# SAML and OAuth and OIDC

## SSO 单点登录

**减少密码管理负担**：用户无需记住多个应用的不同凭据，提高安全性和便利性。

SAML为了认证设计

OAuth实际是为了授权而设计，OIDC（OpenID Connect）基于OAuth2.0实现，加入了认证协议

认证：确认该用户的身份是他所声明的那个人

授权：根据用户的身份授予他访问特定资源的权限

可以这样来思考身份验证和授权之间的区别：假设爱丽丝要参加音乐节。在入口处，她出示门票和另外一种身份证明，以证实她有权持有该门票。之后，她被允许参加音乐节。她通过了身份验证。

然而，仅仅因为爱丽丝位于音乐节场地内，并不意味着她可以随意走动、为所欲为。她可以观看音乐节表演，但不能登台表演，也不能到后台与演员互动，因为没有获得相应的授权。如果她购买了后台通行证，或者除了观众之外还担任表演者，那么她将获得更大的授权。

### 单点登录是什么

单点登录（SSO，Single Sign On），允许用户通过身份提供商（IdP）进行一次身份验证即可访问多个应用程序，它的核心目标是减少用户在不同系统之间重复输入用户名和密码的繁琐操作，提高用户体验和工作效率

比如是在企业内部多个应用系统（如考勤系统、财务系统、人事系统等）场景下，用户只需要登录一次，就可以访问多个应用系统。

同理用户只需注销一次，就可以从多个应用系统退出登录。

简单来说就是，一次登录，全部登录！一次注销，全部注销！！

#### 跨域单点登录

跨域单点登录（Cross-domain SSO）指的是在**多个不同的顶级域名**之间实现单次登录即可访问多个系统。例如：

* a.com 登录后，可以直接访问 b.com 的相关资源，无需再次输入凭据。

**典型应用场景**

* 企业级应用（一个企业的多个子系统分布在不同域名）

* 统一身份认证（OAuth2, OpenID Connect）

* 大型平台（如 Google 账户可用于 google.com、youtube.com、gmail.com 等）

### 跨域与同域 SSO 的区别

|  |  |  |
| --- | --- | --- |
| **对比项** | **同域 SSO** | **跨域 SSO** |
| **适用范围** | 子域（a.example.com和 b.example.com） | 不同顶级域名（a.com 和 b.com） |
| **技术方案** | 共享 Cookie（Domain=example.com） | 跨域通信（OAuth2、JWT、Token 传递） |
| **浏览器限制** | 允许（只要 Domain=主域） | 受 SameSite Cookie 限制，需要 CORS、第三方存储等 |
| **安全性** | 依赖 HTTPS 和 HttpOnly/Secure Cookie | 需要 Token 传递，可能涉及 CSRF、Token 窃取等 |
| **实现复杂度** | 较低（同一域下 Cookie 共享） | 较高（需要额外授权、跨域通信机制） |

跨域 SSO 的安全核心在于 **Token 传递的安全性**，如果实现不当，可能带来 Token 窃取、重放攻击、CSRF 等问题，因此需要严格的安全策略。

### 2.1 SSO 核心构成

* **中心认证服务器**

SSO 系统通常包含一个中心认证服务器，它负责对用户的身份进行验证。当用户首次访问某个应用系统时，该应用系统会将用户重定向到中心认证服务器进行登录

* **票据（Ticket）机制**

用户在中心认证服务器成功登录后，服务器会生成一个包含用户身份信息的票据（通常是一个加密的字符串），并将该票据返回给用户的浏览器。用户的浏览器会将这个票据保存在 Cookie 中或者作为 URL 参数传递给后续访问的其他应用系统。

* **应用系统验证**

当用户访问其他应用系统时，这些应用系统会从用户的请求中获取票据，并将其发送回中心认证服务器进行验证。如果票据验证通过，应用系统就会认为用户已经通过身份验证，允许用户访问相应的资源。

## SAML 2.0

2005

基于XML的开源标准协议

XML：可扩展标记语言，是一种简单的数据储存传输的语言

主要目的是提供跨域浏览器web单点登录

SAML 2.0 通过身份提供者（Identity Provider, IdP）和服务提供者（Service Provider, SP）之间的信任关系来实现 Web 端的单点登录（Single Sign-On, SSO）

### 场景

SAML 2.0 的应用场景

SAML 2.0 在企业级环境中得到了广泛应用，特别是在以下几种情况下：

云服务集成：许多云服务提供商（如 阿里云、腾讯云、Salesforce、Google Workspace、Microsoft Azure AD）都支持 SAML 2.0，使得企业能够轻松地将内部身份管理系统与外部云服务连接起来。

跨组织协作：当不同组织之间的员工需要共享资源时，SAML 2.0 提供了一种标准化的方法来管理和验证用户身份。

B2B 和 B2C 应用：在商业伙伴或消费者之间建立信任关系，确保只有经过认证的用户才能访问敏感信息。

政府和公共部门：用于保护公民个人信息，确保在线服务的安全性。

### 流程

发起请求：用户尝试访问一个受保护的服务或应用，即服务提供者。如果用户尚未认证过，服务提供者会重定向用户到身份提供者页面进行认证。

身份验证：用户到达身份提供者登录页面后，输入凭证（如用户名和密码）进行身份验证。

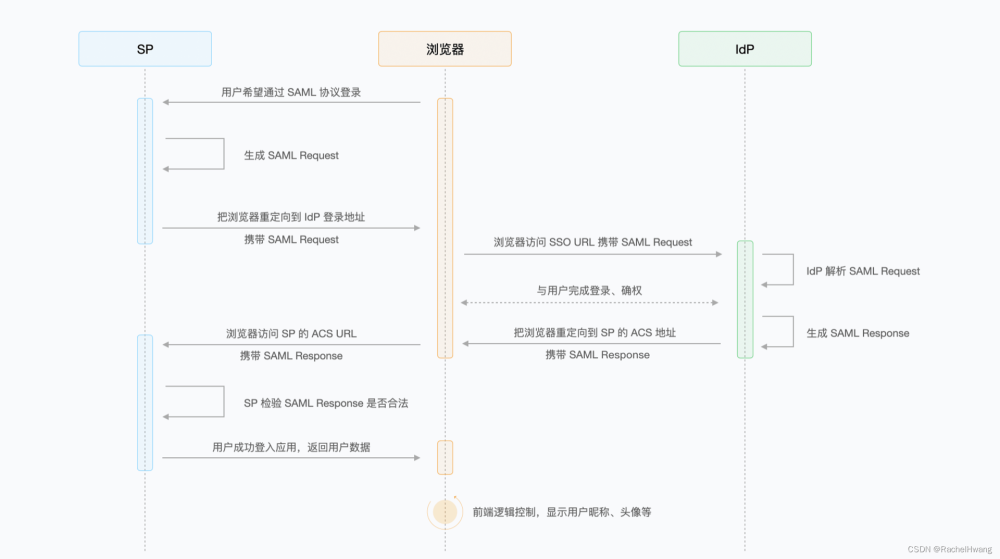
生成断言：用户成功通过身份验证后，身份提供者会创建一个包含用户身份信息的 SAML 断言。此断言包含了关于用户的身份声明、认证上下文等信息。

发送断言：身份提供者将 SAML 断言封装到响应消息中，并将其发送回服务提供者。这通常通过 POST 方法完成，其中 SAML 响应作为表单数据的一部分。

验证断言：服务提供者接收到 SAML 响应后，需要验证断言的有效性。这包括检查数字签名以确保消息未被篡改，以及确认签发者是可信的身份提供者。

授予访问权限：如果断言验证成功，服务提供者可以信任用户的身份，并根据其策略决定是否授予用户访问权限。

用户会话建立：一旦用户被授权，服务提供者将创建一个会话，允许用户在一定时间内无需重新认证即可访问资源。



### 消息传输

SAML 的 Binding 是指在SAML生态系统中传输和接收SAML消息的协议规范和格式。介绍几种 SAML 2.0 中常见的 Binding。

HTTP Redirect Binding：将 SAML 消息编码为 URL 参数，并通过 HTTP 重定向发送。这种方式简单易用，但受限于 URL 长度，适用于传输较小的 SAML 消息，如 AuthnRequest 和 LogoutRequest。

HTTP POST Binding：将 SAML 消息作为 HTML 表单的隐藏字段，通过 HTTP POST 方法发送。这种方式可以传输更大的 SAML 消息，更安全，因为它不会将消息暴露在 URL 中。

HTTP Artifact Binding：将 SAML 消息的一个引用（称为 Artifact）通过 HTTP 传输。这种方式允许将实际的 SAML 消息存储在更高效的后端存储中，并减少传输的数据量。但是，它需要一个额外的步骤来检索实际的 SAML 消息。

SOAP Binding：将 SAML 消息嵌入到 SOAP 消息中，通过 SOAP 协议传输。这种方式适用于需要在客户端和服务器之间进行复杂交互的场景，但增加了实现的复杂性。

<https://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf>

### 部署

TODO

部署 **SAML（Security Assertion Markup Language）** 协议时，主要的配置内容涉及到身份提供者（Identity Provider, IdP）和服务提供者（Service Provider, SP）之间的信任建立和数据交换。SAML通常用于单点登录（SSO）场景，涉及用户认证和授权的信息交换。以下是部署SAML协议时主要的配置内容：

#### 1. 身份提供者（IdP）配置

身份提供者负责验证用户身份，并生成包含用户信息的SAML断言（SAML Assertion）。IdP的配置通常包括以下内容：

* **IdP 元数据**：IdP需要生成元数据文件，包含该身份提供者的基本信息和如何与SP进行通信的细节。例如，IdP的实体ID、证书（用于签名断言）、SAML响应的回调URL等。

* **证书和密钥管理**：IdP需要配置用于签名SAML断言的证书和密钥，以确保响应的完整性和来源的可信性。

* **单点登录URL（SSO URL）**：这是用户进行单点登录请求时需要访问的URL。通常由IdP提供，SP会将用户重定向到该URL进行身份验证。

* **单点注销URL（SLO URL）**：如果启用了单点注销（Single Logout, SLO），IdP需要配置处理注销请求的URL。

* **属性映射**：IdP需要配置如何映射其内部用户信息到SAML断言中的属性（如用户名、邮箱、角色等），这些属性会被发送到SP。

* **协议和加密设置**：IdP需要配置是否使用加密SAML断言，以确保安全传输。

#### 2. 服务提供者（SP）配置

服务提供者是向用户提供服务的应用，SP的配置主要是为了接受IdP发来的SAML断言，并基于这些断言进行用户身份验证和授权。SP的配置通常包括以下内容：

* **SP 元数据**：SP需要生成自己的元数据文件，包含其实体ID、证书、SSO回调URL等信息，供IdP进行信任建立。

* **SSO URL（Assertion Consumer Service, ACS URL）**：这是SP接收IdP发送的SAML断言的URL，IdP会将用户的SAML响应发送到此URL。

* **证书和密钥管理**：SP需要配置用于验证IdP签名的证书，以确保收到的SAML断言没有被篡改。

* **实体ID**：SP的唯一标识符，用于标识该服务提供者。

* **属性映射**：SP需要配置如何从SAML断言中提取用户的属性并将其映射到本地应用的用户信息字段，如用户名、角色等。

* **断言处理配置**：SP需要配置如何验证SAML断言（例如，检查签名、验证时间戳等）以及如何处理成功的认证或认证失败的情况。

* **单点注销URL（SLO URL）**：如果启用了单点注销，SP需要配置处理注销请求的URL，以便与IdP配合使用。

#### 3. 信任关系建立

* **证书交换**：IdP和SP之间需要交换证书，以便相互信任。IdP会将其公钥（用于签名）传送给SP，SP会将其公钥（用于验证SAML断言）传送给IdP。

* **元数据交换**：IdP和SP需要交换元数据文件，确保双方知道彼此的身份和配置。例如，IdP需要知道SP的ACS URL，SP需要知道IdP的SSO URL。

#### 4. SAML断言配置

* **断言签名**：IdP通常会对生成的SAML断言进行签名，以确保其没有被篡改。SP需要配置相应的公钥以验证签名。

* **断言加密**：SAML断言可以进行加密，以确保只有SP能够解密和读取其中的内容。IdP和SP需要配置加密和解密的密钥。

* **断言有效期**：SAML断言通常有时间戳和有效期，IdP需要确保断言在有效期内发出，SP需要验证断言的有效期。

#### 5. 用户授权和角色映射

* **角色/权限映射**：根据SAML断言中的属性（例如用户角色或组），SP可能需要对用户进行权限映射，确定该用户在系统中具有的访问权限。

## OAuth 2.0

2012

OAuth 是一种开放式授权标准，OAuth 使用 JSON 数据包和 API 调用，是现代 Web 的理想选择。

OAuth 2.0 是一种授权框架，允许第三方应用在用户授权下访问其存储在服务提供者上的资源，而无需共享用户凭证。用于不信任第三方应用时，允许用户授权第三方应用访问他们存储在另外的服务提供者上的信息，而不需要将用户名和密码提供给第三方应用或分享他们数据的所有内容。

既方便密码管理，也防止密码泄露。

### OAuth 的应用场景

* **第三方应用授权**

如前面提到的图片编辑应用访问云存储照片的例子，OAuth 为各种第三方应用提供了一种安全、便捷的授权方式，让用户可以自由选择授权哪些应用访问自己的资源

* **社交网络平台开放 API**

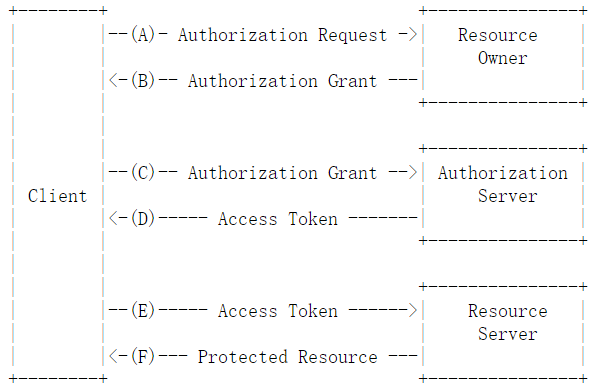
社交网络平台通常通过 OAuth 允许第三方开发者基于其平台开发应用。这些应用可以在用户授权的情况下获取用户在社交网络上的部分信息，如发布动态、获取好友列表等，从而丰富了社交网络平台的生态

* **移动应用授权**

在移动应用领域，OAuth 也被广泛应用。例如，一些移动支付应用需要访问用户的银行账户信息，通过 OAuth 可以在保证安全的前提下获取用户的授权。

<https://zhuanlan.zhihu.com/p/717770658>

### 流程





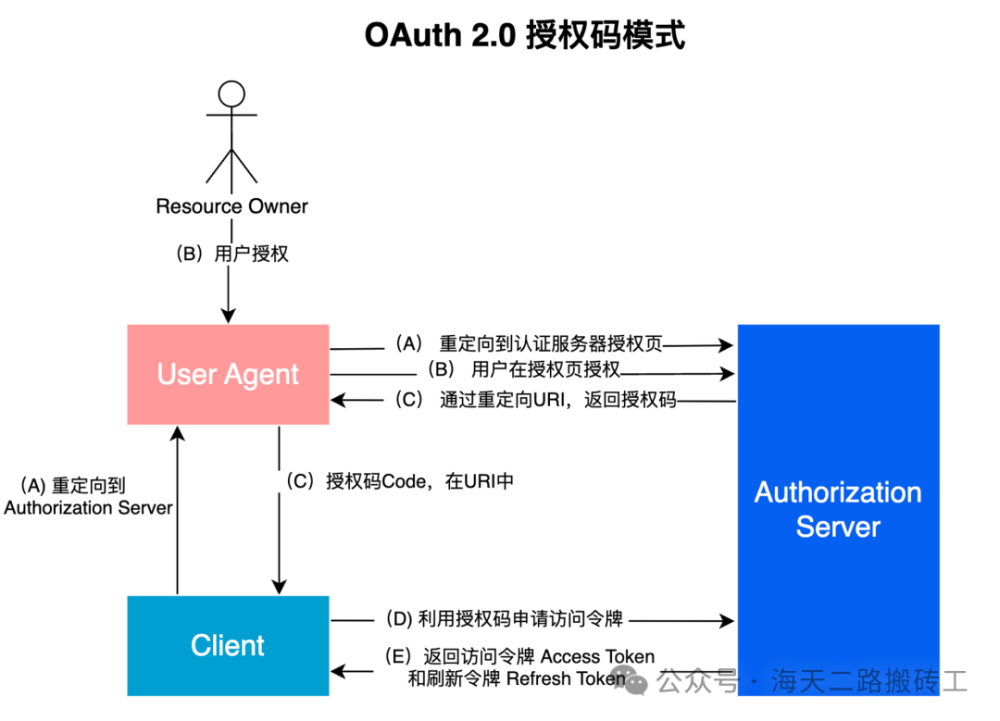
#### 授权码模式

前后端分离

访问令牌不会暴露给用户浏览器，也就避免了令牌泄露的风险

即便授权码被泄露，它也只有短时间的有效期，并且只能在一次交换中使用

授权码模式中，通常会使用状态参数（state）来防止CSRF攻击。



* （A）Client先将页面重定向Authorization Server的授权页；重定向是需要携带授权完毕后要重新打开的页面（携带RedirectURI）。

* （B）Resource Owner在授权页进行授权。

* （C）授权后，Authorization Server将页面重定向会Client的页面（在A步骤中指定的RedirectURI）。同时会在URI中携带授权码Code。授权码Code会经UserAgent最终传递给Client的后端。

* （D）Client（后端）利用授权码向Authorization Server请求访问令牌（Access Token），这里需要指定请求访问的访问Scope等信息。

* （E）Authorization Server 校验授权码通过后，返回访问令牌Access Token和刷新令牌Refresh Token。

#### JWT

头+负载+签名

密钥标识（kid）

OAuth 2.0核心规范没有规定 Access Token 的具体内容，但在实际实现中（如使用JWT作为 Access Token），通常会包含以下信息：

如果 Access Token 使用 JWT 格式（定义在 RFC 7519），则可能包含以下标准声明：

iss（Issuer）：签发 Access Token 的授权服务器的标识符。

sub（Subject）：Access Token 的主体，通常是用户或客户端的唯一标识符。

aud（Audience）：Access Token 的目标受众，通常是资源服务器的标识符。

exp（Expiration Time）：Access Token 的过期时间，通常是一个时间戳。

iat（Issued At）：Access Token 的签发时间，通常是一个时间戳。

scope：Access Token 的作用范围，表示客户端被授予的权限（如read、write等）。

jti（JWT ID）：Access Token 的唯一标识符，用于防止重放攻击。

### 部署

#### 1. 授权服务器配置

授权服务器负责处理授权请求，并颁发访问令牌（Access Token）或刷新令牌（Refresh Token）。主要配置内容包括：

* **授权端点（Authorization Endpoint）**：用于接收客户端的授权请求。客户端通过此端点请求用户授权。

* **令牌端点（Token Endpoint）**：用于客户端请求访问令牌或刷新令牌。

* **客户端认证**：授权服务器需验证客户端的身份，通常通过客户端ID和客户端密钥来完成。

* **重定向URI（Redirect URI）**：客户端在授权完成后，授权服务器会将用户重定向到此URI，附带授权码或令牌。

* **授权类型（Grant Type）**：定义了客户端如何获取授权。常见类型有：

* 1. **授权码（Authorization Code）**：客户端通过授权码换取访问令牌。

* 1. **客户端凭证（Client Credentials）**：用于客户端直接与授权服务器进行交互，通常用于服务间授权。

* 1. **密码凭证（Resource Owner Password Credentials）**：用户提供其用户名和密码，客户端直接获得访问令牌。

* 1. **隐式授权（Implicit）**：适用于SPA等应用，直接返回令牌，避免中间步骤。

#### 2. 客户端配置

客户端是发起授权请求的应用程序，它可以是Web应用、移动应用、桌面应用等。客户端的配置项包括：

* **客户端ID（Client ID）**：唯一标识客户端的字符串，用于授权服务器验证客户端身份。

* **客户端密钥（Client Secret）**：一个私密的密钥，用于与客户端ID一起验证客户端身份。仅应在后端存储，避免泄露。

* **重定向URI（Redirect URI）**：客户端注册时指定的回调URI，授权服务器将在用户授权后重定向回此URI，并附带授权码或令牌。

* **授权范围（Scope）**：定义客户端请求访问的资源范围。范围限制了客户端对资源的访问权限。

* **响应类型（Response Type）**：定义客户端请求的响应类型，常见的是code（授权码）或token（访问令牌）。

#### 3. 资源服务器配置

资源服务器是存储用户数据的服务器，它接收并处理来自客户端的请求。配置项包括：

* **访问令牌验证**：资源服务器需要验证客户端请求中携带的访问令牌是否有效。通常使用JWT或其他令牌格式进行验证。

* **权限控制**：资源服务器需要根据访问令牌中的权限信息（如范围）来决定是否允许访问特定资源。

#### 4. 安全性配置

OAuth 2.0涉及敏感数据（如令牌和用户信息），因此安全配置非常重要：

* **HTTPS**：所有OAuth 2.0通信应通过HTTPS进行加密，以防止中间人攻击和数据泄露。

* **令牌存储**：访问令牌和刷新令牌应妥善存储，避免泄露。常见存储方式有加密数据库或安全的存储解决方案。

* **状态参数（State）**：防止CSRF攻击，客户端在授权请求中附带一个随机生成的state参数，授权服务器返回时会带上该参数，客户端用以验证请求的合法性。

#### 5. 用户授权配置

用户授权是OAuth 2.0的核心环节，涉及到以下配置：

* **用户认证**：授权服务器需要确认用户的身份，通常通过用户名和密码、双因素认证等方式。

* **授权同意界面**：用户在授权时会看到一个界面，确认是否同意授权客户端访问其资源。该界面应显示请求的权限范围，并明确告知用户哪些数据将被共享。

#### 6. 刷新令牌配置

刷新令牌用于在访问令牌过期时获取新的访问令牌。相关配置包括：

* **刷新令牌过期时间**：设置刷新令牌的有效期，超过此时间则刷新令牌失效，需要重新授权。

* **单次刷新令牌使用限制**：为了防止刷新令牌泄露后被滥用，可以设置每个刷新令牌只能使用一次。

#### 7. 日志与监控

为了确保OAuth 2.0协议的安全性与稳定性，需要配置日志和监控：

* **授权请求日志**：记录每次授权请求，包括客户端信息、用户授权情况、请求的范围等。

* **令牌颁发日志**：记录令牌的颁发情况，包括令牌的生命周期、使用情况、失效时间等。

* **安全警报**：对于异常的授权请求、令牌使用情况、客户端行为等，要设置监控和报警机制。

## OIDC

TODO

**OpenID Connect对OAuth的主要扩展就是引入了ID Token**

OpenID Connect（OIDC）是由 OpenID 基金会开发的一种身份层协议，它建立在 OAuth2.0 框架之上，旨在提供一种标准化的方式来验证用户身份并获取其基本信息。与单纯的 OAuth2.0 不同，OIDC 不仅关注于授权（即允许应用程序访问用户在其他服务上的资源），更强调身份验证——确认“你是谁”。OIDC 引入了 ID Token 的概念，这是一种包含用户身份信息的JWT（JSON Web Token），使得应用可以确信“谁”正在访问，而不仅仅是可以访问什么。

### 介绍

为了更加容易理解OIDC，我们先来回一下OAuth2.0所包含的4个角色：

1：受保护资源的拥有者

一般是我们自己

2：受保护的资源

一般就是HTTP的接口形式的API，当然也可以是其他形式

3：三方软件

一般是希望拿到受保护资源的角色

4：授权服务器

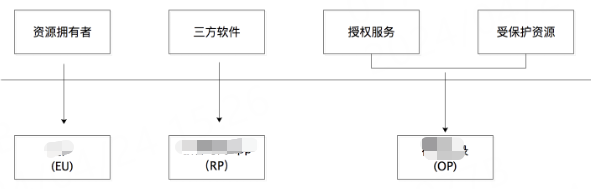
给客户端完成授权的角色

OIDC要做的事情是用户通过认证服务器来证明自己的身份，从而访问三方服务，所以OIDC包含如下3个角色：

用户：规范中叫End User，即EU

认证服务器：规范中叫OpenID Provider，即OP

三方服务：规范中叫relying party,依赖方，即RP



### 流程

OIDC 的工作流程大致可以分为以下步骤：

1. **用户请求访问 RP**：用户尝试访问依赖方（RP）提供的受保护资源。

1. **重定向至 IdP**：RP 将用户重定向到预先配置的身份提供商（IdP）进行登录。

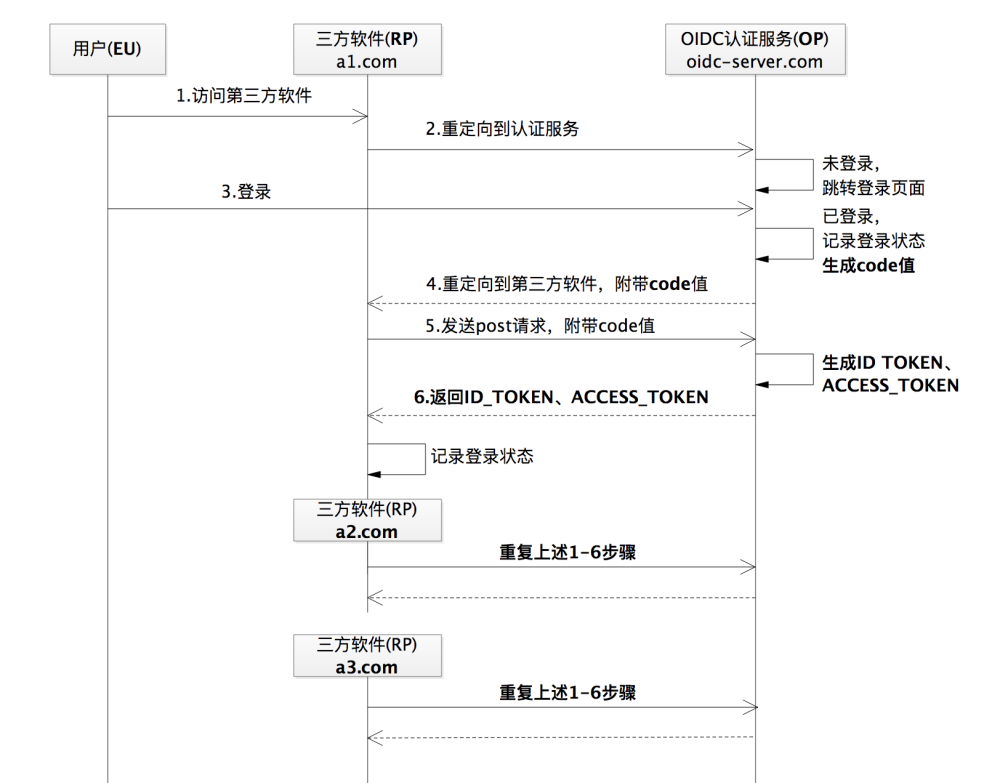
1. **用户身份验证**：用户在 IdP 上输入凭证完成身份验证。

1. **授权码发放**：IdP 向用户代理（通常是浏览器）返回一个授权码，并附带 RP 的重定向 URI。

1. **RP 交换令牌**：RP 通过后端服务器向 IdP 发送授权码，请求换取访问令牌和 ID 令牌。

1. **验证 ID 令牌**：RP 验证 ID 令牌的有效性（签名、过期时间等），并提取用户信息。

1. **访问资源**：验证成功后，RP 允许用户访问受保护资源。



**请求参数 prompt 的影响**：

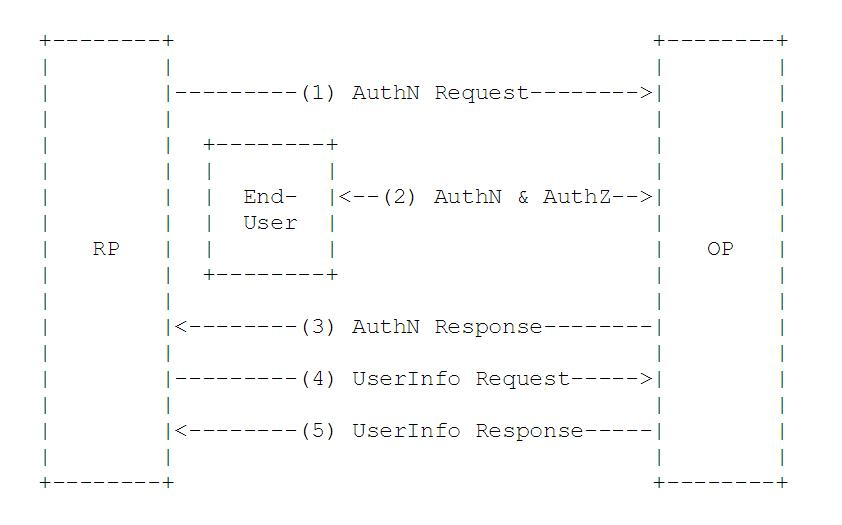
* 在OIDC的认证请求中，**prompt** 参数控制是否强制用户进行认证或授权。如果prompt=none，那么 IdP 将不会要求用户重新进行认证或授权。如果用户未登录且prompt=none，则会返回错误。

* prompt 参数的其他选项：

* 1. **prompt=login**：强制用户登录。

* 1. **prompt=consent**：强制用户进行授权（同意授权）。

* 1. **prompt=select\_account**：如果用户有多个账户，则要求选择账户。



上图取自Core规范文档，其中AuthN=Authentication，表示认证；AuthZ=Authorization，代表授权。注意这里面RP发往OP的请求，是属于**Authentication**类型的请求，虽然在OIDC中是复用OAuth2的**Authorization**请求通道，但是用途是不一样的，且OIDC的AuthN请求中**scope参数必须要有一个值为的openid的参数**（后面会详细介绍AuthN请求所需的参数），用来区分这是一个OIDC的**Authentication**请求，而不是OAuth2的**Authorization**请求。

### 配置

在 **OIDC（OpenID Connect）** 部署时，配置工作涉及到 **身份提供者（IdP）** 和 **服务提供者（RP，Relying Party）** 之间的信任关系、认证和授权流程等。以下是OIDC部署时主要的配置内容：

#### 1. 身份提供者（IdP）配置

身份提供者是用户认证的中心，负责颁发和签发认证令牌（ID Token）及访问令牌（Access Token）。配置时需要关注以下内容：

##### a. IDP元数据

* **元数据文件**：IdP生成一个元数据文件，包含其身份、证书、授权端点（授权URL）、令牌端点等信息。RP通过此文件了解IdP的具体配置。

* 1. **授权端点**：用于处理身份验证请求的URL，通常是 https://idp.example.com/authorize。

* 1. **令牌端点**：用于获取Access Token和ID Token的URL，通常是 https://idp.example.com/token。

* 1. **用户信息端点**：用于获取用户信息的URL，通常是 https://idp.example.com/userinfo。

* 1. **JWK端点**：用于提供公钥，RP使用公钥验证ID Token的签名，通常是 https://idp.example.com/.well-known/jwks.json。

* 1. **登出端点**：用于处理单点注销请求的URL，通常是 https://idp.example.com/logout。

##### b. 客户端应用配置

* **客户端ID**：在IdP中为每个RP（客户端应用）注册一个唯一的客户端ID。

* **客户端密钥**：IdP为每个客户端应用分配一个密钥，用于在授权码流程中验证客户端身份。

* **重定向URI（Redirect URI）**：RP在发起认证请求时指定的URI，IdP会将认证结果（包括Authorization Code）重定向到此URI。

* **授权范围（Scopes）**：IdP需要配置支持的授权范围，如 openid（标识用户身份）、profile（用户个人资料信息）、email（用户邮箱）等。范围指定了RP可以访问哪些用户信息。

##### c. 证书与密钥管理

* **公钥/私钥对**：IdP需要配置用于签名ID Token的私钥，同时公开其公钥，供RP用来验证ID Token的签名。

* **JWK（JSON Web Key）**：IdP发布一个JWK集合，RP通过此集合获取公钥，确保签名验证的正确性。

##### d. 授权码流程（Authorization Code Flow）

* IdP需要支持 **授权码流程**，该流程要求用户进行认证后，通过授权码交换获取ID Token和Access Token。

##### e. 支持多种授权类型

* 支持 **PKCE（Proof Key for Code Exchange）**：为了加强安全性，IdP可以支持PKCE，尤其是针对移动应用等不安全的客户端，防止授权码拦截攻击。

* 支持 **隐式流程（Implicit Flow）** 和 **资源所有者密码凭证流程（Resource Owner Password Credentials Flow）**，但由于隐式流程的安全问题，现代应用多采用授权码流程。

#### 2. 服务提供者（RP）配置

RP是需要身份认证和授权的应用程序。RP的配置通常包括以下内容：

##### a. 注册RP

* **客户端ID和密钥**：RP需要在IdP上注册并获取一个唯一的 **客户端ID** 和 **客户端密钥**，这将用于与IdP通信，验证其身份。

* **授权请求和令牌请求的URI**：RP需要配置与IdP交互的端点，包括 **授权请求端点**（通常是 authorize）、**令牌请求端点**（通常是 token）和 **用户信息端点**（通常是 userinfo）。

##### b. 授权请求

RP需要发起授权请求时，发送以下信息：

* **client\_id**：RP的客户端ID，用来标识RP。

* **redirect\_uri**：认证成功后，IdP将用户重定向回此URI。

* **response\_type**：指定响应类型，通常为 code（授权码）。

* **scope**：请求访问的范围，如 openid profile email。

* **state**：防止CSRF攻击的随机值，用于验证授权响应。

* **nonce**：防止重放攻击的随机值，用于ID Token。

授权请求示例：

plaintext  
  
  
复制编辑  
https://idp.example.com/authorize?  
 response\_type=code&  
 client\_id=your-client-id&  
 redirect\_uri=https://rp.example.com/callback&  
 scope=openid profile email&  
 state=abc123&  
 nonce=xyz456

##### c. 令牌请求

在获得授权码后，RP需要向IdP的令牌端点发送请求，交换获取Access Token和ID Token。

* **grant\_type**：authorization\_code。

* **code**：从授权请求中返回的授权码。

* **redirect\_uri**：与授权请求中使用的URI匹配。

令牌请求示例：

plaintext  
  
  
复制编辑  
POST https://idp.example.com/token  
Content-Type: application/x-www-form-urlencoded  
  
grant\_type=authorization\_code&  
code=authorization\_code\_received&  
redirect\_uri=https://rp.example.com/callback&  
client\_id=your-client-id&  
client\_secret=your-client-secret

##### d. ID Token验证

RP需要验证ID Token的签名，以确认其来自可信的IdP。验证时RP应检查以下内容：

* **签名**：使用IdP提供的公钥验证ID Token的签名。

* **iss（Issuer）**：确认令牌的发布者是否为预期的IdP。

* **aud（Audience）**：确认令牌的受众是否为RP。

* **exp（Expiration）**：确认ID Token是否过期。

* **nonce**：确认ID Token中的nonce与授权请求中的nonce匹配。

##### e. 用户信息获取

RP可以使用Access Token访问IdP的 **userinfo端点**，获取用户的详细信息（如姓名、电子邮件等）。请求方式通常是：

plaintext  
  
  
复制编辑  
GET https://idp.example.com/userinfo  
Authorization: Bearer access\_token

##### f. 注销配置

* RP和IdP需要协同配置 **单点注销（Single Logout, SLO）**，以便在用户从一个应用中注销时，可以同步在所有关联应用中注销用户。

* RP向IdP发送注销请求，IdP会通知所有相关应用执行注销操作。

#### 3. 安全性配置

##### a. PKCE（Proof Key for Code Exchange）

* PKCE 用于防止授权码拦截攻击，RP在授权请求时生成一个随机的**code\_verifier**，并通过**code\_challenge**与授权请求一起发送。IdP在令牌请求时检查这个**code\_verifier**与**code\_challenge**是否匹配。

##### b. HTTPS（安全通信）

* OIDC要求RP和IdP之间的通信必须通过 **HTTPS** 加密，以保证数据在传输过程中不被窃取或篡改。

##### c. CSRF防护

* 在认证请求中使用 **state** 参数来防止CSRF攻击，确保在重定向回RP时验证该值。

#### 总结

部署OIDC时的主要配置内容包括：

* 配置IdP的客户端应用注册、证书、端点、授权范围等信息。

* 配置RP的客户端ID、重定向URI、授权请求、令牌请求、用户信息获取等。

* 配置安全性选项（如PKCE、HTTPS、CSRF防护）。

* 确保会话管理和单点注销等功能的协同工作。

这些配置确保OIDC的认证和授权流程安全、可靠，并且能够满足现代Web应用和移动应用的需求。

## 区别

SAML和OAuth的主要区别在于它们的设计目标和使用场景。SAML主要关注于实现单点登录，而OAuth则关注于授权访问第三方应用程序的权限。此外，SAML使用XML进行数据交换，而OAuth使用多用JWT。

**SAML的一个重要的应用场景就是单点登录（SSO）**。虽然OAuth也可用于SSO，但记住OAuth协议的初衷是解决委托授权（delegated authorization）的问题。它们解决单点登录（SSO）的工作流程很类似。区别在于SAML采用XML格式，而OAuth采用JSON传输信息。SAML更适合于企业用户，而对于移动设备则OAuth更合适。

在SAML协议中，SAML token中已经包含了用户身份信息，但是在OAuth2，在拿到token之后，需要额外再做一次对该token的校验。

SAML的优点在于协议流程简单，信息交换全部由前端浏览器完成，安全性通过HTTP重定向和POST请求保证。然而，SAML需要通过HTTP重定向和POST协议传递信息，这限制了其在非Web应用程序中的应用，如手机应用。

OAuth2.0是一个开放的协议，可以很灵活的扩展，而SAML就可以用来扩展OAuth。SAML2.0对OAuth2.0的扩展体现在两个方面。

**SAML2.0对OAuth2.0的第一个扩展就是对授权类型（Authorization Grant）的扩展**。前面也已经说过，OAuth2.0支持四种授权类型。作为一种扩展，SAML assertion也可以作为授权类型。这时，在client向authorization server (token endpoint)请求access token时，可以将参数grant\_type设置成：

urn:ietf:params:oauth:grant-type:saml2-bearer

**SAML2.0对OAuth2.0的第二个扩展就是client身份验证方式的扩展**。client向authorization server (token endpoint)请求access token时，可以提供client\_id和client\_secret证明自己的身份。作为一种扩展，SAML assertion也可以用来验证client的身份，这样client就无需提供secret了。

### 总结：SAML vs OAuth 2.0 vs OIDC

|  |  |  |  |
| --- | --- | --- | --- |
| 特性/协议 | **SAML** | **OAuth 2.0** | **OIDC** |
| **协议类型** | 身份验证协议 | 授权协议 | 身份验证和授权协议 |
| **数据格式** | XML | JSON | JSON（主要是 JWT） |
| **适用场景** | 单点登录（SSO），企业级认证 | 第三方授权，API授权 | 用户身份验证+授权，适用于 Web 和移动应用 |
| **主要参与者** | 身份提供者（IdP），服务提供者（SP） | 授权服务器，资源服务器，客户端 | 身份提供者（IdP），依赖方（RP），用户 |
| **用户身份信息** | 通过 SAML Assertion 提供 | 不完全标识（sub字段） | 通过 ID Token 提供 |
| **支持的令牌** | 无 | Access Token（授权令牌） | Access Token 和 ID Token |
| **认证方式** | 通过 SAML Assertion 进行身份验证 | 无（仅授权） | 通过 ID Token 进行身份验证 |
| **跨域支持** | 支持跨域认证 | 支持跨域授权 | 支持跨域身份验证与授权 |
| **安全性** | 基于签名的 XML | 依赖于 HTTPS 和令牌的有效性 | 基于 JWT 的签名和加密 |

### 安全性对比

两者各有优缺点，具体安全性取决于使用场景和实现方式。下面是主要的安全对比：

|  |  |  |
| --- | --- | --- |
| **安全性维度** | **SAML** | **OIDC** |
| **数据加密** | SAML 断言可以加密（**但通常只是签名，不加密**） | ID Token & Access Token **通常是 JWT，支持加密和签名** |
| **数据完整性** | SAML 断言使用 **XML 数字签名**，可防篡改 | JWT 使用 **JWS（JSON Web Signature）**，确保完整性 |
| **传输安全性** | SAML 断言 **通过浏览器传输**，容易被劫持或泄露 | OIDC **令牌通过服务器端 API 传输**，更安全 |
| **Token 保护** | SAML 断言 **只能使用一次**，无法存储或复用 | OIDC 的 Access Token **可存储**，存在被盗风险（但支持短期有效性） |
| **Session 管理** | SAML 依赖浏览器 Cookie，无法主动注销 | OIDC 支持 **Token 过期 & Refresh Token**，更灵活 |
| **漏洞风险** | SAML 受 **XML 相关攻击（XML Injection, Signature Wrapping）** 影响 | OIDC 可能受到 **Token 盗窃、Token Replay** 攻击 |
| **双因素认证支持** | SAML 可用于 MFA，但需要额外配置 | OIDC **与 OAuth 2.0 结合，支持 MFA 和设备身份验证** |

### 数据传输优化

🟢 **优化点**：

* **敏感数据（ID Token & Access Token）不经过浏览器，而是通过后端 API 交换**。

* **ID Token（JWT）签名更轻量级，且支持多种加密方式**，相比 SAML 的 XML 签名更易于处理。

* **Access Token 允许 API 授权**，支持细粒度权限管理，而 SAML 只提供身份认证。

✅ **结论：OIDC 比 SAML 更安全，主要优化点包括**：

1. **OIDC 避免了浏览器直接传输身份数据**：

* 1. SAML 断言（身份信息）通过浏览器 POST 传输，有泄露风险。

* 1. OIDC 只在浏览器传输 **授权码**（无敏感信息），令牌交换在 **后端服务器完成**，更安全。

1. **OIDC 使用 JWT（JSON Web Token），比 SAML XML 解析更安全**：

* 1. SAML 使用 **XML 及数字签名**，容易受到 **XML 相关攻击（如 XML 署名绕过）**。

* 1. OIDC 使用 **JWT（JSON）+ JWS 签名**，结构更简单，不易受 XML 攻击。

1. **OIDC 结合 OAuth 2.0，支持更细粒度的安全控制**：

* 1. OIDC 允许 **短生命周期令牌（Access Token）**，避免长期有效的 SAML 断言泄露风险。

* 1. OIDC 还可以支持 **Refresh Token 机制**，增强会话管理，而 SAML 需要额外配置。

### SAML 的主要优势

尽管 OIDC 在数据传输安全性、集成便利性和现代 Web 兼容性上优于 SAML，但 **SAML 仍然在某些应用场景中具备独特优势**，尤其是在 **企业级身份管理和传统 IT 系统** 中。

#### ✅ 1. 更适用于企业级单点登录（SSO）

* **企业、政府机构、大型组织** 更倾向于使用 **SAML 作为 SSO 方案**，因为：

* 1. SAML 是 **单点登录（SSO）行业标准**，已经被 **微软 AD FS、Okta、Ping Identity、OneLogin** 等企业级身份提供者（IdP）广泛采用。

* 1. 企业中常见的 **内部 Web 应用（如 SAP、Oracle、Workday、Salesforce、Google Workspace、Microsoft 365）** 都支持 SAML 进行单点登录。

* 1. 许多 **传统企业 IT 系统** 依赖 SAML **身份联合（Federated Identity）**，因为它可以与 **Active Directory（AD）** 无缝集成。

📌 **为什么这点重要？**

* 许多企业仍然使用 **Windows Active Directory（AD）**，SAML **比 OIDC 更容易与 AD 集成**。

* SAML 的 **企业适配性更强**，适合复杂的企业 IT 环境（如混合云、私有云）。

* **企业 SAML SSO** 配置后可以大规模应用于 **数百个 Web 应用**，简化身份管理。

#### ✅ 2. SAML 适用于浏览器单点登录（Browser-based SSO）

* **SAML 完全基于浏览器工作（Redirect + HTTP POST），更适合 Web 端 SSO**：

* 1. OIDC 需要 **客户端 ID、授权码、服务器端令牌交换**，对单纯的 SSO 可能过于复杂。

* 1. SAML **不需要客户端存储 Token**，适用于 **多浏览器环境**（如公司员工在不同设备上登录）。

📌 **实际案例**：

* 企业办公 Web 应用（如 SAP、Salesforce）通常不需要 API 授权，只需要 SSO，**SAML 更适合**。

* 用户只需要 **打开浏览器访问应用，SAML 自动完成身份认证**，而 **OIDC 需要额外的后端令牌管理**。

#### ✅ 3. SAML 断言携带的信息更丰富

* SAML 断言（Assertion）可以包含：

* 1. **用户身份信息（NameID, Email, UID）**

* 1. **用户属性（Role, Department, Security Group）**

* 1. **认证上下文（Authentication Context，如 MFA 是否启用）**

* 1. **授权信息（但不适用于 API 访问）**

📌 **相比之下，OIDC 主要提供**：

* ID Token 仅包含基础用户信息（如 sub、email）。

* 如果需要用户属性，OIDC 需要额外调用 **UserInfo Endpoint**，而 SAML 直接在 **SAML 断言里提供完整信息**。

#### OIDC 相比 SAML 更好的地方

✅ **OIDC 更适合现代 Web 和移动应用**

* OIDC 适用于 **移动端、单页应用（SPA）、API 访问**，SAML **不适用于移动端**。

* OIDC 使用 **OAuth 2.0 授权机制**，支持 **细粒度 API 访问控制**，SAML **不能用于 API 访问**。

✅ **OIDC 安全性更高**

* SAML 断言 **在浏览器中传输，可能被拦截**，而 OIDC **令牌在服务器端交换**。

* OIDC 允许 **短生命周期 Access Token，避免长期 SAML 断言的泄露风险**。

✅ **OIDC 部署更轻量**

* SAML 需要 **复杂的 XML 解析**，OIDC 采用 **JSON & REST API，易于集成**。

* SAML 认证流程需要 **SP & IdP 预先交换元数据**，OIDC 仅需 **Client ID & Secret**。

## 网址资料

oauth

<https://zhuanlan.zhihu.com/p/717770658>

<https://openid.net/specs/openid-connect-core-1_0.html#ThirdPartyInitiatedLogin>

<https://blog.csdn.net/iamlake/article/details/93415206>

<https://blog.csdn.net/xiaoyi52/article/details/136370402>

<https://blog.csdn.net/weixin_43333483/article/details/126256938> PKCE

saml

<https://blog.csdn.net/u012613251/article/details/137099294>

<https://baijiahao.baidu.com/s?id=1725446632667720841&wfr=spider&for=pc>

<https://baijiahao.baidu.com/s?id=1825077970285524431&wfr=spider&for=pc>

<https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf> SAML标准

<https://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf> SAML消息传输绑定标准