PHP中的session安全吗？

如果不做特殊处理，仅是使用PHP中原生的session的话，确实不安全。PHP只是为我们提供了一个session的实现，后续的安全工作需要程序员自己灵活去掌握，所以说PHP编程真的很灵活。

做PHP开发这么长时间，还真没有真正关注过安全的问题，每次都是以完成项目为主，最近在网上看到了一篇关于安全的文章，看完以后才注意到自己以前的项目都存在着很大的安全漏洞，于是挑了一个项目进行了测试，发现很容易就中招儿了。在这里我会分享自己写的一个测试的例子来说明PHP中的session是如何不安全的，以及在项目中如何加强其安全性。

对于session的原理机制，网上有很多好的文章来介绍，我们可以自行查阅。下面直接分享测试用的例子

这个测试的例子主要就是一个登录页，登录成功以后可以修改密码，就这样一个简单的功能。

界面如下





首先是在项目入口的地方使用函数 session\_start() 开启了session。这样当客户端发起请求的时候，会产生一个身份标识 也就是 SessionID。通过cookie的方式保存在客户端，客户端和服务端每次的通信都是靠这个SessionID来进行身份识别的。

登录成功以后，会将 用户id、用户名存入session中

$\_SESSION[‘userid’] = 用户id

$\_SESSION[‘uname’] = 用户名

以后所有的操作都是通过判断 $\_SESSION[‘userid’]是否存在来检查用户是否登录。代码如下：

***if***(*isset*(**$\_SESSION**['userid'])) ***return true***;

对于修改密码接口的调用是通过ajax post的方式将数据传输到服务端的。

$.post("接口\*\*\*\*\*\*\*",

{

oldpass:oldpass,

newpass:newpass,

userid:uid,

},

**function**(data){

data = eval('(' +data+ ')');

$('.grant\_info').html(infos[data.info]).show();

}

);

注意，我这里将这段代码写在了html页面中，所以说如果看到了html代码，也就知道了接口地址了。

修改密码的接口是这样实现的，首先是判断用户是否登录，如果登录才会进行密码的修改操作。

测试例子的实现思路大概就是上面介绍的那样。

利用SessionID攻击

1. 首先是获取SessionID，当然攻击者获取此标识的方式有很多，由于我的水平有限，至于如何获取我在这里不做介绍。我们可以模拟一下，先正常访问此项目，然后通过浏览器查看SessionID，以此得到一个合法的用户标识。可以在请求头中看到此项ID

|  |  |
| --- | --- |
| Accept | text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8 |
| Accept-Encoding | gzip, deflate |
| Accept-Language | zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3 |
| Connection | keep-alive |
| Cookie | Hm\_lvt\_bf1154ec41057869fceed66e9b3af5e7=1450428827,1450678226,1450851291,1450851486; **PHPSESSID=2eiq9hcpu3ksri4r587ckt9jt7;** |
| Host | mobileapp.modernland.hk |
| Referer | http://mobileapp.modernland.hk/ |
| User-Agent | Mozilla/5.0 (Windows NT 6.1; rv:41.0) Gecko/20100101 Firefox/41.0 |

得到sessionID以后，如果此用户登录成功，那么服务端的session里就有此用户的信息了。

1. 获取到SessionID以后，假如攻击者已经知道修改密码的接口，就可以直接修改此用户的密码了。如果攻击者还没有得到接口地址，可以通过查看页面代码找出接口地址。可以使用如下的命令

#curl --cookie "PHPSESSID=2eiq9hcpu3ksri4r587ckt9jt7" 页面地址

上面我们说过，在此例子中ajax代码是写在html页面中的，所以在此页面可以查看到接口地址

部分html代码如下

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

……

var uid = $(".userid").val();

$.post("接口地址\*\*\*\*",

{

oldpass:oldpass,

newpass:newpass,

userid:uid,

},

function(data){

data = eval('(' +data+ ')');

$('.grant\_info').html(infos[data.info]).show();

}

);

……

<span><input type="password" name="oldpass" id="textfield\_o" placeholder="原密码"></span>

<span><input type="password" name="newpass" id="textfield\_n" placeholder="新密码"></span>

<span><input type="password" name="confirmpass" id="textfield\_c" placeholder="确认密码"></span>

1. 得到接口以后可以通过curl 模拟post发送数据来修改密码

命令如下

# curl --cookie "PHPSESSID=2eiq9hcpu3ksri4r587ckt9jt7" -d oldpass=111111 -d newpass=000000 -d userid=用户id 接口地址

如果此用户已经登录，那么攻击者可以通过执行以上命令修改用户的密码。

解决方法

对于以上方式的攻击，我们可以通过使验证方式复杂化来加强其安全性。其中一种方式就是利用请求头中的User-Agent项来加强其安全性

|  |  |
| --- | --- |
| Accept | text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8 |
| Accept-Encoding | gzip, deflate |
| Accept-Language | zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3 |
| Connection | keep-alive |
| Cookie | Hm\_lvt\_bf1154ec41057869fceed66e9b3af5e7=1450428827,1450678226,1450851291,1450851486; PHPSESSID=2eiq9hcpu3ksri4r587ckt9jt7; |
| Host | mobileapp.modernland.hk |
| Referer | http://mobileapp.modernland.hk/ |
| **User-Agent** | **Mozilla/5.0 (Windows NT 6.1; rv:41.0) Gecko/20100101 Firefox/41.0** |

在项目开始的时候最初我们只是用了session\_start()函数来开启session。现在我们可以在session\_start() 下面 添加这段代码

$\_SESSION[‘User\_Agent’] = md5($\_SERVER[‘HTTP\_USER\_AGENT’]);

然后在每次判断是否登录的时候，添加判断条件如下

If(isset($\_SESSION[‘userid’]) && $\_SESSION[‘User\_Agent’] == md5($\_SERVER[‘HTTP\_USER\_AGENT’])) return true;

这样就可以避免上述简单的攻击

总结：

当然，实际情况中的攻击远非这么简单，首先在获取SessionID这一步就比较困难，然后就是和服务端交互的代码尽量加密，可以避免上述的情况。在我们第二次修改代码以后，可以增加攻击的复杂程度，并不能杜绝攻击。攻击的方式多种多样，这里只是一种简单的方式，仅提供一种思路，但是原理是一样的，在实际情况中可以根据实际情况增强我们代码的安全程度。由于我的水平有限，这里只是分享自己在工作中碰到的问题，权当抛砖引玉，如果大家有什么好的意见，请在下面留言，共同讨论，共同提高。