**Cookie基础**

Cookie常见的属性:

*Name: cookie名。*

Value: cookie值。

*Domain: cookie的域。如果设成.deepred.com，那么a.deepred.com和b.deepred.com域名下，都可以使用.deepred.com的cookie。（注意前面要加点，不然只有当前域可以访问，子域时访问不到的，如果不设置就是当前域名，前面没点，子域名就无法访问）*

Path: cookie的路径。请求资源的路径一定要包含这个path才能携带cookie。一般设置成/即可。（注意：不设置话就会是上一级path）

*Expires/Max-Age: cookie过期时间。默认不设置，则是Session会话，关闭页面后，该cookie立即失效。*

HttpOnly: 设成true后，JS使用document.cookie则访问不到。常用于避免XSS攻击。

*Secure: 标记为Secure的cookie只应通过被HTTPS协议加密过的请求发送给服务端。*

SameSite: 用来限制第三方Cookie(Node,Lax,Strict)。

Name,Domain,Path确定唯一cookie

客户端可以设置cookie 的下列选项：expires、domain、path、secure（有条件：只有在https协议的网页中，客户端设置secure类型的 cookie 才能成功）(只能设置自身及其父域名下path或者父path的cookie)

**Cookie携带的场景**

**http请求携带cookie必须满足下面几个条件**

**1要在同一个域名下（或是父子域名）**

**2都是http或者https，或者不同的情况下Secure属性为false**

**3要发送请求的路径，即请求端Cookie的path属性必须一致，或者是浏览器端Cookie的path的子目录，比如浏览器端Cookie的path为/test，那么xxxxxxx必须为/test或者/test/xxxx等子目录(如果cookie设置在http://www.com/list中，那么请求http://aaa.www.com/list可以携带，但是**[**http://aaa.www.com**](http://aaa.www.com)**请求是携带不了的)**

**总结一下就是会发送他可以访问到的cookie但是收到同站和同域的限制**

我们假设有一个名字为sessionId的cookie，domain设置成了. xiaokefu.com.cn。

1.在star.xiaokefu.com.cn域名下，ajax在该域名下的所有请求，都会自动带上sessionId

子域名可以访问父域名下的cookie

ajax.get('/api/data') // 自动带上sessionId

2.在liver.xiaokefu.com.cn域名下，ajax在该域名下的所有请求，都会自动带上sessionId

ajax.post('/api2/data2') // 自动带上sessionId

3.在star.xiaokefu.com.cn域名下，ajax请求liver.xiaokefu.com.cn的api，需要设置withCredentials才能带上sessionId, 不能自动带上sessionId

**注意一下： https://**liver.xiaokefu.com.cn**/api/data需要支持cors跨域，并且Access-Control-Allow-Origin不能设成\*，要设置成https://b.demo.com，只有这样，withCredentials才有用**

服务端跨域处理

router.get('/api/data', (ctx, next) => {

ctx.set('Access-Control-Allow-Origin', ctx.headers.origin);

ctx.set('Access-Control-Allow-Headers', 'Content-Type, Content-Length, Authorization, Accept, X-Requested-With , myheader');

ctx.set('Access-Control-Allow-Methods', 'PUT, POST, GET, DELETE, OPTIONS');

ctx.set('Access-Control-Allow-Credentials', 'true');

};

4在www.baidu.com域名下，ajax请求star.xiaokefu.com.cn的api，需要设置withCredentials才能sessionId,不自动带上sessionId

目前为止，都是我们所熟知的cookie携带场景。

然而，在chrome 80版本之后，谷歌把cookie的SameSite属性，从None改成了Lax。这时候，会导致第4种场景sessionId因为**跨站**而导致丢失！（需要手动设置为Node，并且secure设为true）(何为同站就是他们有共同的根域名，[http://a.demo.com](https://link.zhihu.com/?target=http%3A//a.demo.com)和[http://b.demo.com](https://link.zhihu.com/?target=http%3A//b.demo.com)属于同站，[http://a.demo.com](https://link.zhihu.com/?target=http%3A//a.demo.com)和[http://a.demo2.com](https://link.zhihu.com/?target=http%3A//a.demo2.com)属于跨站)

**CSRF攻击**

要完成一次CSRF攻击，受害者必须依次完成三个步骤

1.登录受信任网站A，并在本地生成Cookie。

2.在不登出A的情况下，访问危险网站B

3.B诱导用户发送一个危险请求，因为浏览器中有登录cookie,所以A网站受理了这个请求

常用的解决方法

1登录完成以后，服务器返回一个随机token，存放在cookie或者localstorage中，保证登录态

2登录后每次请求时将token放到请求头或者参数中，由服务端取到后验证正确性，B站是取不到token所以伪造不了正确请求