# 前端安全

## XSS

### XSS是什么

XSS是一种经常出现在web应用中的计算机安全漏洞，它允许恶意web用户将代码植入到提供给其它用户使用的页面中。 比如这些代码包括HTML代码和客户端脚本。攻击者利用XSS漏洞旁路掉访问控制——例如同源策略(same origin policy)。 这种类型的漏洞由于被黑客用来编写危害性更大的网络钓鱼(Phishing)攻击而变得广为人知。 对于跨站脚本攻击，黑客界共识是：跨站脚本攻击是新型的“缓冲区溢出攻击“，而JavaScript是新型的“ShellCode”。

示例：  
<script>alert(document.cookie)</script>

### 特点

能注入恶意的HTML/JavaScript代码到用户浏览的网页上，从而达到Cookie资料窃取、会话劫持、钓鱼欺骗等攻击。

中>

### 原因

* Web浏览器本身的设计不安全。浏览器能解析和执行JS等代码，但是不会判断该数据和程序代码是否恶意。
* 输入和输出是Web应用程序最基本的交互，而且网站的交互功能越来越丰富。如果在这过程中没有做好安全防护，很容易会出现XSS漏洞。
* 程序员水平参差不齐，而且大都没有过正规的安全培训，没有相关的安全意识。
* XSS攻击手段灵活多变。

### 危害

* 盗取各类用户帐号，如机器登录帐号、用户网银帐号、各类管理员帐号
* 控制企业数据，包括读取、篡改、添加、删除企业敏感数据的能力
* 盗窃企业重要的具有商业价值的资料
* 非法转账
* 强制发送电子邮件
* 网站挂马
* 控制受害者机器向其它网站发起攻击

### 如何防范

* 将重要的cookie标记为http only, 这样的话Javascript 中的document.cookie语句就不能获取到cookie了.
* 表单数据规定值的类型，例如：年龄应为只能为int、name只能为字母数字组合。。。。
* 对数据进行Html Encode 处理
* 过滤或移除特殊的Html标签， 例如: <script>, <iframe> , &lt; for <, &gt; for >, &quot for
* 过滤JavaScript 事件的标签。例如 "onclick=", "onfocus" 等等。

参考资料： \* [前端安全系列（一）：如何防止XSS攻击？](https://juejin.cn/post/6844903685122703367)

## CSRF

CSRF（Cross-site request forgery）跨站请求伪造，也被称为“One Click Attack”或者Session Riding，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。尽管听起来像跨站脚本（XSS），但它与XSS非常不同，XSS利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。与XSS攻击相比，CSRF攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比XSS更具危险性。

### 特点

* 攻击一般发起在第三方网站，而不是被攻击的网站。被攻击的网站无法防止攻击发生。
* 攻击利用受害者在被攻击网站的登录凭证，冒充受害者提交操作；而不是直接窃取数据。
* 整个过程攻击者并不能获取到受害者的登录凭证，仅仅是“冒用”。
* 跨站请求可以用各种方式：图片URL、超链接、CORS、Form提交等等。部分请求方式可以直接嵌入在第三方论坛、文章中，难以进行追踪。

### 防御

* 通过referer、token、Cookie 的 SameSite 属性来检测用户提交。
* 尽量不要在页面的链接中暴露用户隐私信息。
* 对于用户修改删除等操作最好都使用post操作 。
* 避免全站通用的cookie，严格设置cookie的域。

[前端安全系列之二：如何防止CSRF攻击？](https://juejin.im/post/5bc009996fb9a05d0a055192)

## CSP

CSP通过指定有效域——即浏览器认可的可执行脚本的有效来源——使服务器管理者有能力减少或消除XSS攻击所依赖的载体。

一个CSP兼容的浏览器将会仅执行从白名单域获取到的脚本文件，忽略所有的其他脚本 (包括内联脚本和HTML的事件处理属性)。

[内容安全策略( CSP )](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/CSP)

## cookies的保护方式

Cookie 中有个属性secure，当该属性设置为true时，表示创建的 Cookie 会被以安全的形式向服务器传输，也就是只能在 HTTPS 连接中被浏览器传递到服务器端进行会话验证，如果是 HTTP 连接则不会传递该cookie信息，所以不会被窃取到Cookie 的具体内容。就是只允许在加密的情况下将cookie加在数据包请求头部，防止cookie被带出来。

另一个是 HttpOnly属性，如果在Cookie中设置了"HttpOnly"属性，那么通过程序(JS脚本、Applet等)将无法读取到Cookie信息，这样能有效的防止XSS攻击。

secure属性是防止信息在传递的过程中被监听捕获后信息泄漏，HttpOnly属性的目的是防止程序获取cookie后进行攻击。

但是这两个属性并不能解决cookie在本机出现的信息泄漏的问题(FireFox的插件FireBug能直接看到cookie的相关信息)。

## 前端鉴权

[前后端常见的几种鉴权方式](https://blog.csdn.net/wang839305939/article/details/78713124)