登录

session，cookie认证会话中的安全问题

小猪佩琪 / 2019-03-08 09:01:00 / 浏览数 1748 安全技术 WEB安全 顶(1) 踩(0)

前言

在互联网中，对于用户来说不仅仅是单向信息传输，更重要的是信息交互，但是在web中，http协议是无状态的，当一个请求等到了服务器的回应之后，两者连接便中断了，

为什么需要会话

首先要理解tcp的三次握手，一般在TCP三次握手就创建了一个会话，在会话中可以传递信息，等TCP关闭连接就关闭会话了。HTTP协议是基于TCP协议的，所以http会话也

简单点，会话就是让你在信息交互的时候让对方知道你是谁，并且一直知道你是谁，但是服务器不是人，没有自我辨别的功能，这时候需要一种凭证，也就是session。站在

```
GET /login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
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
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/
Cookie: Hm_lvt_9d483e9e48ba1faa0dfceaf6333de846=1551249391,1551253956; PHPSESSID=akc7o4l74kkt2vq7orv9hn1k14; Hm_lpvt_9d483e9e48ba1faa0dfceaf6333de846=1551333272
Connection: close
Cache-Control: max-age=0
```

php中的session

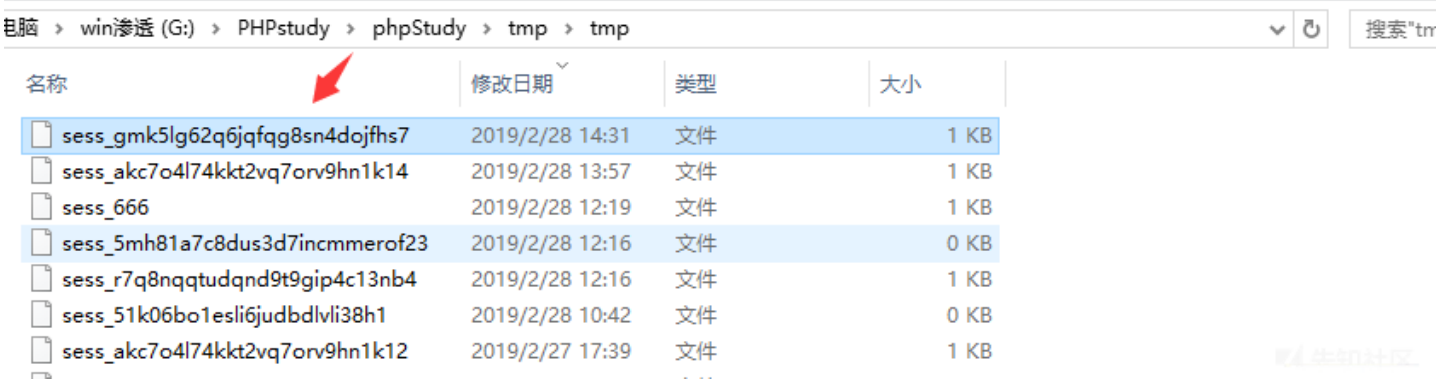
在上面我们提到了php中的session，那么一旦phpsessid泄露，相当于别人也可以使用你的身份，而且一般session在浏览器关闭后也不会注销，仅仅是你已经丢失了你的se
但是在php中并不是在登录之后才会给PHPSESSID，而是在

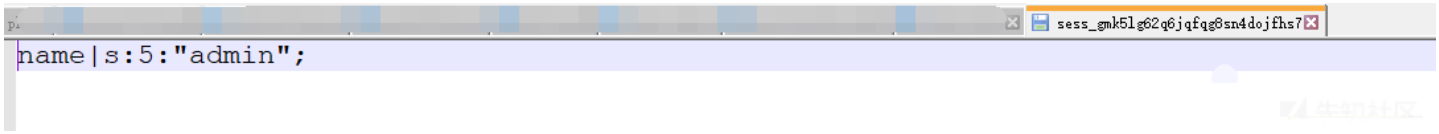
```
session_start();
```

开始之后，http响应头部信息会返回
set-cookie:PHPSESSID=xxxxxxx,浏览器便会自动创建这个cookie，并且关联这个域名。同时，也在服务器端创建一个以Session
ID命名的文件，此时你并没有登录，但是已经给了你相应的session，这时候带着账号密码去登录，登录成功后，Session
ID才能匹配你的信息，否则不登录，在服务端创建的session文件内容为空，登录成功后，在服务端以你的phpsessid创建关于你信息的文件，当你带着这个phpsessid继续

```
POST /login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
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
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/login.php
Cookie: Hm_lvt_9d483e9e48ba1faa0dfceaf6333de846=1551249391,1551253956,1551335342; PHPSESSID=gmk5lg62q6jqfqg8sn4dojfhs7; Hm_lpvt_9d483e9e48ba1faa0dfceaf6333de846=1551335342
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 51

zhanghao=admin&mima=admin&denglu=%E7%99%BB%E5%BD%95
```

而在服务器端，会有一个sess_..... 命名的文件保存用户信息。





这里后台代码仅仅简单写了一个用户权限功能，实际上会记录更全面的用户信息，而里面的信息全部被序列化存储。

session劫持

要明白session劫持，首先需要清楚session id的存储位置，sessionid不仅仅存储在cookie中，也可能存储在post
get（很少见），或者由url重写来完成，还有一部分存储在隐藏域中，一般首选是存储在cookie中，当禁用了cookie传递之后就会重写在url里，在url里传递很容易暴露造成

session劫持实际上就是自己的PHPSESSID被攻击者以某种方式获取，然后在会话的有效期内，利用被攻击者的身份登录网站，来达到身份劫持，伪装成合法用户。一般PHP

session成功劫持一般需要几个步骤，

1.用户访问的平台是使用session来进行身份认证。
2.用户已经使用账号密码登录该平台，随即该用户会得到一个sessionid。
3.通过劫持获取到sessionid，并且在sessionid的有效期内使用（未注销前）。

那么如何获得sessionid是关键，接下来会谈到，一般可以使用

1.劫持：XSS劫持，局域网嗅探，会话固定结合，任意文件读取漏洞等等。
2.爆破：直接通过大流量爆破出sessionid（一般是不太可能）。
3.得到session生成规则，并且得到签名通过计算获取。

规则爆破

通过知道规则来进行爆破，不过这种方法基本上是没可能的局限性太多，且概率极小，php中的sessionid举例而言。生成规则如下。

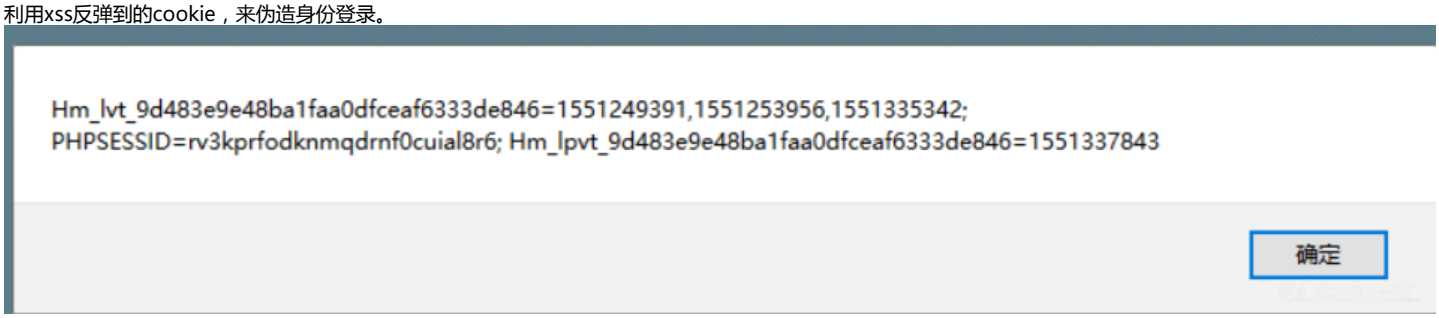
PHPSESSID生成
生成规则是根据hash_func散列来生成的，相关的参数有：

- 客户端IP
- 当前时间（秒）
- 当前时间（微秒）
- PHP自带的随机数生产器

秒微秒可以爆破，随机数也可能是伪随机，客户端IP也可以某种手段获取，但是四种同时精准获取，难度极大，但是不排除有成功的可能性。根据生成规则，也有可能爆破出

利用XSS或者文件读取

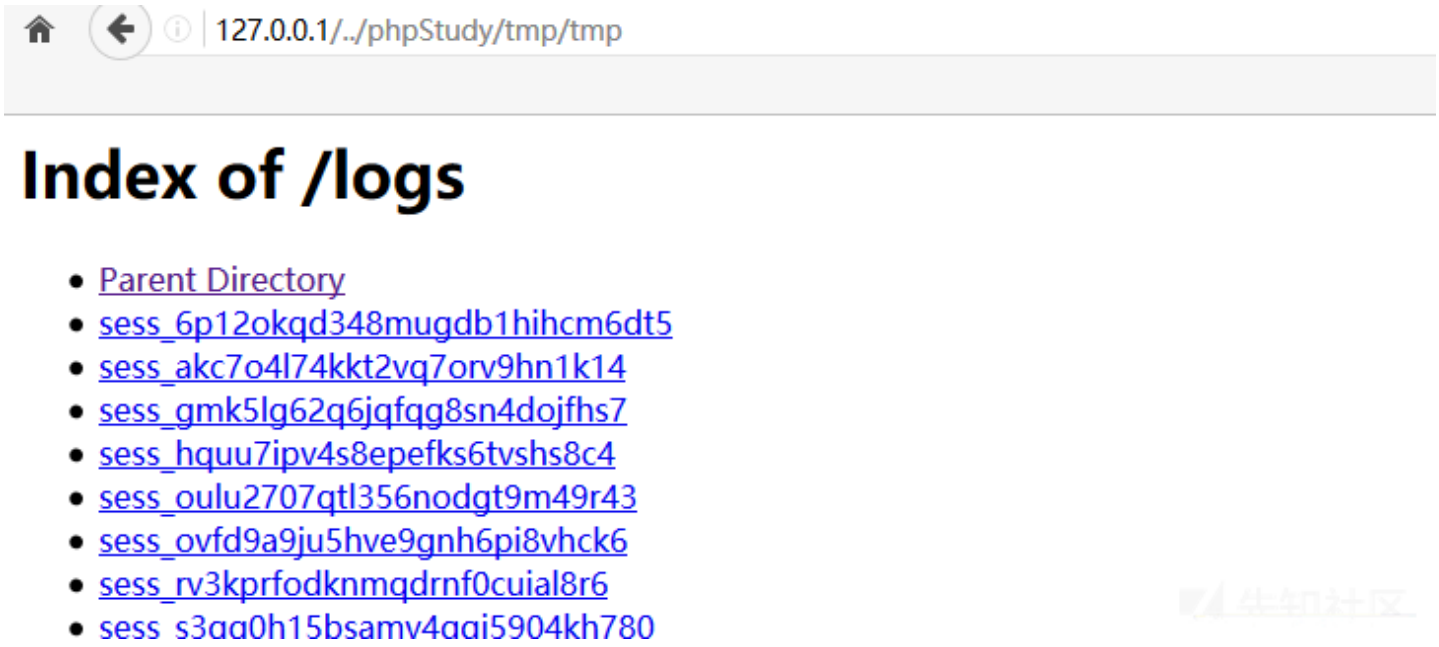
利用xss反弹到的cookie，来伪造身份登录。



使用被攻击者的PHPSESSID，rv3kprfodknmqdrnf0cuial8r6

即可利用管理员的session来登录，前提是必须在会话有效期内使用。

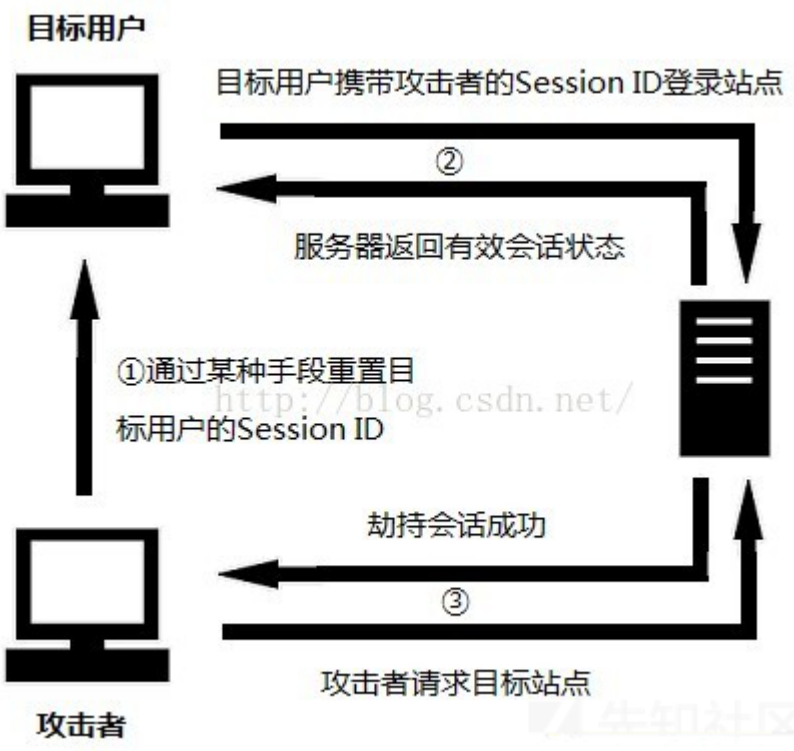
或者利用任意文件读取，XXE等来进行读取文件名来获取sessionid，服务端保存的都是在有效期内的session。



会话固定劫持

会话固定也属于劫持的一种，或者说他是通过攻击者诱导使用固定的sessionid。会话固定的流程主要为

1.攻击者通过构造url参数或者表单传递来固定sessionid。
2.诱使用户使用攻击者构造的参数来登陆。
3.在用户登录之后，攻击者直接使用自己之前构造的固定sessionid来登陆。



会话固定漏洞并不常见，并且在早期通常是使用url来传递sessionid，这样更极大方便了攻击者，因为只要简单构造一条url，并且对于用户是很正常的url，例如，www.taob

1.通过xss漏洞，可以通过js设置cookie中的sessionid。document.cookie="sessionid=test123"
2.通过url传递，直接构造url-->www.taobao.com?sessionid=test123
3.使用Set-Cookie的HTTP响应头部设置Cookie。攻击者可以使用一些方法在Web服务器的响应中加入Set-Cookie的HTTP响应头部,通过自己的服务器页面绑定sessionid。

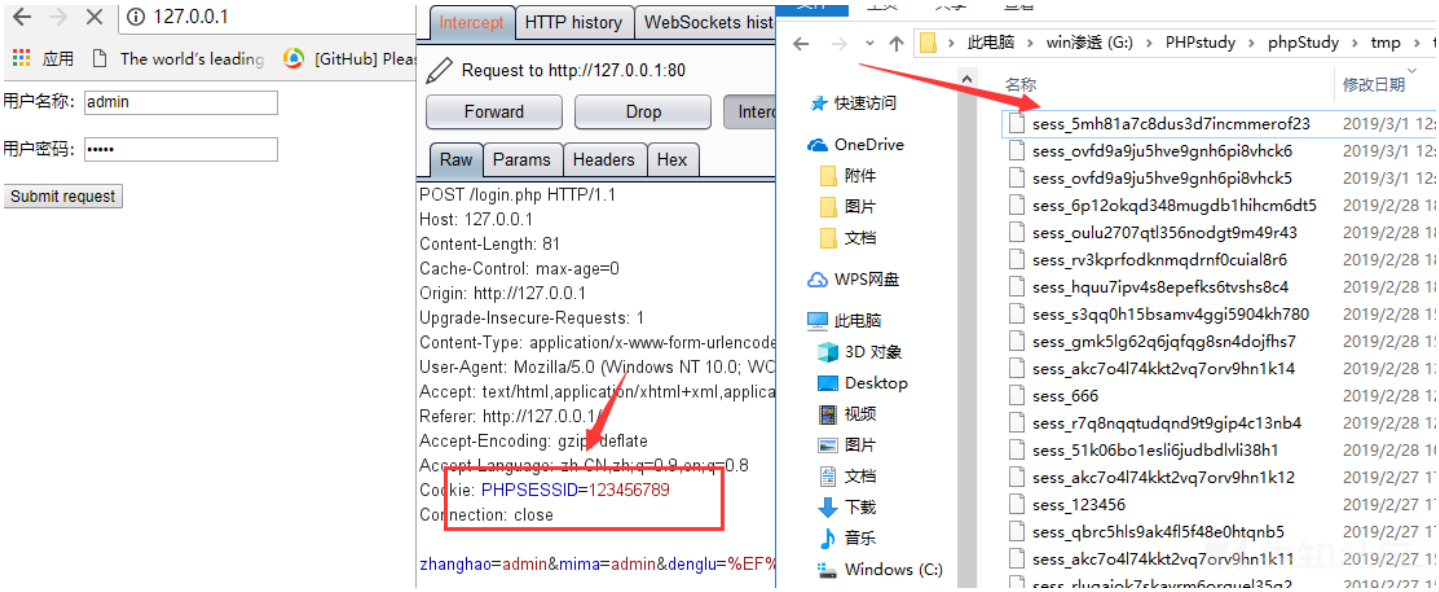
接下来我们通过一个简单的php认证来做一次会话固定劫持，后端对身份验证代码如下。

```
<?php session_start();?>
if($_POST['zhanghao'] == $username && $_POST['mima'] == $password)
   {
```

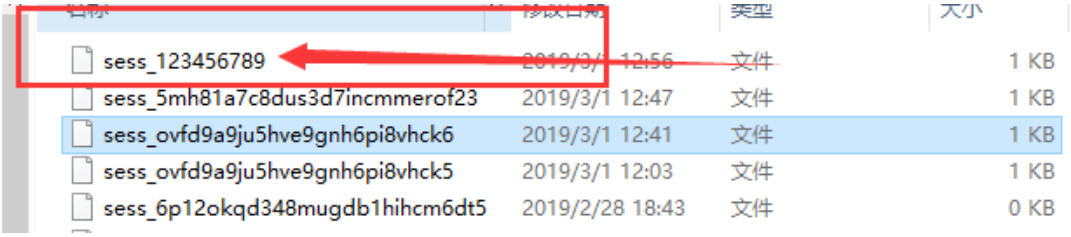
```
   $_SESSION["name"]=$username;
   }

if ($_SESSION['name'] == "")
{
   echo "<script>alert('[redacted]');location.href='index.php'</script>";
}
```

此时攻击者构造一个表单页面。通过js来设置cookie造成会话固定攻击。此时使用户登陆。

```
<html>
   <meta charset="utf-8"  content="text/html; charset=gb2312"/>
     <!-- CSRF PoC - generated by Burp Suite Professional -->
     <body>
     <script>history.pushState('', '', '/')</script>
       <form action="http://127.0.0.1/login.php" method="POST">
       [redacted]<input type="text" name="zhanghao"><br/><br/>
       [redacted]<input type="password" name="mima"><br/><br/>
         <input type="hidden" name="denglu" value="™»½•" />
         <input type="submit" value="Submit request" />
         <meta http-equiv='Set-Cookie' content='PHPSESSID=22333'>
       </form>
       <script type='text/javascript'> document.cookie='PHPSESSID=123456789' </script>
     </body>
   </html>
```



此时抓包可以看到，提交的SESSID是我们构造的，而且服务器端暂时未创建123456789的用户session，然后用户点击提交。



可以看到服务器端成功创建了我们构造的session，然后攻击者便可以通过PHPSESSID=123456789造成合法登陆，由此便造成了一次完整的会话固定劫持，当然如果是url

cookie造成的一些安全隐患

在session身份认证中，一切用户的session都会存储在服务器，但是如果一旦用户量达到一个上限，那么对于服务器的荷载也是巨大的，但是在cookie实现身份认证中，并

在session如果不设置过期时间，在服务器上就会一直存在，永远不会注销，在cookie中如果不设置过期时间，则在关闭浏览器时销毁该cookie，session会给服务器造成很

信息泄露

cookie中存取了大量用户的信息，如身份id，浏览次数，手机号，等一些隐私信息，截取了某平台的cookie信息，存储的信息量非常多，Cookie的主要功能是实现用户个人

Cookie以纯文本的形式在浏览器和服务器之间传送，大多明文的形式被非法用户截获之后无疑是身份被盗取了，并且身份的信息也被盗取了，通过xss漏洞很容易获得cooki

越权

在cookie的身份认证中，极容易造成越权，切大多数越权修改参数都是在cookie中修改，大多数cookie中会有用户的userid，如果未合理的处理或者多重认证，很容易造成

参考：https://blog.csdn.net/h_mxc/article/details/50542038
参考：https://blog.csdn.net/u010084228/article/details/78269345

上一篇：初探php拓展层面(二) 下一篇：初探php拓展层面(二)

1. 0 条回复
   - 动动手指，沙发就是你的了！