最近，一位之前一直在寻找它的用户在边肖向我们提出了一个问题。相信这也是很多币圈朋友经常疑惑的问题：如何保证单点登录的令牌不被盗，令牌单点登录的原理，带着这个问题，让专业的编辑来告诉你为什么。

单点登录顾名思义就是所有被管理设备的访问，通过堡垒机器完成。这意味着网络中的所有主机都必须通过堡垒机器才能访问它。那么这个安全保障的依赖就是网络限制和会话限制。。安全的机制就在于这种依赖的会话监控手段，所以所有通过堡垒机器访问设备的操作，理论上都是有记录的。该记录可用于审计、检查非法运营商、IP、操作内容、终端信息等。

寒假学习小话题整理记录之前的笔记(长预警)因为当时看到的东西涉及很多，有些地方没有深入讨论。

百度百科：单点登录，缩写为SSO。，这是最流行的企业业务集成解决方案之一。SSO的定义是，在多个应用系统中，用户只需要登录一次，就可以访问所有可信的应用系统。

简而言之，用户处于多个可信应用系统中。您只需登录一次即可访问其他可信应用系统。这里的关键是一次登录和一次注销对所有系统都有效。

在普通登录中，比如典型的B/S场景，浏览器访问服务器，发送登录请求。发送用户名和密码后，服务器会生成用户&#039；会话来规范用户&#039；的状态，比如登录或注销，并给浏览器一个cookie。因此，用户在继续访问时会带上这个cookie。服务器会根据这个Cookie找到对应的会话，通过会话判断这个用户的登录状态。比如在php中使用phpsessid。当然，您也可以定制会话的生命周期。如果一个会话的生命周期太长，一旦会话被盗，用户就会被盗。同时，生命周期长的会话配置会占用更多的服务器资源。

单点登录主要是针对同一平台下的多个应用。一种多系统场景下多重登录的解决方案。单点登录相当于连接多个应用的认证系统。

假设现在一个平台上有三个具有登录功能的应用，从上面的普通登录情况可以想到。，这三台服务器将记录自己的会话。那么为了实现单点登录，出现了一种最简单的方法：共享会话。

共享会话是实现单点登录最方便直接的方式。。在这三个子系统中，我们使用相同的附加服务器来记录会话。例如，我们可以使用redis服务器来存储三个系统的会话。

用户登录到应用程序1，并获得应用程序1返回的cookies。，用cookies再次访问应用程序1的其他功能被登录，但是有一个新的问题。虽然实现了共享会话，但是用户登录应用1，得到应用1返回的cookies。然而，因为饼干可以&#039；t跨域，用户可以&#039；不要使用应用程序1中的cookies来访问应用程序2。这里我们需要将系统的全局cookie域的属性设置为顶级域名，比如应用1的域名是1.test.com。应用2的域名是2.test.com。在正常登录的情况下，应用1的cookie域的属性是1.test.com，也就是说这个cookie只能在这个子域中使用。。我们将系统的全局cookie域设置为顶级域名，即test.com，这样用户可以登录应用1，然后在登录状态下访问应用2和3。

在上面共享会话的情况下，三个应用都有登录功能，还有一个类似的情况，应用1和应用2都有登录模块和其他模块，还有一个单独的SSO系统只有登录模块：

共享会话的方法虽然简单，但是有局限性，因为使用了cookie顶级域的特性，所以不能跨域。一个公司或一个平台可能没有一个一级域名下的所有域名。，所以同一个域名下的单点登录不是完整的单点登录。

Let&#039；让我们先来谈谈openid。openid是一个认证标准，规定了如何认证！也就是说，它侧重于登录时的身份认证。官方场景。一方是用于存储已注册openid账户的openid身份服务器，另一方是该openid身份服务器信任的服务或应用。。Openid协议提供openid身份服务器和可信服务或应用程序之间的通信。比如我们可以用QQ登录很多网站，腾讯&#039；这里的QQ是openid&#039的身份服务器。我们要登录的网站是受信任的服务或应用程序。

使用openid实现单点登录的方法有很多种。可以用上面共享会话的方法，就是把openid带入cookie。但是，这也会造成cookie同样的跨域问题。

在实际场景中，当我们访问提供服务的应用程序时，当我们检测到我们没有登录时，我们会直接跳转到openid身份服务器。，或者点击选择使用第三方openid在登录表单附近登录，无需重定向，使用账号密码登录(这样可以保证我们登录的服务器无法获取我们的敏感认证信息)。具体流程如下：

CAS，全称CentralAuthenticationService，是面向企业的多语言单点登录解决方案，致力于成为认证授权需求的综合平台。。CAS是单点登录的现成demo，企业只需简单修改即可使用。

CAS支持多种协议，如SAML、OAuth、OpenID、OIDC等，支持LDAP、Radius、JWTX。、509等等，还有各种常用语言的客户端，比如Java、PHP、C#等等。不管怎么说，它是一个非常完整和兼容的SSO框架。

了解CAS如何实现单点登录。。你可以在官网上看到流程图。这个图特别详细，一目了然，可以直接在原图上标注：

学习了以上单点登录的知识，结合实际场景，可以看出跨域单点登录才是真正的单点登录因为现实中很多平台或者域名都可以&#039；不一定都在一级域名下。在解决跨域单点登录问题时，上面提到了几种方式，但其根本是使用一个SSO认证中心来实现认证和授权。当然跨域单点登录还会有其他的解决方案，但大致流程和cas差不多。

例如，在上图的步骤11中，您还可以使用POST包，或者JSONP和iframe方法跨域发送重定向请求。

使用认证中心实现单点登录是一种常见的解决方案，那么有没有不使用认证中心解决跨域单点登录的解决方案呢？

使用JSONP同步登录状态，一般流程如下：

在学习单点登录的过程中，对认证过程中授权令牌的传递等相关信息不是特别详细。而且，在考虑单点登录的时候，会有一个比较矛盾的问题：单点登录的目标是让用户在一个相互信任的系统中登录一次，但是如果所有用户真的可以访问所有系统，就赢了&#039；这不会带来越权的问题吗？是不是要对不同的用户授予不同的授权，甚至限制对应用的访问，但这不是原来狭义的单点认证吗？

在讨论单点登录的认证和授权之前，让&#039；让我们谈谈统一身份认证和单点登录的区别，我&#039；我一直想弄明白。。说起单点登录，可能很少听说，但是统一身份认证肯定不陌生，无论是企业还是高校都会有这样的统一身份认证系统。

统一身份认证最重要的一个方面是身份认证，另一个是与身份认证相关的授权控制。，访问控制。单点登录是对多个应用的登录，也可以称为统一登录，可以理解为主要是在认证方面。对于统一身份认证，会有账户管理，比如LDAP、认证管理OAuth、SMAL等。所以我认为统一身份认证一般包括狭义单点登录，狭义单点登录，即多个应用只需要一次登录。但是现在，SSO系统不仅需要这些东西，它的范围也在慢慢扩大。

单点登录的身份验证和授权。在前面提到的CAS中单点登录的实现中，您会看到需要ticket来进行身份验证和授权。CAS支持多种认证方案，如OAuth、OpenID、SAML等。我们可以比较这些协议之间的差异。，或者在哪些情况下适合使用哪些身份验证方案。单点登录本身没有权限控制功能，但是由于这些认证协议的要求，它自然支持权限控制。

在使用SAML进行认证的过程中，可以看到下图。基本流程差不多。需要注意的是，用户成功登录认证中心后，重定向时会返回SAML令牌和XML节点。这里的令牌将包括用户&#039；的身份信息，用户名等等。

在OAuth2.0的标准中，流程基本和上面一样，但是因为OAuth2的客户端根本不是浏览器，所以令牌默认是不签名的。这里可能没有反映出来，但是OAuth2的目标是授权所以token更注重权威。token在向认证服务器认证时会有不同的授权，但既然是授权，就间接实现了认证。

传统身份验证基于会话机制。具体的会话模式上面也提到了。根据其特点可以知道，

API接口的安全性主要是保证数据不会被篡改和重复调用。实现方案主要围绕三种机制设计：令牌、时间戳和签名。

1。令牌授权机制

用户使用用户名和密码登录后，服务器向客户端返回一个令牌(必须是唯一的。，可以结合UUID和本地设备标识)，并将Token-UserId以键-值对的形式存储在缓存服务器中(我们使用Redis)，并设置过期时间。服务器在接收到请求后验证令牌，如果令牌不存在，，表示请求无效。令牌是客户端访问服务器的证书。

2。时间戳超时机制

每次用户请求时，都会带来当前时间的时间戳。收到时间戳后，服务器比较当前时间。如果时间差大于某个时间(如30秒)，则认为请求无效。时间戳超时机制是防止重复调用和抓取数据的有效手段。

当然，这里需要注意的是确保&quot当前时间&quot客户端和服务器的一致性。我们采用的对齐方式是客户端第一次连接服务器时，请求一个接口获取服务器的当前时间A1。，然后和客户端的当前时间B1做差分计算(A1-B1=AB)，得到差值AB，在客户端后续的请求中传输给服务器。

3。API签名机制

用MD5算法加密请求的API参数、时间戳和salt，加密后的数据就是这个请求的签名。服务器收到请求后，会用同样的算法得到签名，并与当前签名进行比较。如果不一样，，表示参数已经更改，直接返回错误标识。签名机制确保数据不会被篡改。

4。注意事项5。安全摘要

在上述机制下，

如果有人劫持请求，修改请求中的参数，签名就不会通过；

如果有人利用劫持的网址进行DOS攻击和数据抓取，那么他只能使用30s最多；

签名算法全部泄露怎么办？可能性很小，因为&quot盐&quot这里的价值只有我们自己知道。

本文你将看到：

&quot；前端存储&quot这个涉及一个发射，一个存储，一个区域，很容易发。登录界面直接返回前端，所以前端需要想办法存储。

前端的存放方式有很多种。

是，饼干.Cookie也是前端存储的一种，但是相比于localStorage等其他方式，借助HTTP头和浏览器能力，可以让cookie前端无感知。一般流程如下：

&quot；配置：域/路径

cookie用于限制：&quot空间范围&quot:它经过两级：域/路径。

&quot；配置：过期/最长期限&quot

cookie还可以限制：&quot时间范围&quot*到期和最大年龄之一。

&quot；配置：Secure/httponly&quot；

cookie可以限制为：&quot用法&quot:

&quot；HTTP头读写cookies&quot回过头来看，HTTP是如何编写和交付cookies及其配置的？？HTTP返回的Set-Cookie头用于写入&quot一个(且只有一个)&quot浏览器的cookie。格式为cookie键值配置键值。比如：

如果我想一次设置更多cookie该怎么办？给出更多Set-

Cookie头(HTTP请求中允许重复)

浏览器使用HTTP请求的Cookie标头发送符合当前配置&quot空间、时间和用途&quot到服务器。因为浏览器已经做了过滤判断，所以不需要返回配置内容，只需要发送键值即可。。

&quot；前端读写cookies&quot前端可以自己创建cookies。如果服务器创建的cookie没有添加HttpOnly，那么恭喜你，你也可以修改他给的cookie。。调用[xss_clean]可以创建和修改cookie。和HTTP一样，[xss_clean]一次可以并且只能操作一个cookie。

也可以通过调用[xss_clean]来读取cookie，可以像HTTP一样读取所有非HttpOnly cookie。

现在回想一下，你刷卡的时候发生了什么？？

此操作在前端和后端认证系统中称为会话。典型的会话登录/验证过程：

&quot；会话的存储模式&quot显然，服务器只给cookie一个sessionId，以及会话的具体内容(可能包括用户信息、会话状态等。)应该是自己救的。有几种存储方式：

&quot；会话&quot**很简单，销毁存储的会话数据即可。****&quot会话分布式问题&quot**通常情况下，服务器是一个集群，当用户请求负载平衡时，会进行负载平衡。，不一定要哪个机器。那么一旦机器被用户请求&#039；的后续接口与他请求登录的机器不一致，或者请求登录的机器停机，是不是&#039；会话无效吗？现在这个问题有几种解决方案。

但是，通常采用第一种方法，因为第二种方法相当于阉割负载均衡，仍然不能解决&quot用户要求的机器停机时间&quot。**&quot**node.js下的会话处理**前面的图很清楚。在服务器端实现对cookie和session的访问还有很多事情要做。在npm中，已经有打包的中间件，比如express-session-
npm，所以不公布用法。。这是另一种cookie:

Express-Session——NPM主要意识到：

Session的维护给服务器造成了很大的麻烦，必须找个地方存放。，还要考虑分配的问题，甚至要单独为它启用一套Redis集群。有没有更好的办法？回顾过去，你不会&#039；对于登录场景，不必在会话中保存太多东西。为什么不直接打包成饼

干呢？这样，服务器不会&#039；不需要保存它，只需检查&quot证书&quot每次都是cookie带来的，而且还可以携带一些轻量级的信息。这种方法通常被称为令牌。

token的过程如下：

&quot；客户端令牌的存储方法&quot如前所述，cookie不是客户端存储凭证的唯一方式。。因为它的&quot无国籍&quot，token&#039的有效期和使用限制都被包裹在token的内容中，对cookie的管理能力依赖更小，客户端保存更自由。但是web应用的主流方式还是放在cookie里。毕竟，唐&#039；别担心。"令牌到期&quot那么我们如何控制令牌的到期日期呢？它&#039；这很简单。只要把&quot过期时间&quot和数据结合在一起，并在验证时进行判断。

编码方式丰富节俭。。**&quotbase64&quot**比如在节点端的cookie-session-npm库

默认配置中，我给他一个userid，他会这样保存：

。

eyj1c2vyawqiojin0=这里只是{&quotuserid&quot:&quotabb&quot}.防篡改

是。所以看情况。如果令牌涉及敏感权限，我们必须找到防止令牌被篡改的方法。解决方案是对令牌进行签名，以识别它是否被篡改。例如，在cookie-session-npm库中。，添加两个配置：

，这将创建一个.sigcookie，其中的值由{&quotuserid&quot:&quotabb&quot}和iAmSecret通过加密算法。,commonsuchasHMACSHA256class(system.Security.Encryption)|MicrosoftDocument.

Allright.现在cdd可以伪造eyj1c2vyawqiojin0=，但是它可以&#039；不要伪造sig的内容，因为它不会&#039；我不知道这个秘密。**&quotJWT&quot；**但上述做法增加了cookie的数量。数据本身没有标准格式，于是JSONWebToken简介——jwt.io诞生了。

它是一个成熟的令牌字符串生成方案。，它包含我们前面提到的数据和签名。为什么不看看JWT代币长什么样：

这一串东西是怎么来的？看图：

类型，加密算法选项。和JWT标准数据字段，可以参考RFC7519JSONwebtoken(jwt)节点，也有相关的库实现：express-jwtNPMKOA-jwtNPM。

token，作为权利的守护者，最重要的是&quot安全性&quot。业务接口用于身份验证的令牌称为访问令牌。企业对特权越敏感。我们越希望访问令牌的有效期足够短，以避免被窃取。然而，有效期太短将导致访问令牌频繁过期。过期了怎么办？一种方法是让用户重新登录以获得新的令牌，这显然不够友好。您知道，一些访问令牌可能会在几分钟后过期。另一种方式是拥有另一个令牌，一个专门生成访问令牌的令牌，我们称之为刷新令牌。

有了刷新令牌，几种情况的请求流程就变成了这样：

如果刷新令牌也过期了，就只能重新登录了。

会话和令牌都是边界模糊的概念。如前所述，刷新令牌也可以以会话的形式组织和维护。在狭义上我们通常认为会话是一种&quot植入cookie和存储在服务器上的数据，而token是一种&quot存储在客户端的任何位置，数据存储在令牌&quot。。会话和令牌的比较实质上是&quot客户端保存cookie/位置&quot和&quot服务器保存数据/不保存数据&quot。。**&quot客户端保存cookie/其他位置&quot**Cookies保存方便，但问题也很明显：

异地存储可以解决没有cookie的场景；手动磁带通过参数等方式。，可以避开CSRF的攻击。正如我们之前所知，&quot服务器存储数据/不存储数据&quot

，在同一个域下的客户/服务器认证系统中，通过携带凭证的客户端维持一段时间的登录状态。。但是，当业务线越来越多时，更多的业务系统会分散在不同的域名下，这就需要&quot单点登录，所有线路通用&quot，这叫做&quot单点登录&quot。

简单，如果业务系统都在同一个主域名下。例如，Wenku.baidu.comtieba.baidu.com就很容易。你可以直接将cookie域名设置为主域名Baidu.com，这就是百度所做的。

比如滴滴这样知名度的公司，拥有didichuxing.com、xiaojukeji.com、didiglobal.com等域名。而这种饼干是完全避免不了的。这将启用&quot单点登录&quot，全面通用，是真正的单点登录。在这种情况下，我们需要一个独立的认证服务，通常称为SSO。"从系统A到系统B的完整过程，无需登录&quot

&quot；完整版：考虑浏览器的场景&quot以上流程好像没问题。其实对于很

多app来说已经足够了。但是在浏览器中可能不好用。看这里：

对于一个浏览器来说，如何保存SSO域下返回的数据，才能在访问A时带走？浏览器对跨域有严格限制，cookie、localStorage等方法有域限制。这就要求A只能提供在A域下存储凭证的能力。。一般我们是这样做的：

在图中，我们用颜色标注浏览器当前所在的域名。注意图中灰色文字描述的变化。

谢谢。

令牌是收据。不过，比机票温柔多了。如果你把票弄丢了，你得花钱重新买一张。如果您丢失了令牌，您可以再次验证它。因此，丢失令牌的成本是可以忍受的。——前提是你不&#039；不要经常丢失它。如果您让用户偶尔进行一次身份验证，将会失去用户体验。。X0dx0ax0dx0a客户端：除非你有非常安全的方法，比如操作系统提供的私有数据的存储，那么token肯定会被泄露。比如我拿你的手机，抄你的代币。过期前可以在其他地方以你的身份登录。解决这个问题的简单方法。存放时，对称存放令牌，及时解锁。。X0dx0a2。结合请求URL、时间戳和令牌，添加盐和签名，服务器验证有效性。这两种方法的出发点是更容易窃取你存储的数据。很难分解你的程序黑客，你的加密，解密和签名算法。不过说难不难，毕竟是防君子防小人的做法。。也就是说，一个客户端的加密存储，如果被人打开了，就不会以纯文本的形式存储了？X0dx0a方法一：可以&#039；t获取存储的密文；方法二：它不&#039；t不知道你的签名算法和盐，可以一起吃。X0dx0a但是如果token被戴上手铐，他可以很自然的植入到手机里，那么他的手机也可以当你用，你就瞎了。然后X0dx0a可以提供一个机制让用户主动过期，类似于之前的令牌，被盗时可以远程止损。。X0dx0a一个人如果会怎么谈安全&#039；连他的手机都不保护？X0dx0ax0dx0a令牌在网络层面明文传输会非常危险，建议使用HTTPS，将令牌放在postbody中。。

心血来潮，让&#039；讨论cookie和会话的机制以及单点登录的原理。如有可能欢迎指正&#039；t.

要说单点登录，还得说一下cookie机制。这些知识也是面试中的高频问题。小马过去喜欢说话，如果有人帮忙的话。它&#039；只是大部分学生要么概念模糊，要么完全放弃这个知识点，往往被忽略。

cookie机制是一种会话机制，起源于无状态HTTP协议。。当浏览器发起的http请求想要保持与服务器的会话状态时，需要中介媒介的帮助，cookie机制正好。

Let&#039；举个栗子。假设我现在访问王者农药官网。

1。打开浏览器，输入xxx域名；

2。浏览器会在硬盘中寻找关于xxx的Cookie。通常，登录状态标识符存储session_id。，然后将Cookie放入HTTP请求中，然后将请求发送到Web服务器；

3。服务器识别cookie。，将在服务器上查询和检查对应于session_id的用户会话记录。如果存在且有效，则通过执行此登录数据操作(如写入基本登录会话数据)，用户将被视为正常登录状态。这时候浏览器看到的就是登录状态。如果没有相关信息，则用户不会登录。

4。如果没有登录，用户将在浏览器中登录。成功登录后向浏览器写入一个cookie(session_id)。同时，服务器会有相应的session_id会话记录(默认为文件存储)。当浏览器关闭或用户注销(服务器删除登录的会话)时，会话结束，会话信息(session_id)变为无效。当然有一个例外，就是如果设置了cookie的有效期，会话消息会保存到过期，客户端的cookie和服务器的会话信息仍然有效。想想这个&quot自动登录&quot我们通常在登录时看到的检查。，下次访问可以直接登录，这是原则。

嗯，那&#039；cookie和会话之间的会话机制。

从上面我们可以看出，cookie大致可以分为两类：会话cookie和持久cookie。

会话Cookie是记录用户&#039；对网站的访问。，记录用户访问网站时的设置和偏好；关闭浏览器，会话Cookie将被删除。

永久Cookie存储在硬盘上，无论浏览器是否退出或计算机是否重新启动，它们都将继续存在。。永久Cookie有一个过期时间。

它们的共同点是都存储在客户端(浏览器)和硬盘上。不同的浏览器和不同的操作系统可能会将Cookie存储在不同的位置。。然后自然会话存储在服务器上(注意敲黑板会要求提问)。session的存储方式有哪些？浏览器禁用cookie后还能使用会话吗？默认情况下，

会话保存文件。当然也可以配置成DB存储或者缓存存储，比如redis。浏览器禁用cookie后还能使用会话吗？是的，如果将session_id放在url参数中，服务器也可以识别会话信息。但是，处理起来就没那么优雅了。说白了，cookie只是一种更合适的客户端存储介质。对于一些禁用了浏览器cookie的用户来说，这是兼容的，但是很少，基本忽略。

我们说到点子上了，什么是单点登录？？

单点登录(简称SSO)是企业业务集成的流行解决方案之一。单点登录被定义为多个应用系统。用户只需登录一次，即可访问所有可信应用系统。

Let&#039；让我们抛出一个问题来帮忙。如果我的服务器现在是集群，如何处理登录状态问题？比如我现在还是农药官网，成功登录网站xxx。但是第二次访问的时候，被告知已经登录了。我又刷了一下，显示已经登录了。这是什么鬼东西？原因是如果xxx域名下的服务器是分布式集群，同时采用默认文件存储方式存储session会话信息。。因此，第一次处理请求时，机器A存储了文件会话信息。第二个会话带着cookie去服务器的时候，正好B机在处理请求，所以找不到会话信息文件，自然浏览器以为没有登录。

以上问题可以通过在服务器上共享登录信息，修改session到DB或redis缓存的存储方式实现多机共享，解决单域名下的分布式集群登录共享。

如果是同一个域名下的几个服务器，在顶级域名下设置cookie的路径(比如域名xxx有a.xxx和b.xxx子域)。，以便所有子域都可以读取cookie中的令牌(或会话id)。但是这里有一个问题。如果是跨域的，可以&#039；我吃不到饼干。你该怎么办？因为cookie没有跨域性质。如果不同的域名xxx和ccc想要共享登录状态，共享同一个cookie是无法解决的。于是就有了令牌机制，用一个令牌证书来保持各个系统之间的登录状态。

使用令牌认证机制

要实现SSO，需要以下主要功能(本段参考百科描述):

所有应用系统共享一个身份认证系统。统一的身份认证系统是单点登录的先决条件之一。。认证系统的主要功能是比较用户&#039；的登录信息与用户信息库，并验证用户&#039；的登录。认证成功后，认证系统要生成统一的认证票，并返回给用户。此外，认证系统还应该验证票证。，来判断其有效性。

所有应用系统都可以识别和提取票证信息。为了实现单点登录的功能，让用户只登录一次，应用系统必须能够识别已经登录的用户。应用系统应该能够识别和提取票证。通过与认证系统的通信，可以自动判断当前用户是否已经登录，从而完成单点登录的功能。

小马觉得有点像微信接口调用的access_token机制。

那&#039；对于如何保证单点登录的令牌不被窃取的介绍已经足够了。感谢您花时间阅读本网站的内容。唐&#039；别忘了搜索更多关于令牌单点登录原理以及如何

确保单点登录令牌不被窃取的信息。