如果你使用76+版本的chrome浏览器，通过开发者面板查看每个网络请求，会发现都有几个Sec-Fetch开头的请求头，例如访问百度首页https://www.baidu.com/的请求：

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

这是用来干嘛的呢，简单来说，就是网络请求的元数据描述，服务端根据这些补充数据进行细粒度的控制响应，换句话说，服务端可以精确判断请求的合法性，杜绝非法请求和攻击，提高web服务的安全性。

Fetch Metadata Request Headers

Sec-Fetch开头的请求头都属于Fetch Metadata Request Headers，于2019年发布的新草案，目前处于Editor's Draft阶段，支持度还不是很高，还需要注意的是，这些请求头都是Forbidden header，也就是不能被篡改的，是浏览器自动加上的请求头，这样也保证了数据的准确性，还需要注意的是如果资源是本地缓存加载，那么就不会添加这些请求头了，这也容易理解，就不多说了。

规范的意义

近些年web领域发展迅速，但是安全问题也十分突出，从最初浏览器的同源模型到CSP，再到Fetch Metadata Request Headers，都是对web安全不断的完善和加强，以往很多安全策略侧重于客户端的防护，服务端需要识别非法请求往往比较困难，因为缺乏判断请求的依据，控制比较粗线条，而Fetch Metadata Request Headers的出现就为服务端过滤非法请求提供了元数据，避免csrf,xssi等攻击就很容易了。

接下来探究一下这四个请求头的含义；

Sec-Fetch-Dest

含义：

表示请求的目的地，即如何使用获取的数据；

取值范围：

说明：

Dest是destination的缩写，根据上面的取值范围可很容易理解了，这个请求头指明客户端请求的目的，期望需要什么样的资源；

Sec-Fetch-Mode

含义

该请求头表明了一个请求的模式；

取值范围：

cors：跨域请求；

no-cors：限制请求只能使用请求方法(get/post/put)和请求头(accept/accept-language/content-language/content-type)；

same-origin：如果使用此模式向另外一个源发送请求，显而易见，结果会是一个错误。你可以设置该模式以确保请求总是向当前的源发起的；

navigate：表示这是一个浏览器的页面切换请求(request)。 navigate请求仅在浏览器切换页面时创建，该请求应该返回HTML；

websocket：建立websocket连接；

说明：

cors表示跨域请求，且要求后端需要设置cors响应头；no-cors并不是代表请求不跨域，而是服务端不设置cors响应头，什么情况下会是这种模式呢，图片/脚本/样式表这些请求是容许跨域且不用设置跨域响应头的，而no-cors也是默认的模式；same-origin表示同源请求，这就限制了不能跨域，前面说的cors和no-cors是容许跨域的，只是要求服务端的设置不同而已，熟悉fetch接口的同学对mode属性应该不陌生，其实跟这里的含义是一样的，只是fetch的mode大家可以手动设置，而Sec-Fetch-Mode不能干预而已；

Sec-Fetch-Site

含义：

表示一个请求发起者的来源与目标资源来源之间的关系；

取值范围：

cross-site：跨域请求；

same-origin：发起和目标站点源完全一致；

same-site：有几种判定情况，详见说明；

none：如果用户直接触发页面导航，例如在浏览器地址栏中输入地址，点击书签跳转等，就会设置none；

说明：

same-site有几种情况(A->B)：

A B same site

(" https ", " example.com ") (" https ", " sub.example.com ") true

(" https ", " example.com ") (" https ", " sub.other.example.com ") true

(" https ", " example.com ") (" http ", " non-secure.example.com ") false

(" https ", " r.wildlife.museum ") (" https ", " sub.r.wildlife.museum ") true

(" https ", " r.wildlife.museum ") (" https ", " sub.other.r.wildlife.museum ") true

(" https ", " r.wildlife.museum ") (" https ", " other.wildlife.museum ") false

(" https ", " r.wildlife.museum ") (" https ", " wildlife.museum ") false

(" https ", " wildlife.museum ") (" https ", " wildlife.museum ") true

在地址有重定向的情况下，Sec-Fetch-Site取值稍微复杂一点，直接参考一下示例：

1.https://example.com/ 请求https://example.com/redirect，此时的Sec-Fetch-Site 是same-origin;

2.https://example.com/redirect重定向到https://subdomain.example.com/redirect，此时的Sec-Fetch-Site 是same-site （因为是一级请求二级域名）;

3.https://subdomain.example.com/redirect重定向到https://example.net/redirect，此时的Sec-Fetch-Site 是cross-site （因为https://example.net/和https://example.com&https://subdomain.example.com/是不同站点）;

4.https://example.net/redirect重定向到https://example.com/，此时的Sec-Fetch-Site 是cross-site（因为重定向地址链里包含了https://example.net/）;

Sec-Fetch-User

含义：

取值是一个Boolean类型的值，true(?1)表示导航请求由用户激活触发(鼠标点击/键盘)，false(?0)表示导航请求由用户激活以外的原因触发；

取值范围：

?0

?1

说明：

请求头只会在导航请求情况下携带，导航请求包括document , embed , frame , iframe , or object ；

安全策略

了解了上面是个请求头的含义之后，我们就可以根据项目实际情况来制定安全策略了，例如google I/O提供的一个示例：

# Reject cross-origin requests to protect from CSRF, XSSI & other bugs

def allow\_request(req):

# Allow requests from browsers which don't send Fetch Metadata

if not req['sec-fetch-site']:

return True

# Allow same-site and browser-initiated requests

if req['sec-fetch-site'] in ('same-origin', 'same-site', 'none'):

return True

# Allow simple top-level navigations from anywhere

if req['sec-fetch-mode'] == 'navigate' and req.method == 'GET':

return True

return False

1.浏览器不支持Sec-Fetch-\*请求头，则不做处理；

2.容许sec-fetch-site为same-origin, same-site, none三种之一的请求；

3.容许sec-fetch-mode为navigate且get请求的方法；

4.容许部分跨域请求，可设置白名单进行匹配；

5.禁止其他非导航的跨域请求，确保由用户直接发起；

在使用Fetch Metadata Request Headers时，还需要注意Vary响应头的正确设置，Vary这个响应头是干嘛的呢，其实就是缓存的版本控制，当客户端请求头中的值包含在Vary中时，就会去匹配对应的缓存版本(如果失效就会同步资源)，因此针对不同的请求，能提供不同的缓存数据，可以理解为差异化服务，说明白了Vary响应头之后，就明白了Fetch Metadata Request Headers与Vary的影响关系了，因为要确保缓存能正确处理携带Sec-Fetch-\*请求头的客户端响应，例如Vary: Accept-Encoding, Sec-Fetch-Site，因此有没有携带Sec-Fetch-Site将会对应两个缓存版本。

————————————————

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

原文链接：https://blog.csdn.net/juruiyuan111/article/details/114937963