SSO 单点登录 （2:09:40-end）

概念

单点登录SSO（Single Sign On）说得简单点就是在一个多系统共存的环境下，用户在一处登录后，就不用在其他系统中登录，也就是用户的一次登录能得到其他所有系统的信任。单点登录在大型网站里使用得非常频繁，例如像阿里巴巴这样的网站，在网站的背后是成百上千的子系统，用户一次操作或交易可能涉及到几十个子系统的协作，如果每个子系统都需要用户认证，不仅用户会疯掉，各子系统也会为这种重复认证授权的逻辑搞疯掉。实现单点登录说到底就是要解决如何产生和存储那个信任，再就是其他系统如何验证这个信任的有效性，因此要点也就以下两个：

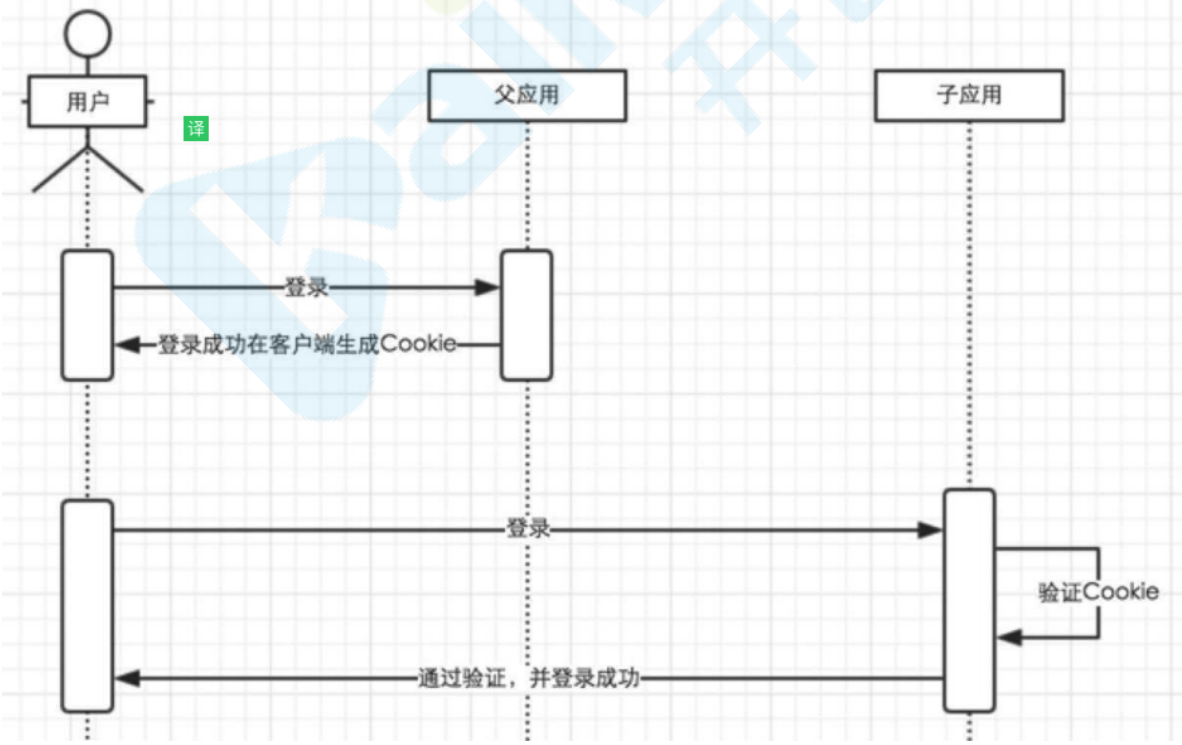
- 存储信任
- 验证信任

> https://blog.csdn.net/qq_39089301/article/details/80615348

实现方式

- Cookie为凭证媒介

  最简单的单点登录实现方式，是使用cookie作为媒介，存放用户凭证。
  用户登录父应用之后，应用返回一个加密的cookie，当用户访问子应用的时候，携带上这个cookie，授权应用解密cookie并进行校验，校验通过则登录当前用户。



不难发现以上方式把信任存储在客户端的Cookie中，这种方式很容易令人质疑：

- Cookie不安全
- 不能跨域实现免登

对于第一个问题，通过加密Cookie可以保证安全性，当然这是在源代码不泄露的前提下。如果Cookie的加密算法泄露，攻击者通过伪造Cookie则可以伪造特定用户身份，这是很危险的。
对于第二个问题，更是硬伤

- JSONP方式

  对于跨域问题，可以使用JSONP实现。
  用户在父应用中登录后，跟Session匹配的Cookie会存到客户端中，当用户需要登录子应用的时候，授权应用访问父应用提供的JSONP接口，并在请求中带上父应用域名下的Cookie，父应用接收到请求，验证用户的登录状态，返回加密的信息，子应用通过解析返回来的加密信息来验证用户，如果通过验证则登录用户

这种方式虽然能解决跨域问题，但是安全性其实跟把信任存储到Cookie是差不多的。如果一旦加密算法泄露了，攻击者可以在本地建立一个实现了登录接口的假冒父应用，通过绑定Host来把子应用发起的请求指向本地的假冒父应用，并作出回应。
因为攻击者完全可以按照加密算法来伪造响应请求，子应用接收到这个响应之后一样可以通过验证，并且登录特定用户。

- **使用独立登录系统**

一般说来，大型应用会把授权的逻辑与用户信息的相关逻辑独立成一个应用，称为用户中心。
用户中心不处理业务逻辑，只是处理用户信息的管理以及授权给第三方应用。第三方应用需要登录的时候，则把用户的登录请求转发给用户中心进行处理，用户处理完毕返回凭证，第三方应用验证凭证，通过后就登录用户。