Catfish(鲶鱼) Blog V1.3.15存储型 xss

Catfish(鲶鱼) Blog前台文章评论处存在 存储型xss漏洞。

在 application\index\controller\index.php 文件第 692行

```
public function pinglun()
    {
        $beipinglunren = Db::name('posts')->where('id',Request::instance()-
>post('id'))->field('post_author')->find();
        $comment = Db::name('options')->where('option_name','comment')-
>field('option_value')->find();
        $plzt = 1;
        if($comment['option_value'] == 1)
        {
            $plzt = 0;
        }
        $uid = 0;
        if(Session::has($this->session_prefix.'user_id'))
        {
            $uid = Session::get($this->session_prefix.'user_id');
        }
        $data = [
            'post_id' => Request::instance()->post('id'),
            'url' => 'index/Index/article/id/'.Request::instance()->post('id'),
            'uid' => $uid,
            'to_uid' => $beipinglunren['post_author'],
            'createtime' => date("Y-m-d H:i:s"),
            'content' => $this->filterJs(Request::instance()->post('pinglun')),
            'status' => $plzt
        ];
        Db::name('comments')->insert($data);
        Db::name('posts')
            ->where('id', Request::instance()->post('id'))
            ->update([
                'post_comment' => date("Y-m-d H:i:s"),
                'comment_count' => ['exp','comment_count+1']
            ]);
        $param = '';
        Hook::add('comment_post',$this->plugins);
        Hook::listen('comment_post',$param,$this->ccc);
    }
```

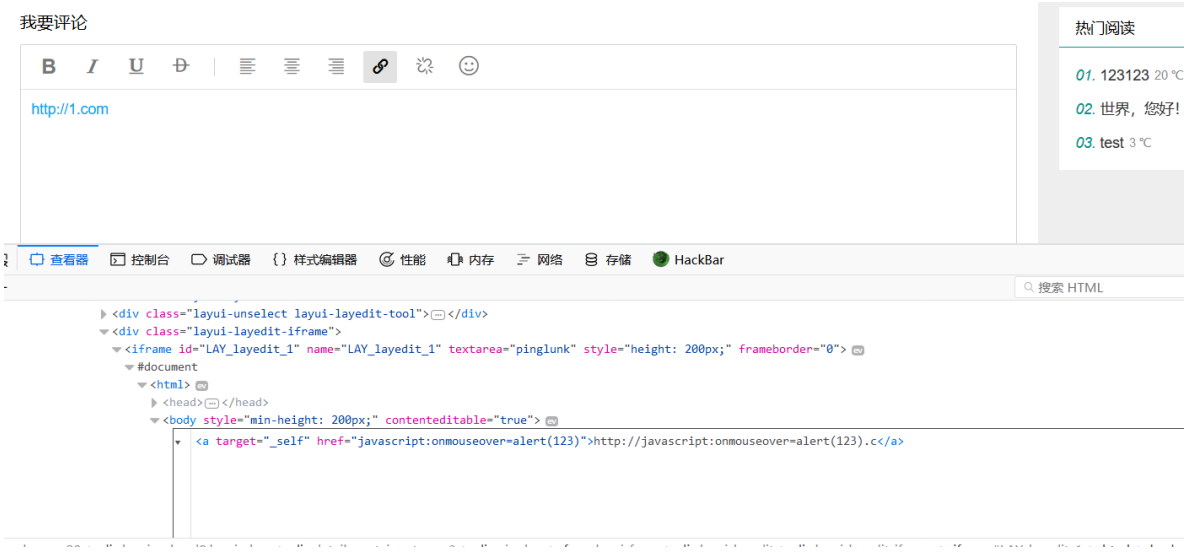
参数`content`经过了`filterJs`过滤，继续跟进`filterJs`

在文件 application\index\controller\Common.php 第831行

```
protected function filterJs($str)
    {
        while(stripos($str,'<script') !== false || stripos($str,'<style') !==
false || stripos($str,'<iframe') !== false || stripos($str,'<frame') !== false
|| stripos($str,'onclick') !== false)
        {
            $str = preg_replace(['/<script[\s\S]*?
<\/script[\s]*>/i','/<style[\s\S]*?<\/style[\s]*>/i','/<iframe[\s\S]*?
[<\/iframe|\/][\s]*>/i','/<frame[\s\S]*?[<\/frame|\/][\s]*>/i','/on[A-Za-z]+
[\s]*=[\s]*[\'|"][\s\S]*?[\'|"]/i'],'',$str);
        }
        return $str;
    }
```

看一下这里的逻辑，判断字符串是否存在 `<script`，`<style`，`<iframe`，`<frame`，`onclick` 如果存在就进行替换。那么我们提交的payload 只要不存在 上面判断的几个字符串就可以绕过。

漏洞复现： 前台页面找一篇文章，进行评论。评论加入超链接，这里因为有前端过滤所以不能直接输入payload，随便填写，然后F12修改超链接地址为 xsspayload `<a target="_self" href="javascript:onmouseover=alert(123)">http://javascript:onmouseover=alert(123).c</a>` 这样就绕过了前端认证，提交评论。



然后访问文章，点击评论触发xss漏洞。