实都需要运行之后才能发现问题所在.

笔者在windows系统中是由wampser搭建还比较简单,这里先简单介绍安装方法:

1. 用git把代码拉下来
2. 配置单独的虚拟主机目录(不要放到二级目录,貌似有问题)
3. 新建了一个数据库
4. 导入sql文件,文件位置 /doc/bbs_cate.sql
5. 修改一下/config/dbconfig.php文件中的数据库账号密码信息

通过上面的安装步骤之后,应该可以看到如下面的界面了.有一个默认板块和一个默认分区,就说明连接数据库成功了.



现在我们开始去挖掘里面的漏洞,在项目介绍中看到有SQL注入和XSS以及CSRF问题,但是没有告知存在漏洞的位置,所以我们需要先分析每个漏洞的对应场景.

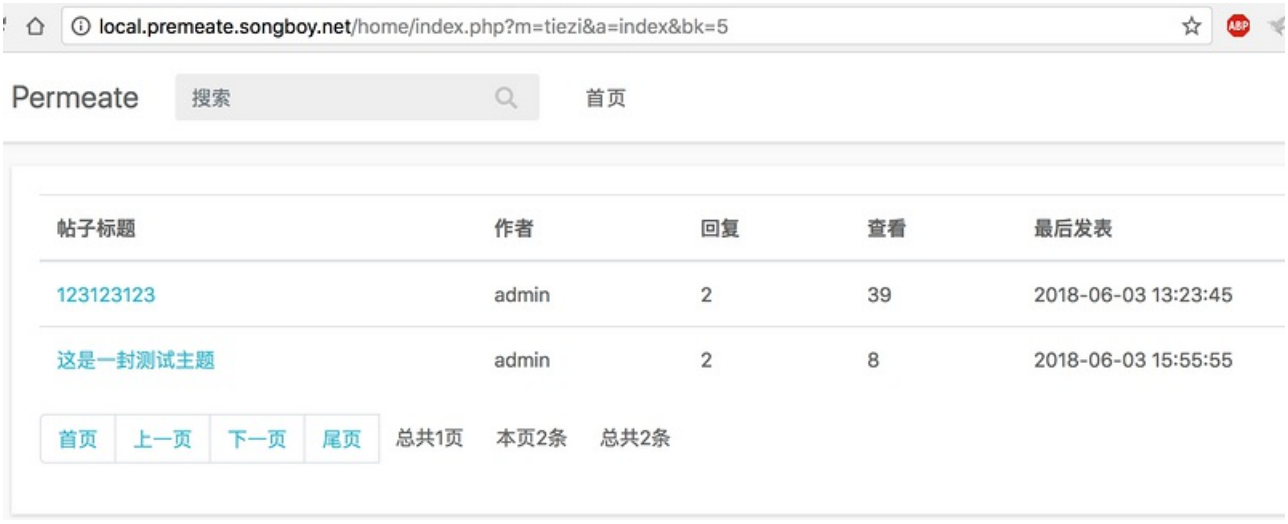
先来说说SQL注入挖掘吧.

一. SQL注入挖掘

懂点SQL注入知识应该都可以想到sql注入是因为攻击者可以控制sql语句中的参数所造成的,那么我们就先找一个需要传参的地址,在刚才的首页中可以看到有一个默认板块,那么就点击默认板块好了,点击之后上面的URL地址变成了

```
http://permeate.localhost/home/index.php?m=tiezi&a=index&bk=5
```

在URL中可以看到,有三个参数,但根据经验来说,前面两个参数m和a有点像是路由,所以这两个先暂时用排除法排除,最后一个参数bk是一个数字,感觉应该是板块的ID,所以可以重点关注一下,我们先记住未测试之前的页面是什么样子



现在先用手动测试快速测试一下,怎么测试呢? 可以在bk=5后面加一个单引号,或者加一个%27,得到URL地址如下

```
http://permeate.localhost/home/index.php?m=tiezi&a=index&bk=5'
```

这个时候看一下页面的运行效果如何,发现帖子列表中帖子已经不存在了.

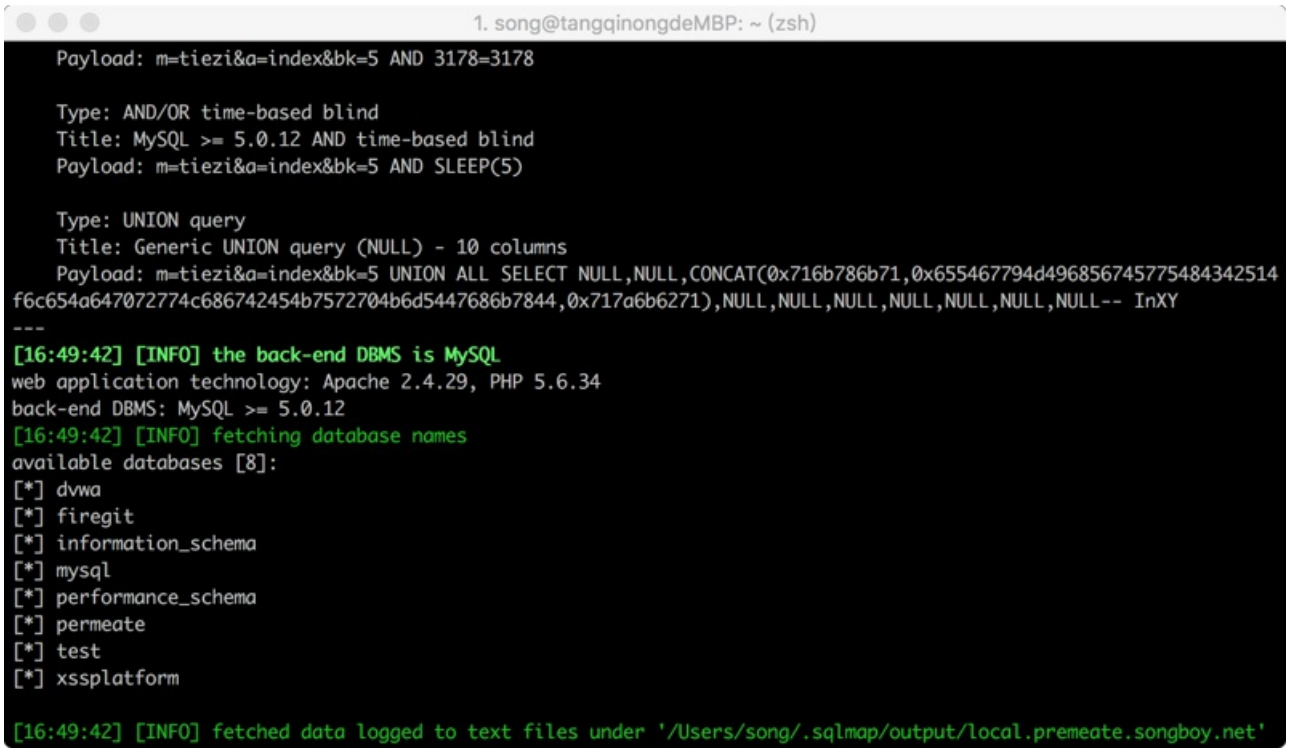
这个时候我们可以初步的得出结论,这个地方可能存在SQL注入问题,但是还不能肯定,要肯定这个地方是否存在注入问题,我们这样深入去验证一下,可以使用参数值 5' or '1'='1 来进行验证,得到URL地址如下

```
http://permeate.localhost/home/index.php?m=tiezi&a=index&bk=5' or '1'='1
```

访问之后,发现页面又发生了一次变化,这个时候我们就可以肯定这个地方存在了SQL注入问题,如下图

我们可以使用sqlmap来看看数据库存在哪些数据库信息,sqlmap命令如下:

```
sqlmap -u "http://permeate.localhost/home/index.php?m=tiezi&a=index&bk=5"  --dbs
```



通过sqlmap的反馈结果可以看出,这个地方确实存在了注入问题.

下面我们接着找一下XSS漏洞漏洞

二. XSS跨站

造成xss的主要成因我们知道是参数会被在页面输出,所以在找XSS漏洞的时候,我们先看看站点有什么功能;

在首页的图片和帖子列表页中可以大致看出有搜索功能,和发帖,回复帖子等功能,这些地方都会把接收的参数作为内容展示出来,所以我们可以挨个去测试.

先来一个最简单的搜索页吧,在导航栏有一个搜索框,我首先在搜索框中输入test吧,得到URL地址如下

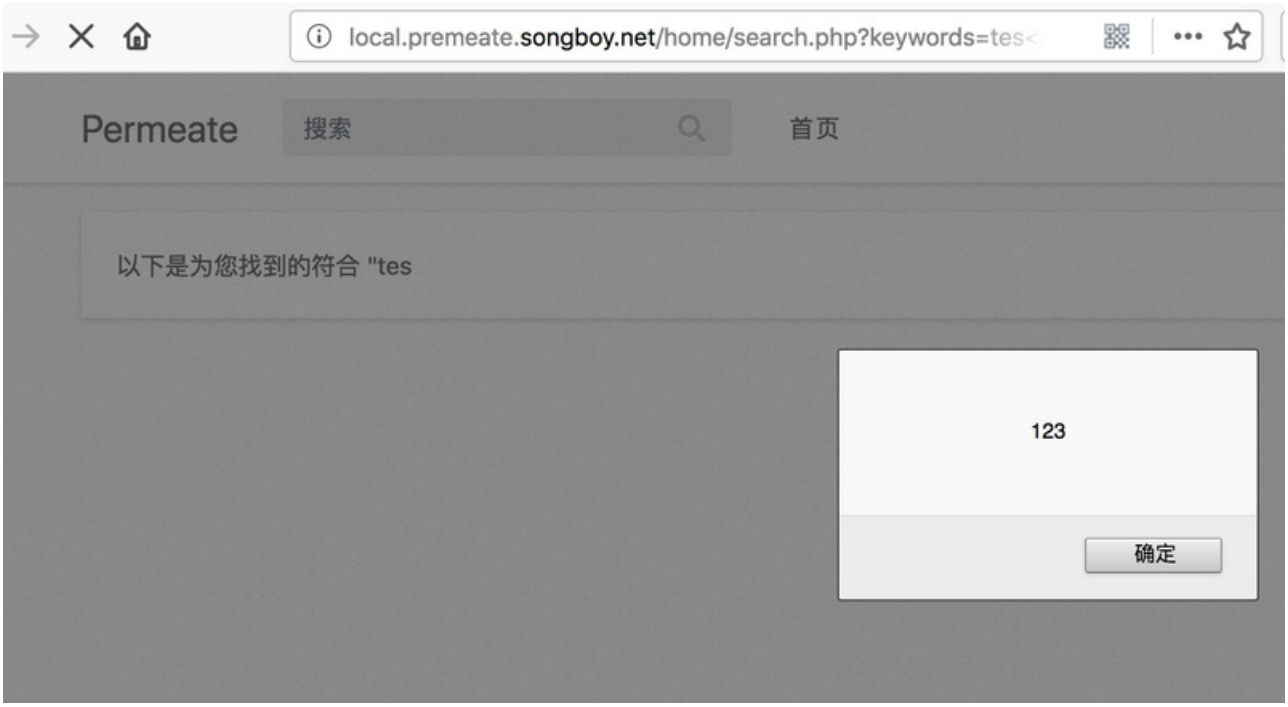
```
http://permeate.localhost/home/search.php?keywords=test
```

从URL地址可以看出搜索的关键词会通过keywords来传递,传递之后也会显示在页面内容当中,如下图

现在分析显示html元素,在关键词test的父级节点,可以看到是div,div中写入script标签是会被执行的,所以可以直接使用下面的payload

```
http://permeate.localhost/home/search.php?keywords=test<script>alert(123)</script>
```

进行测试,通过浏览器访问此连接,可以看到已经弹出123确认框



不过xss不仅限于script标签可以被执行,也可以用img标签的onerror属性来执行,可以构造如下的payload

```
http://permeate.localhost/home/search.php?keywords=test<img src=x onerror=alert(456)>
```

依然访问URL地址,可以看到456的确认框被弹出来了

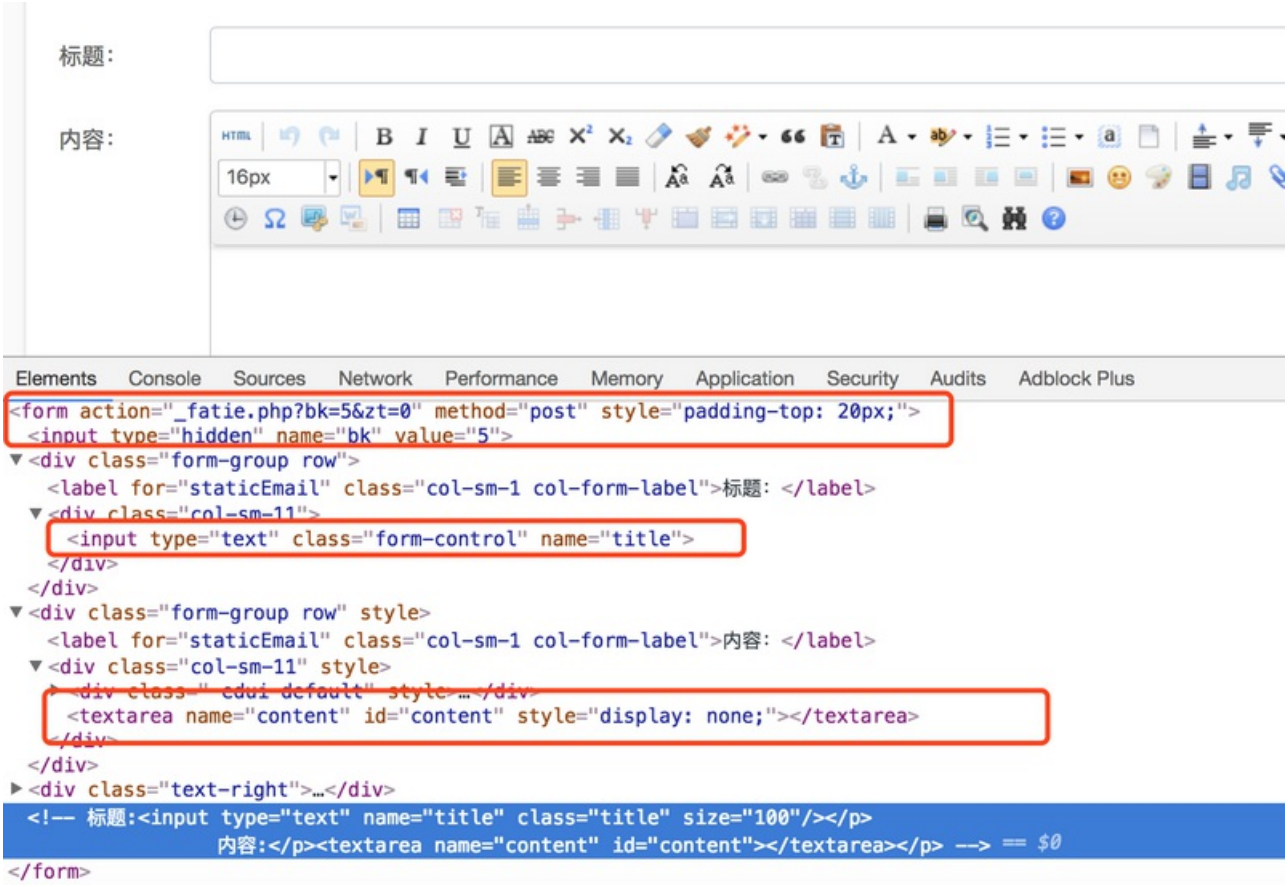
上面的XSS都属于反射型的,存储型的项目也表明存在,这里我们先略过吧,先看看CSRF的漏洞好了.

三. CSRF

CSRF漏洞主要成因是因为服务端接收表单请求时候没有验证是用户发送的请求还是浏览器发送的请求,所以在挖掘此类表单的时候先去找表单的位置,在前面的截图当中,可以看到有一个发帖的按钮,可以进去点进去看看,点击发帖,URL地址为

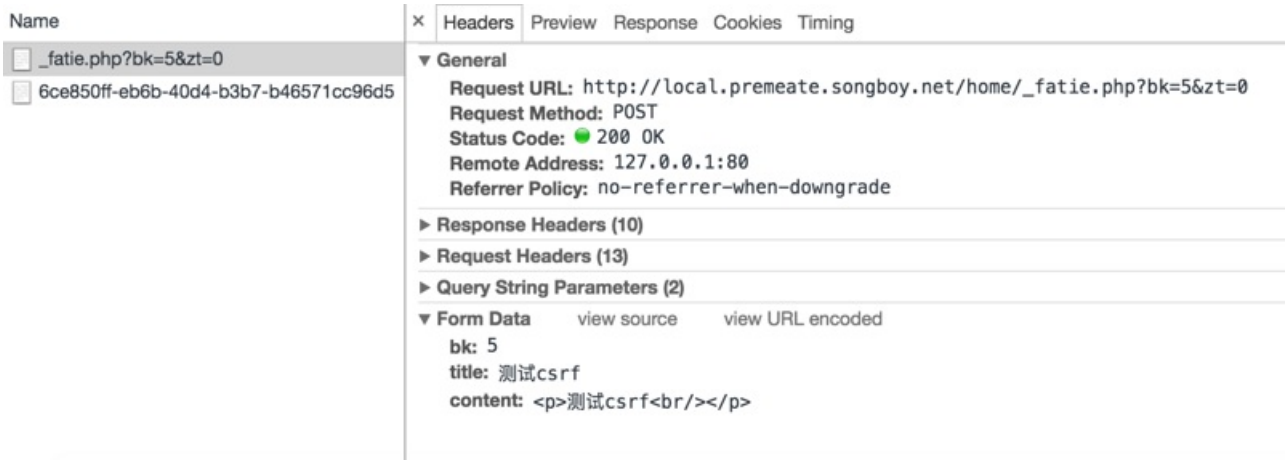
```
http://permeate.localhost/home/fatie.php?bk=5
```

在这个页面中可以看到有一个发帖的表单,我们主要看一下表单有没有token令牌,如果没有的话,那就可能存在CSRF的漏洞问题,通过浏览器的审查元素截图如下



在页面中确实没有存在token信息,因此我们可以初步得出结论,这个地方存在CSRF的可能,现在需要验证一下,

在验证的时候我们需要制定CSRF有GET型和POST型,get型利用起来相对简单很多,而在这个地方表单提交虽然是通过POST,但并不排除GET提交也可以利用,所以我们先尝试用GET型来提交数据,先通过抓包浏览器的网络分析模块来看,发帖会发送哪一些数据,如下图

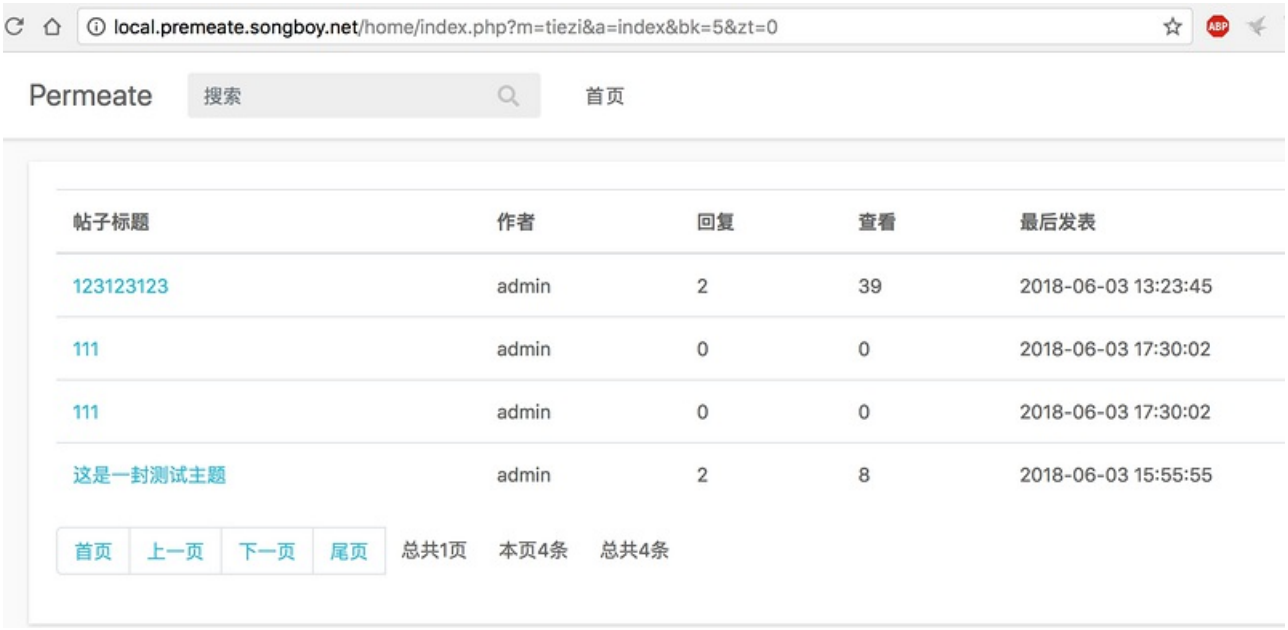


在图中可以看到,post会传递三个参数过去,bk和title以及content三个参数,我们先用get表单构造出一个payload出来,得到URL地址如下:

```
http://permeate.localhost/home/_fatie.php?bk=5&zt=0&title=111&content=222
```

然后去浏览器打开这个地址,看看是否能提交表单成功,访问后发现弹出了一个确认框,告诉我已经发帖成功了,如下图

去列表确认一下,发现确实已经发帖成功,如下图帖子列表



这个CSRF相对来说比较低级,很多情况下用get并不能提到post提交,因此我们再来尝试用post方法构造一个payload出来,代码如下

```
<html>
<head>test</head>
<body>
    <form action="http://permeate.localhost/home/_fatie.php?bk=5&zt=0" id="test" method="POST">
        <input type="hidden" value="11111" name="title"><br>
        <input type="hidden" value="22222" name="content">
    </form>
</body>
<script>
    var f=document.getElementById("test");
    f.getElementsByTagName("input")[0].value="title";
    f.getElementsByTagName("input")[1].value="content";
    f.submit();
</script>
</html>
```

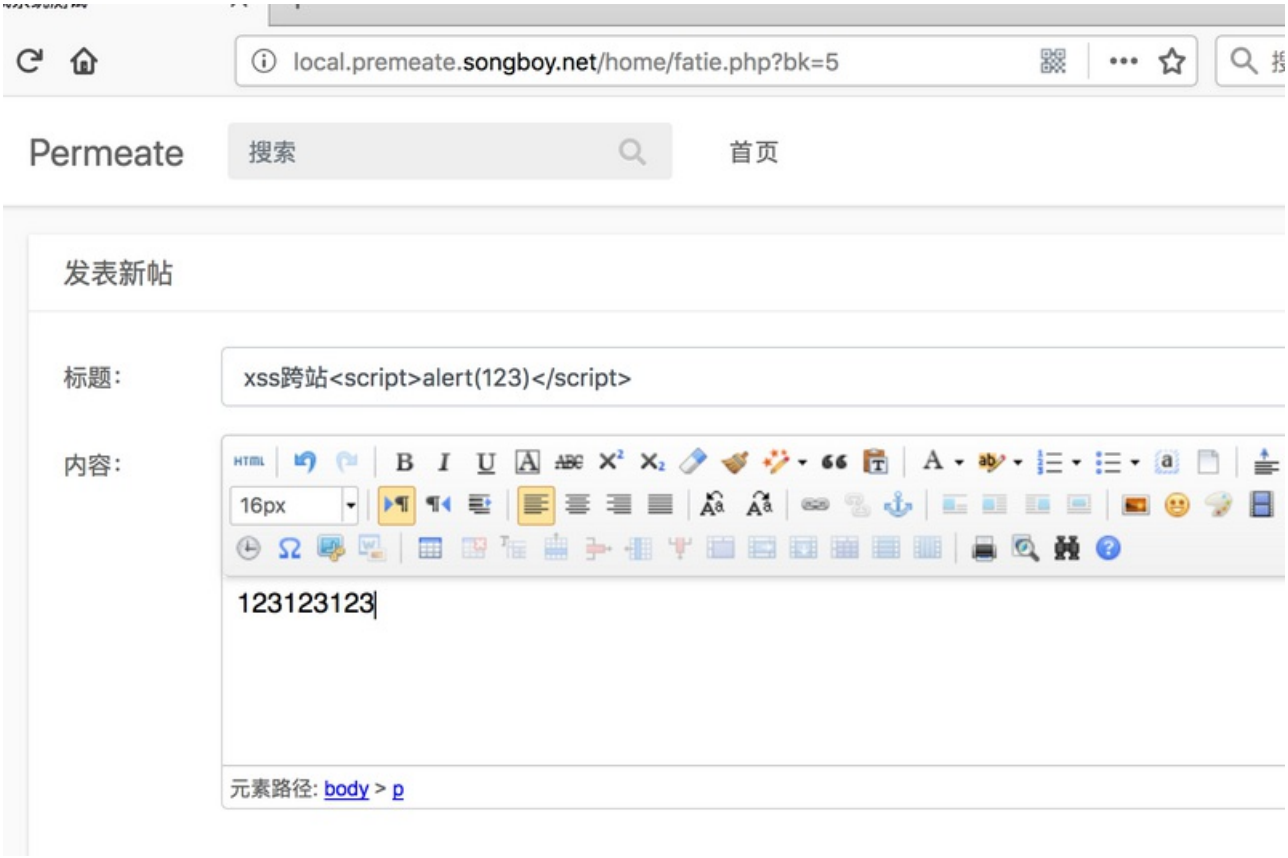
在上面代码中可以看出,表单里面的值已经事先做好了定义,当受害者打开之后,变回自动提交表单.

前面提到了这个系统中存在存储型XSS,又存在CSRF漏洞,那么我们是不是可以用来做一个XSS蠕虫试验呢?

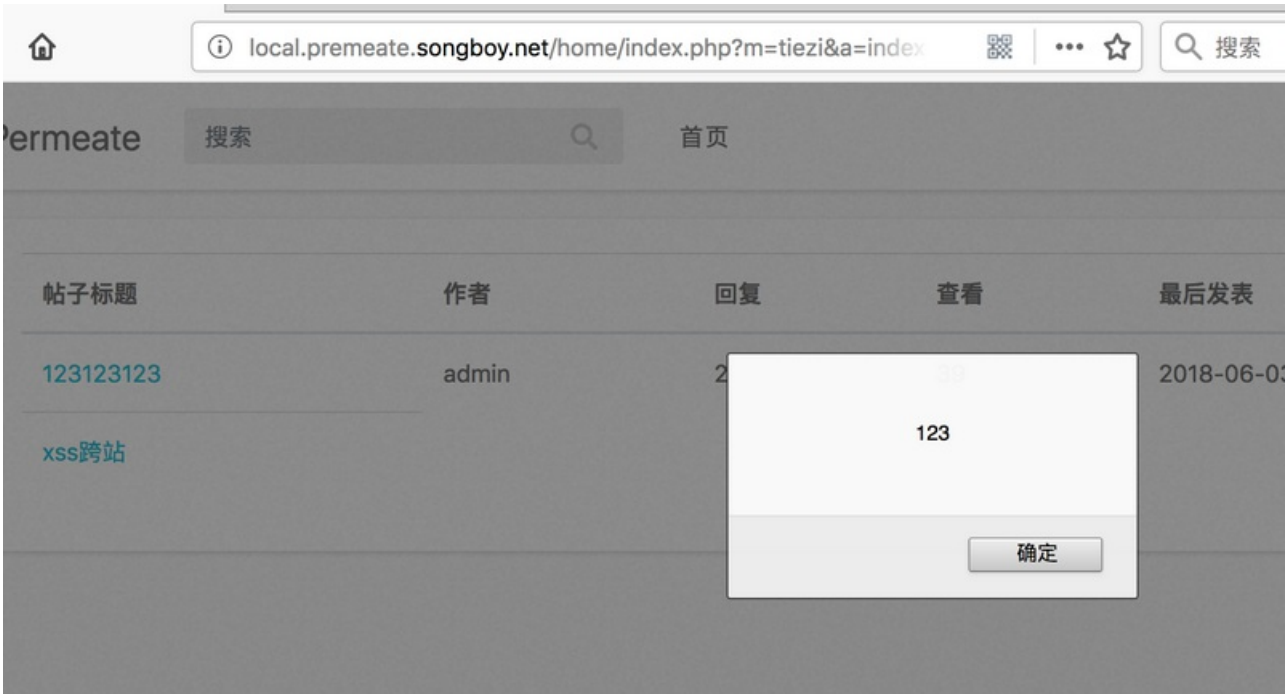
四. 蠕虫XSS

满足蠕虫XSS需要两个必要条件,存储型XSS,和CSRF漏洞,这两个这个系统都有,所以实现起来没有问题,需要的是先找一下存储型XSS

在前面发帖的位置会把参数存储起来,并展示,因此这个地方是一个比较好的试验点,我们在发帖位置先来试验一下吧. 如下图,表单中提交了xss验证代码,当成功触发的时候会被弹框123,如下图



点击发布帖子按钮,发布帖子成功,下来来到帖子列表页面,发现alert(123)已经被触发成功了,如下图



现在CSRF有了,存储型XSS也有了,那么接下来就是需要把它们结合起来使用了,但蠕虫XSS利用代码相对alert(123)来说代码会比较长,所以我们需要先把利用代码放到一个文件当中,然后再通过script引入进去,我们构造的代码如下

```
var strrand = +new Date();
var str = 'http://premeate.localhost/home/_fatie.php?bk=5&zt=0&title=1111<script src%3D/a.js%3F'+strrand+'>

var tag = document.createElement('img');
tag.src = str;
document.body.append(tag.src);
```

在上面的代码中,只要一被执行,页面将会被插入一个img标签,其中标签的src属性又会去请求表单,表单里面的内容又是一段xss代码,这样变回造成XSS蠕虫攻击者,每次请求都会是倍数增长.如下图,浏览器刷新3次之后,已经有很多帖子了.

作者：汤青松
微信：songboy8888