你了解 CSRF 攻击么？

网络千万条，安全第一条。网安不规范，网站都完蛋！

对于前端工程师来说，遇到最多的网络安全是 XSS 攻击，上一篇文章已经给大家说过了。今天我们来看看前端工程师遇到第二多的网络安全问题 CSRF 攻击。

什么是 CSRF 攻击？

CSRF 概念：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 则通过伪装成受信任用户的请求来利用受信任的网站。

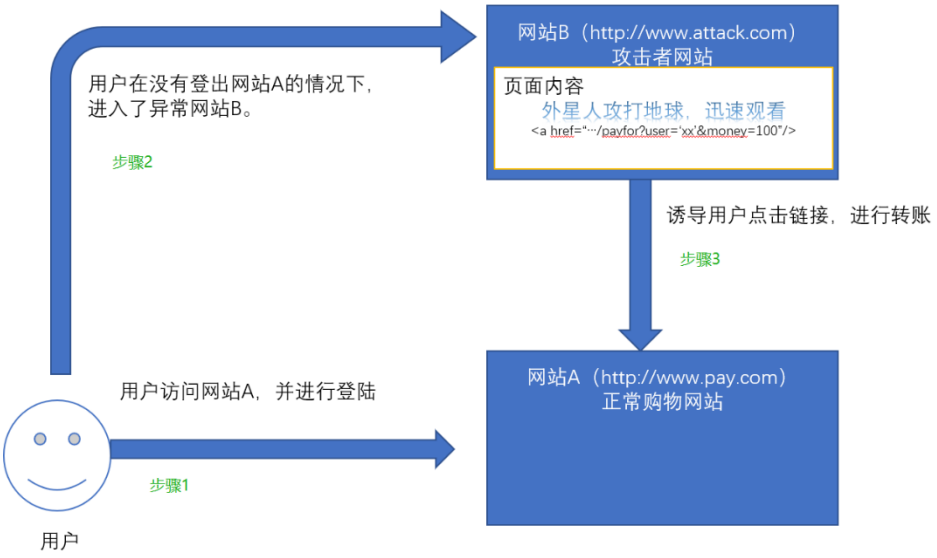
与 XSS 攻击相比，CSRF 攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比 XSS 更具危险性。

造成 CSRF 攻击原理

发送请求自动带上 cookie：浏览器中 HTTP(s) 请求是会自动帮我们把 cookie 带上传给服务端的。这样在每次请求的时候通过 cookie 获取 session id，然后通过它在服务端获取登录信息即可完成用户权限的校验。

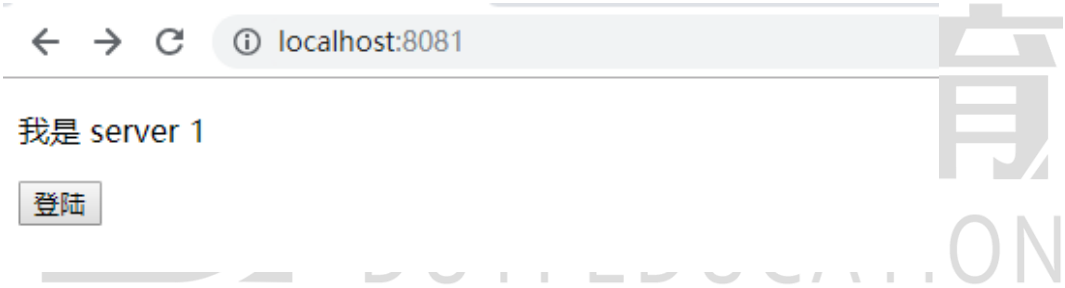
本来这也是个不错的功能。但是由于 cookie 实在是太开放了，如果一个用户在 A 网站登录了，并且在没有登出的情况下（没有清除本地信息，包括 cookie），用户在 B 网站访问的时候发送了一个 A 网站的请求，那么这个请求其实是带有这个用户在 A 网站的登录信息的。如果这时候 B 站发送的 A 网站请求是用户不知道的，并且可以进行用户非本意操作（获取信息，修改内容，转账等等）。那就是非常严重的危害了。以上的过程就是跨站请求攻击，即 Cross-Site Request Forgery，即 CSRF。

下面来看一张图:



接下来按照上面的步骤，基于 NodeJS 服务的代码来模拟 CSRF 攻击。

1. 用户访问网站 A（localhost:8081）



网站 A 的页面结构，以及相关交互。

server1.html

```
<p>我是 server 1</p>
  <button>登陆</button>
  <script>
      //
      $('button').on('click', function () {
          $.ajax({
              type: 'GET',
              url: '/login',
              success(data) {
                  alert('成功')
              }
          });
```

```
        })

    </script>
```

服务器 A 的相关代码

server1.js

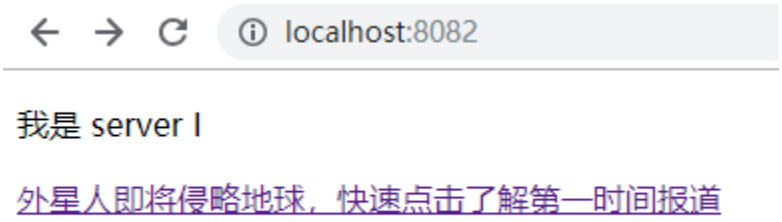
```
const express = require('express');
const cookieParser = require('cookie-parser');
const url = require('url');
const server1 = express();
server1.use(cookieParser());
server1.use(express.static('./server1'));
// 设置登陆接口
server1.get('/login', function (req, res) {
    res.cookie('id', 3000)
    res.end('ok')
});

// 设置查询接口
server1.get('/getMoney', function (req, res) {
    res.end("" + server1Money);
});

// 设置支付接口
server1.get('/payMoney', function (req, res) {
     // 存在用户 cookie id 几个进行交易。
    if (req.cookies.id) {
        // 执行交易过程
        const query = url.parse(req.url, true).query;
        users[query.user] += +query.money;
        users.server1 -= +query.money;
        res.json(users);
    // 不存在跳转到登陆界面，不允许操作。
    } else {
        res.redirect('/')
    }
});
// 监听 8081 端口
server1.listen(8081, function () {
```

```
    console.log('server is running at 8081');
});
```

2. **用户点击登陆后，离开网站 A，访问网站 B（localhost:8082）**



3.
```
<p>我是 server I</p>
<a href="http://localhost:8081/payMoney?money=10000&user=server2">外星人即将侵略地球，快速点击了解第一时间报道</a>
```

4. **当用户点击链接够，调用的其实是支付的接口。**

经历了这样的过程，之后完成了一次 CSRF 攻击，就是利用了浏览器里面 cookie 使用过于便利，当从有问题的网站发送出来，还是会获取 cookie 获取权限，最终导致这样的问题。

CSRF 的危害

简单总结 CSRF 漏洞就是利用网站权限校验方面的漏洞在用户不知觉的情况下发送请求，达到“伪装”用户的目的。攻击者利用 CSRF 实现的攻击主要有以下几种：

1. 攻击者能够欺骗受害用户完成该受害者所允许的任一状态改变的操作，比如：更新账号细节，完成购物，注销甚至登录等操作
2. 获取用户的隐私数据
3. 配合其他漏洞攻击
4. CSRF 蠕虫

其中 CSRF 蠕虫如其名所指就是产生蠕虫效果，会将 CSRF 攻击一传十，十传百。如：某社区私信好友的接口和获取好友列表的接口都存在 CSRF 漏洞，攻击者就可以将其组合成一个 CSRF 蠕虫——当一个用户访问恶意页面后通过 CSRF 获取其好友列表信息，然后再利用私信好友的 CSRF 漏洞给其每个好友发送一条指向恶意页面的信息，只要有人查看这个信息里的链接，CSRF 蠕虫就会不断传播下去，其可能造成的危害和影响非常巨大！

从本质上讲，CSRF 漏洞就是黑客将一个 http 接口中需要传递的所有参数都预测出来，然后不管以什么方式，他都可以根据他的目的来任意调用你的接口，对服务器实现 CURD 操作，就是对服务器中的数据进行增删改查。

如何防范 CSRF 攻击

从上文的描述中我们可以知道 CSRF 有两个特点：利用 cookie 自动携带的特性以及跨站攻击。那么针对这两个特性可以使用如下解决方法。

检查 Referer 字段

大家都知道 HTTP 头中有一个 Referer 字段，这个字段用以标明请求来源于哪个地址。通过在网站中校验请求的该字段，我们能知道请求是否是从本站发出的。我们可以拒绝一切非本站发出的请求，这样避免了 CSRF 的跨站特性。

这种方式利用了客户端无法构造 Referrer 的特性，虽然简单，不过当网站域名有多个，或者经常变换域名的时候会变得非常的麻烦，同时也具有一定的局限性。

Token 验证

由于 CSRF 是利用了浏览器自动传递 cookie 的特性，另外一个防御思路就是校验信息不通过 cookie 传递，在其他参数中增加随机加密串进行校验。为每一个提交增加一个指定串参数，该参数服务端通过页面下发，放到 localStorage 中，每次请求的时候补充到提交参数中，服务端通过校验该参数是否一致来判断是否是用户请求。由于 CSRF 攻击中攻击者是无从事先得知该字符串，所以服务端就可以通过校验该值拒绝可以请求。