安全研究员视角下的第三方cookie技术

### 案例：广告平台通过第三方Cookie实现用户跨站追踪

用户访问新闻网站A域名：www.news-a.com，页面嵌入广告联盟ads-track.com的JavaScript脚本：<script src="https://ads-track.com/tracker.js"></script>

浏览器加载脚本时，向ads-track.com发起请求，响应头包含：

Set-Cookie: user\_id=3FQ9X; domain=.ads-track.com; path=/; expires=Mon, 09-Aug-2026 17:20:00 GMT

此Cookie被标记为第三方Cookie（因域名与当前网站不同）。

用户访问电商网站B域名：www.shop-b.com，同样嵌入ads-track.com的广告脚本，浏览器自动携带之前设置的Cookie：

Cookie: user\_id=3FQ9X

广告平台通过user\_id关联两站行为，脚本会将用户的行为数据上报到广告平台后端，根据user\_id关联统一用户，构建用户画像。

### 关键知识补充

#### Cookie属性-domain、samesite属性

浏览器会自动携带所有满足属性值的 Cookie去请求，包括：domain、samesite、httponly等。

#### Cookie域属性控制跨站访问

domain=.ads-track.com允许所有子域名共享Cookie，实现跨合作站点追踪。跨子域需要设置，不跨子域不需要设置。domain是cookie的一个属性。

设置domain=.ads-track.com允许所有子域名共享 Cookie ‌并非对同源策略的突破‌，而是同源策略在 Cookie 处理上的固有特性。Cookie 的同源策略本身包含对子域名的共享支持，当设置 Domain 属性时，浏览器允许同一顶级域名下的子域名访问该 Cookie，例如 track.ads-track.com 可读取 ads-track.com 设置的 Cookie。这源于 Cookie 设计中的灵活性，不同于严格限制的 XHR（XMLHttpRequest）同源策略。

若未设置HttpOnly（也是cookie的属性之一），恶意脚本可通过document.cookie窃取造成XSS（但第三方Cookie通常默认禁止JS访问）。

#### SameSite属性限制

默认SameSite=Lax时，仅允许导航跳转携带第三方Cookie（如点击广告链接）。

**本例子中需设置为SameSite=None且Secure时，允许任意跨站请求携带（需HTTPS）。**

广告脚本需要设置 SameSite=None 的主要原因是：‌虽然广告脚本均来自同一个域名（如 ads-track.com），但其嵌入的网站（如新闻站、电商站等）与广告域名属于不同站点（即跨站场景）‌。浏览器会根据请求来源和目标域名的关系判定是否跨站。

广告脚本需要设置强制Secure是使用 HTTPS 加密传输，防止 Cookie 被窃，与 SameSite=None 绑定使用的。

#### 同源策略

浏览器的同源策略会使得广告脚本（如ads-track.com）无法读取新闻网站（第一方）设置的Cookie。每个域名仅能读写自身设置的Cookie，无法直接访问其他域名的Cookie数据。

#### 本例子中突破同源策略-CORS&SameSite=None且Secure

SameSite=None 作用层级是Cookie 属性，允许浏览器在跨站请求中发送 Cookie。解决“浏览器是否发送 Cookie”的问题。未设置则请求根本不会携带。

CORS作用层级是HTTP 请求/响应头，‌控制服务器是否接受跨域请求及是否允许携带凭证。解决“服务器是否接受带 Cookie 的跨域请求”的问题。未配置则请求被浏览器拦截。服务器响应头包含 Access-Control-Allow-Origin: news.com 和 Access-Control-Allow-Credentials: true。

根据 W3C 的 CORS 标准，当请求需要携带凭证时：1、Access-Control-Allow-Origin 必须指定具体域名（如 https://news.com），不能使用通配符 \*。2、必须配套设置 Access-Control-Allow-Credentials: true。

如果响应头包含 Access-Control-Allow-Origin: \*，浏览器会强制拒绝任何携带凭证的请求（即使客户端设置了 credentials: 'include'）

**CORS和samesite缺一不可。**

### 基于以上的补充知识，重新梳理下上述案例的过程

1、广告脚本—‌浏览器发起跨域请求‌

若请求需携带 Cookie，客户端需显式启用凭证（如 fetch(URL, { credentials: 'include' }) 或 XHR withCredentials=true）。

2、‌浏览器检查 Cookie 的 SameSite 属性‌

仅当 Cookie 满足 SameSite=None; Secure（HTTPS）时，浏览器才会附加 Cookie 到跨域请求中。未设置或值无效（如未配 Secure）时，Cookie 默认被阻止。此处有新闻平台的cookie和广告脚本的cookie，同源策略限制了访问新闻平台就携带新闻平台的cookie，访问广告平台携带广告平台的cookie。

1. 广告平台‌服务器响应 CORS 头部

CORS 预检请求（Preflight）机制

当请求满足以下条件时，浏览器会先发送 OPTIONS 预检请求进行协商：

使用 PUT、DELETE 等非简单方法

包含自定义头部（如 X-Requested-With）

Content-Type 非简单值（如 application/json）

‌预检请求示例‌：

OPTIONS /resource HTTP/1.1

Origin: https://news.com

Access-Control-Request-Method: POST

Access-Control-Request-Headers: X-Custom-Header

‌服务器必须响应‌：

HTTP/1.1 204 No Content

Access-Control-Allow-Origin: https://news.com

Access-Control-Allow-Methods: POST, GET, OPTIONS

Access-Control-Allow-Headers: X-Custom-Header

Access-Control-Allow-Credentials: true

Access-Control-Max-Age: 86400

正式请求的验证流程

‌浏览器发送实际请求‌，携带 Origin 头：

GET /data HTTP/1.1

Origin: https://news.com

Cookie: sessionid=1234

‌服务器响应必须包含：

Access-Control-Allow-Origin: https://news.com // 需明确域名，不能为 \*

Access-Control-Allow-Credentials: true // 允许携带凭证

Access-Control-Allow-Origin和Access-Control-Allow-Credentials缺失任意一项，浏览器将拒绝客户端读取响应，CORS的本质是资源访问控制，即news.com 的脚本能否读取 ads-track.com 的响应‌。CORS不涉及 Cookie 的跨域传递（意思是新闻网站是无法读取携带广告平台cookie的，Cookie 的发送由浏览器底层根据域名自动过滤，这就是浏览器的同源策略）‌

4、‌客户端处理响应‌

仅在 CORS 头部验证通过后，浏览器才将响应数据返回给前端脚本。

下次分享争取不是纯文本，我太懒了

‌