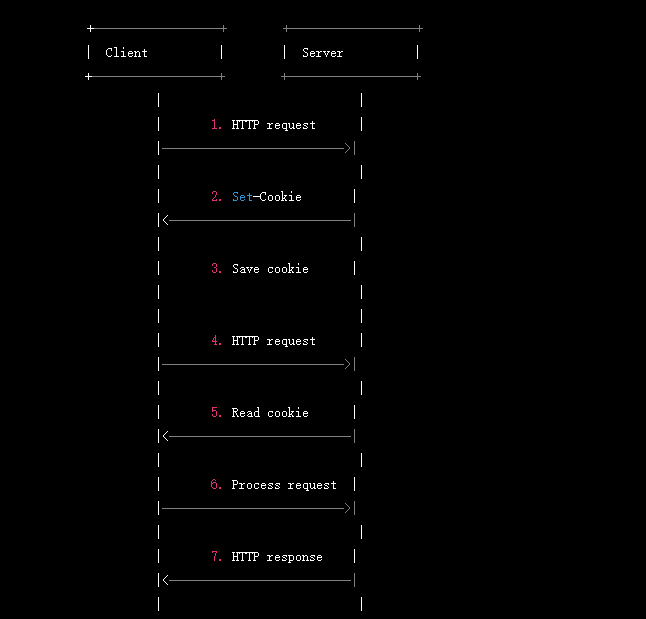
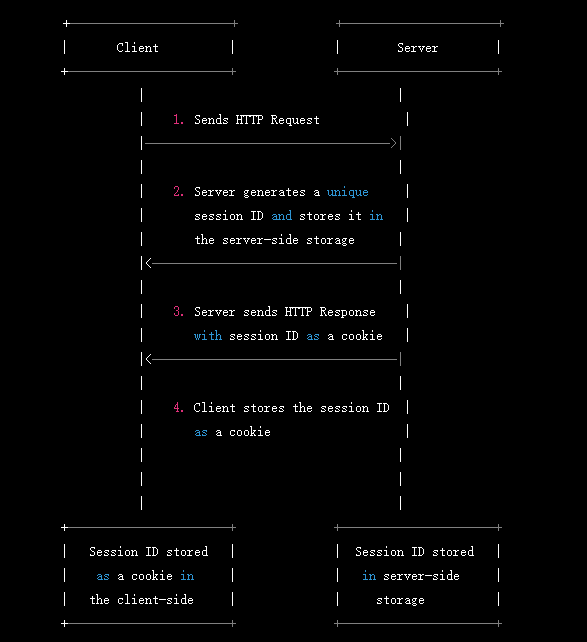
Web开发-PHP应用&Cookie脆弱&Session固定&Token唯一&身份验证&数据库通讯



#知识点：  
1、安全开发-原生PHP-超级全局变量  
2、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
0、安全开发-原生PHP-数据库通讯  
1、安全开发-原生PHP-身份验证技术  
2、安全开发-Cookie&Session&Token  
3、安全开发-原生PHP-代码审计案例  
  
#章节点：(待补充)  
#具体点：(待补充)



演示案例：

* WEB开发-原生PHP-超级全局变量
* WEB开发-原生PHP-代码审计案例
* WEB开发-原生PHP-通讯数据库
* WEB开发-Cookie&Session&Token
* WEB开发-身份验证-代码审计案例

1、数据库操作 验证用户  
login.php->index.php  
  
2、Cookie验证 后台登录  
loginc.php->indexc.php->loginc\_out.php  
  
3、Session验证 后台登录  
logins.php->indexs.php->logins\_out.php  
  
4、Session+Token 防爆破登录  
loginst.php->logincheck.php->indexst.php->loginst\_out.php  
  
5、两例代码审计-Cookie&Session  
XHCMS Cookie脆弱  
YXCMS Session固定  
https://xz.aliyun.com/t/2025  
  
#身份验证-Cookie使用  
生成cookie的原理图过程：见图  
1、客户端向服务器发送HTTP请求。  
2、服务器检查请求头中是否包含cookie信息。  
3、如果请求头中包含cookie信息，则服务器使用该cookie来识别客户端，否则服务器将生成一个新的cookie。  
4、服务器在响应头中设置cookie信息并将其发送回客户端。  
5、客户端接收响应并将cookie保存在本地。  
6、当客户端发送下一次HTTP请求时，它会将cookie信息附加到请求头中。  
7、服务器收到请求并检查cookie的有效性。  
8、如果cookie有效，则服务器响应请求。否则，服务器可能会要求客户端重新登录。  
  
$\_COOKIE：是一个关联数组，包含通过cookie传递给当前脚本的内容。  
setcookie(): 设置一个cookie并发送到客户端浏览器。  
unset(): 用于删除指定的cookie。  
  
  
#身份验证-Session使用  
1、客户端向服务器发送HTTP请求。  
2、服务器为客户端生成一个唯一的session ID，并将其存储在服务器端的存储器中（如文件、数据库等）。  
3、服务器将生成的session ID作为一个cookie发送给客户端。  
4、客户端将session ID保存为一个cookie，通常是在本地浏览器中存储。  
5、当客户端在发送下一次HTTP请求时，它会将该cookie信息附加到请求头中，以便服务器可以通过该session ID来识别客户端。  
6、服务器使用session ID来检索存储在服务器端存储器中的与该客户端相关的session数据，从而在客户端和服务器之间共享数据。  
  
session\_start(): 启动会话，用于开始或恢复一个已经存在的会话。  
$\_SESSION：是一个关联数组，包含当前脚本中的所有session内容。  
session\_destroy(): 销毁当前会话中的所有数据。  
session\_unset(): 释放当前会话中的所有变量。  
Session存储路径：PHP.INI中session.save\_path设置路径  
  
#唯一性判断-Token使用  
1、生成Token并将其存储在Session  
2、生成Token并将其绑定在Cookie触发  
3、尝试登录表单中带入Token验证逻辑  
4、思考Token安全特性  
  
#具体安全知识点：  
-Cookie和Session都是用来在Web应用程序中跟踪用户状态的机制  
1、存储位置不同：  
Cookie是存储在客户端（浏览器）上的，而Session是存储在服务器端的。  
2、安全性不同：  
Cookie存储在客户端上，可能会被黑客利用窃取信息，而Session存储在服务器上，更加安全。  
3、存储容量不同：  
Cookie的存储容量有限，一般为4KB，而Session的存储容量理论上没有限制，取决于服务器的硬件和配置。  
4、生命周期不同：  
Cookie可以设置过期时间，即便关闭浏览器或者重新打开电脑，Cookie仍然存在，直到过期或者被删除。而Session一般默认在浏览器关闭后就会过期。  
5、访问方式不同：  
Cookie可以通过JavaScript访问，而Session只能在服务器端进行访问。  
6、使用场景不同：  
Cookie一般用于存储小型的数据，如用户的用户名和密码等信息。而Session一般用于存储大型的数据，如购物车、登录状态等信息。  
总之，Cookie和Session都有各自的优缺点，选择使用哪一种方式，取决于具体的应用场景和需求。一般来说，如果需要存储敏感信息或者数据较大，建议使用Session；如果只需要存储少量的数据，并且需要在客户端进行访问，可以选择使用Cookie。  
  
-在Web应用程序中，使用token和不使用的主要差异在于身份验证和安全性。  
1.身份验证：采用token机制的Web应用程序，用户在登录成功后会收到一个token，这个token可以在每次请求时发送给服务器进行身份验证。而不采用token机制的Web应用程序，一般会使用session机制来保存用户登录状态，服务器会在用户登录成功后创建一个session，之后的每个请求都需要在HTTP头中附带这个session ID，以便服务器能够验证用户身份。  
2、安全性：采用token机制的Web应用程序，在服务器上不会存储用户的登录状态，只需要存储token即可。因此，即使token被盗取，黑客也无法获得用户的密码或者其他敏感信息。而不采用token机制的Web应用程序，一般会在服务器上存储用户的登录状态，因此如果服务器被黑客攻击，黑客可能会获得用户的敏感信息。  
3、跨域访问：采用token机制的Web应用程序，在跨域访问时，可以使用HTTP头中的Authorization字段来传递token信息，方便实现跨域访问。而不采用token机制的Web应用程序，在跨域访问时，需使用cookie或session来传递用户身份信息，比较麻烦。  
总之，采用token机制可以提高Web应用程序的安全性，并且方便实现跨域访问。不过，使用token机制也需要开发者自己来实现身份验证和token的生成和验证，相对来说比较复杂。而不采用token机制，使用session机制则相对简单，但是安全性相对较低。因此，具体采用哪种机制，需要根据实际情况进行权衡和选择。

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)