# 【深度知识】JSON Web令牌(JWT)的原理，流程和数据结构

2019-07-10阅读 11.1K0



# **1. 摘要**

JSON Web Token（缩写 JWT）是目前最流行的跨域认证解决方案，本文介绍它的原理，用法和详细的数据结构。

# **2. JWT的定义**

Json web token（JWT）是为了网络应用环境间传递声明而执行的一种基于JSON的开发标准（RFC 7519），该token被设计为紧凑且安全的，特别适用于分布式站点的单点登陆（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

****什么情况下使用JWT比较适合？**** ****授权：****这是最常见的使用场景，解决单点登录问题。因为JWT使用起来轻便，开销小，服务端不用记录用户状态信息（无状态），所以使用比较广泛； ****信息交换：****JWT是在各个服务之间安全传输信息的好方法。因为JWT可以签名，例如，使用公钥/私钥对儿 - 可以确定请求方是合法的。此外，由于使用标头和有效负载计算签名，还可以验证内容是否未被篡改。

# **3. JWT的原理和流程**

## **3.1 跨域认证的问题**

互联网服务离不开用户认证。一般流程是下面这样： 1、用户向服务器发送用户名和密码。 2、服务器验证通过后，在当前对话（session）里面保存相关数据，比如用户角色、登录时间等等。 3、服务器向用户返回一个 session\_id，写入用户的 Cookie。 4、用户随后的每一次请求，都会通过 Cookie，将 session\_id 传回服务器。 5、服务器收到 session\_id，找到前期保存的数据，由此得知用户的身份。

这种模式的问题在于，扩展性（scaling）不好。单机当然没有问题，如果是服务器集群，或者是跨域的服务导向架构，就要求 session 数据共享，每台服务器都能够读取 session。

举例来说，A 网站和 B 网站是同一家公司的关联服务。现在要求，用户只要在其中一个网站登录，再访问另一个网站就会自动登录，请问怎么实现？

一种解决方案是 session 数据持久化，写入[数据库](https://cloud.tencent.com/solution/database?from=10680" \t "https://cloud.tencent.com/developer/article/_blank)或别的持久层。各种服务收到请求后，都向持久层请求数据。这种方案的优点是架构清晰，缺点是工程量比较大。另外，持久层万一挂了，就会单点失败。 另一种方案是服务器索性不保存 session 数据了，所有数据都保存在客户端，每次请求都发回服务器。JWT 就是这种方案的一个代表。

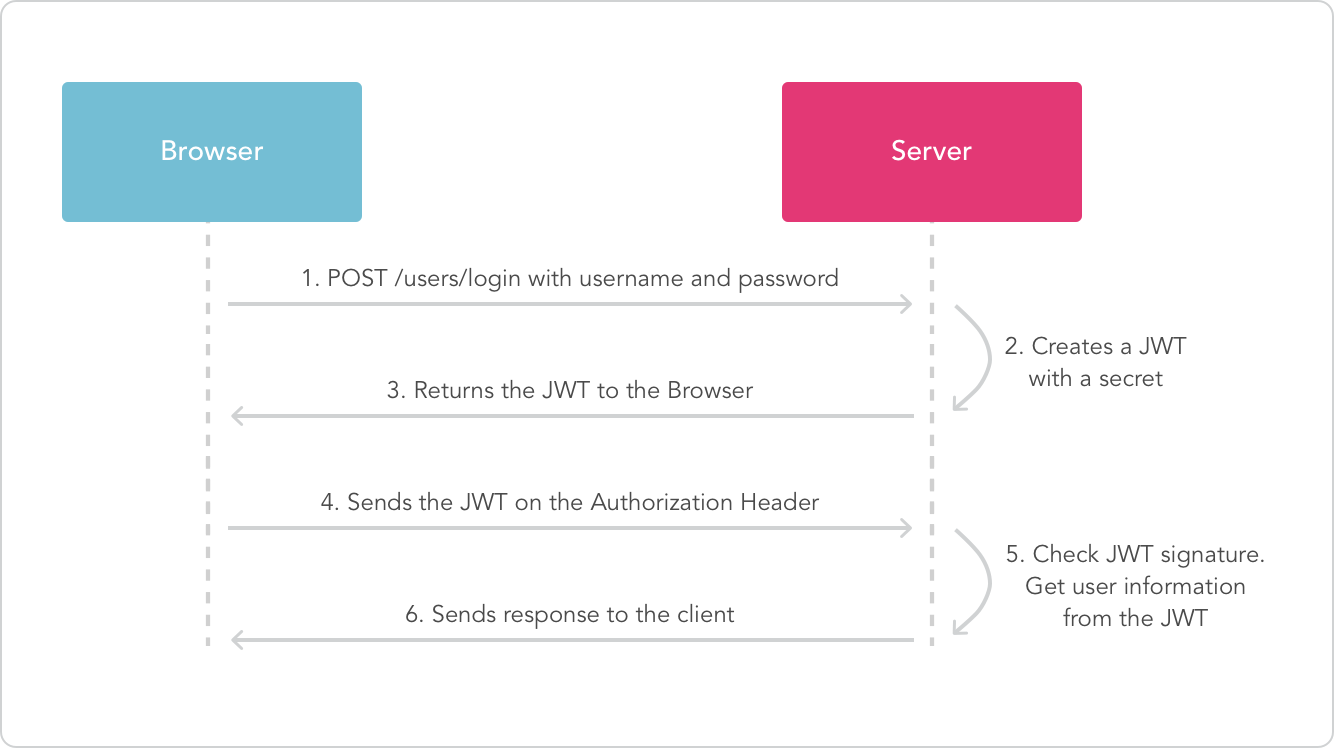
## **3.2 JWT 的原理**

JWT 的原理是，服务器认证以后，生成一个 JSON 对象，发回给用户，就像下面这样。 { "姓名": "张三", "角色": "管理员", "到期时间": "2018年7月1日0点0分" } 以后，用户与服务端通信的时候，都要发回这个 JSON 对象。服务器完全只靠这个对象认定用户身份。为了防止用户篡改数据，服务器在生成这个对象的时候，会加上签名。

****服务器就不保存任何 session 数据了，也就是说，服务器变成无状态了，从而比较容易实现扩展。****

****区别**** (1) session 存储在服务端占用服务器资源，而 JWT 存储在客户端 (1) session 存储在 Cookie 中，存在伪造跨站请求伪造攻击的风险 (2) session 只存在一台服务器上，那么下次请求就必须请求这台服务器，不利于分布式应用 (3) 存储在客户端的 JWT 比存储在服务端的 session 更具有扩展性

## **3.3 JWT的认证流程图**



****流程说明：**** 1，浏览器发起请求登陆，携带用户名和密码； 2，服务端验证身份，根据算法，将用户标识符打包生成 token, 3，服务器返回JWT信息给浏览器，JWT不包含敏感信息； 4，浏览器发起请求获取用户资料，把刚刚拿到的 token一起发送给服务器； 5，服务器发现数据中有 token，验明正身； 6，服务器返回该用户的用户资料；

## **3.4 JWT的6个优缺点**

1、JWT默认不加密，但可以加密。生成原始令牌后，可以使用改令牌再次对其进行加密。 2、当JWT未加密方法是，一些私密数据无法通过JWT传输。 3、JWT不仅可用于认证，还可用于信息交换。善用JWT有助于减少服务器请求数据库的次数。 4、JWT的最大缺点是服务器不保存会话状态，所以在使用期间不可能取消令牌或更改令牌的权限。也就是说，一旦JWT签发，在有效期内将会一直有效。 5、JWT本身包含认证信息，因此一旦信息泄露，任何人都可以获得令牌的所有权限。为了减少盗用，JWT的有效期不宜设置太长。对于某些重要操作，用户在使用时应该每次都进行进行身份验证。 6、为了减少盗用和窃取，JWT不建议使用HTTP协议来传输代码，而是使用加密的HTTPS协议进行传输。

# **4. JWT 的数据结构**

## **4.1 JWT消息构成**

一个token分3部分，按顺序:

* 头部（header)
* 载荷（payload)
* 签证（signature) 对象为一个很长的字符串，字符之间通过"."分隔符分为三个子串。注意JWT对象为一个长字串，各字串之间也没有换行符，一般格式为：xxxxx.yyyyy.zzzzz 。 例如 yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c



## **4.2 头部（header)**

JWT的头部承载两部分信息：

* 声明类型，这里是jwt
* 声明加密的算法 通常直接使用 HMAC SHA256

JWT里验证和签名使用的算法，可选择下面的：

| JWS | 算法名称 | 描述 |
| --- | --- | --- |
| HS256 | HMAC256 | HMAC with SHA-256 |
| HS384 | HMAC384 | HMAC with SHA-384 |
| HS512 | HMAC512 | HMAC with SHA-512 |
| RS256 | RSA256 | RSASSA-PKCS1-v1\_5 with SHA-256 |
| RS384 | RSA384 | RSASSA-PKCS1-v1\_5 with SHA-384 |
| RS512 | RSA512 | RSASSA-PKCS1-v1\_5 with SHA-512 |
| ES256 | ECDSA256 | ECDSA with curve P-256 and SHA-256 |
| ES384 | ECDSA384 | ECDSA with curve P-384 and SHA-384 |
| ES512 | ECDSA512 | ECDSA with curve P-521 and SHA-512 |

JWT的头部描述JWT元数据的JSON对象参考： { "alg": "HS256", "typ": "JWT" }

代码样例如下:

// header Map

Map<String, Object> map = new HashMap<>();

map.put("alg", "HS256");

map.put("typ", "JWT");

复制

## **4.3 载荷（payload)**

Payload 部分也是一个 JSON 对象，用来存放实际需要传递的数据。JWT 规定了7个官方字段，供选用。

iss (issuer)：签发人 exp (expiration time)：过期时间 sub (subject)：主题 aud (audience)：受众 nbf (Not Before)：生效时间 iat (Issued At)：签发时间 jti (JWT ID)：编号

除以上默认字段外，我们还可以自定义私有字段，如下例： { "sub": "1234567890", "name": "chongchong", "admin": true } 注意，JWT 默认是不加密的，任何人都可以读到，所以不要把秘密信息放在这个部分。 这个 JSON 对象也要使用 Base64URL 算法转成字符串。 代码样例如下:

JWT.create().withHeader(map) // header

.withClaim("iss", "Service") // payload

.withClaim("aud", "APP")

.withIssuedAt(iatDate) // sign time

.withExpiresAt(expiresDate) // expire time

.withClaim("name", "cy") // payload

.withClaim("user\_id", "11222");

复制

## **4.4 签名（signature)**

Signature 部分是对前两部分的签名，防止数据篡改。 首先，需要指定一个密钥（secret）。这个密钥只有服务器才知道，不能泄露给用户。然后，使用 Header 里面指定的签名算法（默认是 HMAC SHA256），按照下面的公式产生签名。

HMACSHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

算出签名以后，把 Header、Payload、Signature 三个部分拼成一个字符串，每个部分之间用"点"（.）分隔，就构成整个JWT对象TOKEN， 就可以返回给用户。

### **4.4.1 Base64URL算法**

前面提到，Header 和 Payload 串型化的算法是 Base64URL。这个算法跟 Base64 算法基本类似，但有一些小的不同。 JWT 作为一个令牌（token），有些场合可能会放到 URL（比如 api.example.com/?token=xxx）。Base64 有三个字符+、/和=，在 URL 里面有特殊含义，所以要被替换掉：=被省略、+替换成-，/替换成\_ 。这就是 Base64URL 算法。

## **4.5 JWT的用法**

客户端接收服务器返回的JWT，将其存储在Cookie或localStorage中。 此后，客户端将在与服务器交互中都会带JWT。如果将它存储在Cookie中，就可以自动发送，但是不会跨域，因此一般是将它放入HTTP请求的Header Authorization字段中。

Authorization: Bearer <token>

当跨域时，也可以将JWT被放置于POST请求的数据主体中。

# **5. JWT、JWS、JWE的区别**

1）JWT(JSON Web Tokens)，jwt长度较小，且可以使用URL传输(URL safe)。不想cookies只能在web环境起作用。 JWT可以同时使用在web环境和RESTfull的接口。

2)对于开发者来说，JWT与另外两种相近的标准:JWS(JSON Web Signature)、JWE(JSON Web Encryption)，容易引起混乱。

3)关于JWT的描述，可以参考RFC7519([https://tools.ietf.org/html/rfc7519](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc7519" \t "https://cloud.tencent.com/developer/article/_blank))的描述: \*\*JSON Web Token (JWT) \*\*是一个间接地、URL安全的，表现为一组声明，可以在双方之间进行传输。一个JWT的声明，是指经过编码后的一个JSON对象，这个JSON对象可以是一个JSON Web Signature(JWS)结构的荷载(payload)，或者是一个JSON Web Encryption(JWE)结构的明文。允许使用声明进行数字签名，或者通过一个Message Authentication Code(MAC)进行完整性保护可选择是否加密。

简单来说，JWTs表现为一组被编码为JWS and/or JWE结构的JSON object的声明(Claim).

换言之，一组JWT声明(就是表现为JSON格式的Claims)被通过JWS结构或者JWE结构(或者同时使用两种)发送，决定于你如何去实现它。所以，当你发送JWT给别人是，它实际上是一个JWT荷载或者JWE荷载。JWS荷载更加常用。

4)关于JWS 顾名思义，JWS模式对这个内容进行了数字化签名。这个内容被用来存放JWT的声明.服务端签名出JWT并且发送到客户端，并在用户成功认证后进行应答。服务器期望客户端在下次请求的时候将JWS作为请求的一部分，发送回服务端。

如果我们处理的客户端是欺骗者怎么办呢？这就是签名(signature)需要出场的地方了。签名携带了完整的可验证的信息。换句话说，服务器可以确认，接收到的JWT声明里的JWS是没有经过欺骗客户端、中间者进行修改的。 服务端通过验证消息的签名来确保客户端没有修改声明。如果服务端检测到任何修改，可以采取适当的动作(拒绝这次请求或者锁定客户端之类的)。 客户端同样可以验证签名，为了做到这点，客户端也需要服务端的secret(密钥)(如果这个JWT签名是HMAC算法),或者需要服务端对公钥(如果这个WJT是数字化签名)。 特别注意：对于JWS，荷载(声明部分)没有进行加密，所以，不要发送任何敏感信息。

5)关于JWE JWE模式会对内容加密，而不是签名。JWT的声明会被加密。因此JWE带来了保密性。JWE可以被签名并附在JWS里。这样的话就可以同时加密和签名。因此得到了保密性(Confidentiality)、完整性(Integrity)、可认证(Authentication)。

6)那么对于客户端，如何分辨JWS或者JWE呢？ JWS的Header与JWE的Header是不同的，可以通过检查“alg”Header参数的值来区分。如果这个值表现为一个数字签名或者MAC的算法，或者是”none“，则它是一个JWS。 如果它表现为一个 Key Encryption, Key Wrapping, Direct Key Agreement, Key Agreement with Key Wrapping, or Direct Encryption algorithm。则它是一个JWE。 还可以通过Header里的“enc”(encryption algorithm)是否存在来判断，如果"enc"这个成员存在的话说明是JWE，否则的话就是JWS.